SpaceWire 2013

Fifth International Conference
10th - 14th of June 2013
Gothenburg, Sweden

Prof. Steve Parkes, University of Dundee, Scotland
Martin Suess, European Space Agency
Sandi Habinc, Aeroflex Gaisler, Sweden

2013.spacewire-conference.org

SpaceWire-2013

Proceedings of the 5th International SpaceWire Conference

Gothenburg 2013

Editors: Steve Parkes and Carole Carrie

Space
Technology
Centre
University of Dundee

SpaceWire-2013

Proceedings of International SpaceWire Conference

Gothenburg 2013

ISBN: 978-0-9557196-4-6

Space Techology Centre
University of Dundee

Preface

These proceedings contain the papers presented at the 2013 International SpaceWire Conference, held in the Radisson Blu Scandinavia Hotel, Gothenburg, Sweden, between 10 and 13 June, 2013. The International SpaceWire Conference aims to bring together SpaceWire product designers, hardware engineers, software engineers, system developers and mission specialists interested in and working with SpaceWire to share the latest ideas and developments related to SpaceWire technology. SpaceWire technology is now being used or designed into over one hundred spacecraft, covering science, exploration Earth observation and commercial applications. High profile missions like James Webb Space Telescope, Astro-H, GAIA, ExoMars, Bepicolombo, Sentinels 1, 2, 3 and 5 precursor, and GOES-R are using SpaceWire extensively. SpaceWire is being used in Europe, Japan, USA, Russia, China, India, and other countries of the World.

The conference covers many different aspects of SpaceWire technology and includes both academic and industrial presentations. Sessions address recent developments of the SpaceWire set of standards, space missions and other applications using SpaceWire, new components, sensors and cables which support the SpaceWire standard; products supporting SpaceWire including onboard equipment, instruments and related onboard software; methods and equipment to aid the test and verification of SpaceWire components, units and systems; and SpaceWire networks, their architecture, configuration, and discovery, as well as "plug and play" concepts, other higher level protocols and related hardware and software design issues. A technical seminar on SpaceWire at the conference was presented by several world leading experts on SpaceWire providing hints and tips on using SpaceWire based on many years' experience.

The community of engineers working on SpaceWire meet regularly at the SpaceWire Working Group meetings to help with the further development of SpaceWire and related standards and technologies. This group includes engineers from many parts of the World with substantial contributions from Europe, Japan, Russia and the USA. The SpaceWire Conference complements these Working Group meetings with more formal presentations from a wider range of contributors.

There is growing interest in the SpaceFibre which aims to provide multi-gigabit/s network technology for future space flight application like high-resolution multi-spectral imaging and synthetic aperture radar. A second seminar introduced and demonstrated SpaceFibre technology which can operate over both electrical and fibre-optic media. A growing number of papers in the conference addressed SpaceFibre.

The conference committee would like to acknowledge the support and hard work of the many individuals who made International SpaceWire Conference 2013 a reality. First, we thank the authors and the keynote speakers for their high-quality contributions. We express our gratitude to the Technical Committee for their assistance in the review process. We thank all people supporting us at Aeroflex, the Space Technology Centre at the University of Dundee and the European Space Agency.

The Conference Chairpersons,

Martin Suess, *European Space Agency, The Netherland*
Steve Parkes, *Space Technology Centre, University of Dundee, UK*
Sand Habinc, *Aeroflex Gaisler, Sweden*
Teresa Farris, *Aeroflex, USA*

Technical Committee

Alan Senior - SEA, UK

Yuriy Sheynin - St. Petersburg State University of Aerospace Instrumentation, Russia

Tatiana Solokhina - ELVEES, Russia

Martin Suess - ESA, The Netherlands

Tadayuki Takahashi - JAXA

Antonis Tavoularis - Teletel

Raffaele Vitulli - ESA, The Netherlands

Takahiro Yamada - JAXA/ISAS, Japan

Takayuki Yuasa – JAXA, Japan

Programme Overview

Monday 10 June

15:30 – 19:00 Registration

16:00 – 18:00 Tutorials of SpaceWire and SpaceFibre

Tuesday 11 June

09:00 – 10:00 Conference Opening / Keynote Presentations (60 min)

10:00 – 10:50 Standardisation 1 (50 min)

11:10 – 12:25 Standardisation 2 (75 min)

13:45 – 15:25 Test & Verification 1 (100 min)

15:45 – 17:00 Onboard Equipment & Software (75 min)

Wednesday 12 June

09:00 – 10:55 Components 1 (115 min)

11:15 – 12:05 Components 2 (50 min)

12:05 – 12:35 Onboard Equipment & Software (30min)

13:55 – 15:10 Networks & Protocols (75 min)

15:10 – 16:40 Poster Session (90 min)

Thursday 13 June

09:00 – 09:45	Standardisation (45 min)
09:45 – 10:45	Test & Verification (60 min)
11:05 – 12:35	Missions & Applications (90 min)
13:55 – 15:10	Components (75 min)
15:30 – 17:10	Networks & Protocols (100 min)

Programme is subject to change

Tuesday 11 June

Standardisation 1 (Long)

SpaceFibre: Multiple Gbit/s Network Technology with QoS, FDIR and SpaceWire Packet Transfer Capabilities

SpaceWire Standardisation, Long Paper

Steve Parkes, Chris McClements,
Space Technology Centre, University of Dundee,
Dundee, DD1 4EE, UK
sparkes@computing.dundee.ac.uk

Albert Ferrer, Alberto Gonzalez,
STAR-Dundee
STAR House, 166 Nethergate, Dundee, DD1 4EE, UK

Abstract— **SpaceFibre is a very high-speed serial link designed specifically for use onboard spacecraft. It carries SpaceWire packets over virtual channels and provides a broadcast capability similar to SpaceWire time-codes but offering much more capability. SpaceFibre operates at 10 times the data-rate of SpaceWire, can run over fibre optic or electrical media, provides galvanic isolation, includes coherence Quality of Service (QoS) and Fault Detection Isolation and Recovery (FDIR) support, and provides low-latency signalling. SpaceFibre can run over distances of 5m with copper cable and 100 m or more with fibre optic cable.**

SpaceFibre is compatible with the packet level of the SpaceWire standard (ECSS-E-ST-50-12) and is therefore able to run the SpaceWire protocols defined in ECSS-E-ST-50-51C, 52C and 53C. This means that applications developed for SpaceWire can be readily transferred to SpaceFibre.

The aim of SpaceFibre is to provide point-to-point and networked interconnections for very high data-rate instruments, mass-memory units, processors and other equipment, on board a spacecraft.

This paper introduces SpaceFibre, describes the SpaceFibre QoS, FDIR and network level operation of SpaceFibre.

Index Terms—*SpaceWire, SpaceFibre, networks, spacecraft onboard processing*

I. INTRODUCTION

SpaceFibre [1] [2] [3] [4] is a very high-speed serial data-link being developed by the University of Dundee for ESA which is intended for use in data-handling networks for high data-rate payloads. SpaceFibre is able to operate over fibre-optic and electrical cable and support data rates of 2 Gbit/s in the near future and up to 5 Gbit/s long-term. It aims to complement the capabilities of the widely used SpaceWire onboard networking standard [5]: improving the data rate by a factor of 10, reducing the cable mass by a factor of four and providing galvanic isolation. Multi-laning improves the data-rate further to well over 20 Gbits/s.

SpaceFibre provides a coherent quality of service mechanism able to support best effort, bandwidth reserved, scheduled and priority based qualities of service. It substantially improves the fault detection, isolation and recovery (FDIR) capability compared to SpaceWire.

SpaceFibre aims to support high data-rate payloads, for example synthetic aperture radar and hyper-spectral optical instruments. It provides robust, long distance communications for launcher applications and supports avionics applications with deterministic delivery constraints through the use of virtual channels. SpaceFibre enables a common onboard network technology to be used across many different mission applications resulting in cost reduction and design reusability. SpaceFibre uses a packet format which is the same as SpaceWire enabling simple connection between existing SpaceWire equipment and high-speed SpaceFibre links and networks.

The SpaceFibre interface is designed to be implemented efficiently, requiring substantially fewer logic gates than a RapidIO interface. It is currently being prototyped in a range of onboard processing, mass memories and other spacecraft applications. Interoperability tests between independent Japanese and European implementations were carried out successfully in December 2012 and April 2013.

II. SPACEFIBRE PROTOCOL STACK

The SpaceFibre protocol stack is illustrated in Figure 1.

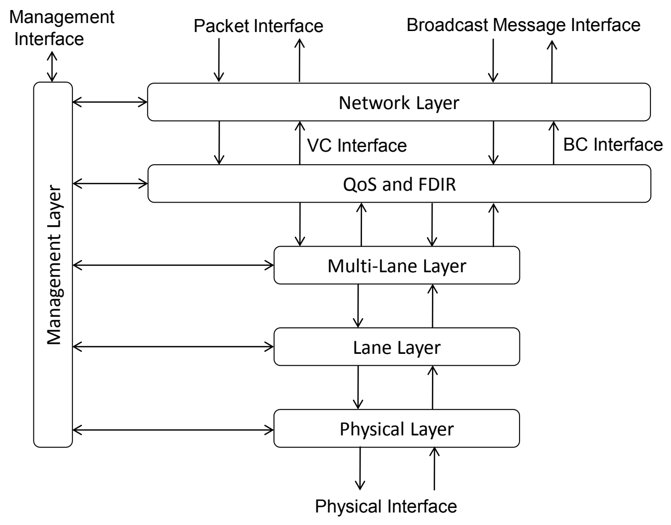


Figure 1 SpaceFibre Protocol Stack

The network layer protocol is responsible for the transfer of application information over a SpaceFibre network. It provides two services: Packet Transfer Service and Broadcast Message Service. The Packet Transfer Service transfers SpaceFibre packets over the SpaceFibre network, using the same packet format and routing concepts as SpaceWire uses. SpaceFibre supports both path and logical addressing. The broadcast message service is responsible for broadcasting short messages (8 bytes) to all nodes on the network. These messages can carry time and synchronisation signals and be used to signal the occurrence of various events on the network.

The management layer is responsible for configuring, controlling and monitoring the status of all the layers in the SpaceFibre protocol stack. For example it can configure the QoS settings of the virtual channels in the QoS and FDIR layer.

The QoS and FDIR layer is responsible for providing quality of service and managing the flow of information over a SpaceFibre link. It frames the information to be sent over the link to support QoS and scrambles the packet data to reduce electromagnetic emissions. The QoS and FDIR layer also provides a retry capability, detecting any frames or control codes that go missing or arrive containing errors and resending them. With this inbuilt retry mechanism SpaceFibre is very resilient to transient errors.

The Multi-Lane layer is responsible for operating several SpaceFibre lanes in parallel to provide higher data throughput. In the event of a lane failing the Multi-Lane layer provides support for graceful degradation, automatically spreading the traffic over the remaining working links.

The Lane layer is responsible for lane initialisation and error detection. In the event of an error the lane is automatically re-initialised. The Lane layer encodes data into symbols for transmission using 8B/10B encoding and decodes these symbols in the receiver. 8B/10B codes are DC balanced supporting AC coupling of SpaceFibre interfaces.

The Physical layer is responsible for serialising the 8B/10B symbols and for sending them over the physical medium. In the receiver the Physical layer recovers the clock and data from the serial bit stream, determines the symbol boundaries and recovers the 8B/10B symbols. Both electrical cables and fibre-optic cables are supported by SpaceFibre.

III. SpaceFibre Quality of Service

A SpaceFibre interface includes a number of virtual channels. Each provides a FIFO type interface like a SpaceWire link. When data from a SpaceWire packet is placed in a SpaceFibre virtual channel it is transferred over the SpaceFibre link and placed in the same numbered virtual channel at the other end of the link. Data from the several virtual channels are interleaved over the physical SpaceFibre connection. To support the interleaving, data is sent in short frames of up to 256 SpaceWire N-chars each. A virtual channel can be assigned a quality of service which determines the precedence with which that virtual channel will compete with other virtual channels for sending data over the SpaceFibre link. Priority, bandwidth reservation, and scheduled qualities of service can be supported all operating together using a simple precedence mechanism.

In this section the SpaceFibre quality of service mechanism is described.

A. Frames and Virtual Channels

To provide quality of service, it is necessary to be able to interleave different data flows over a data link or network. If a large packet is being sent with low priority and a higher priority one requests to be sent, it must be possible to suspend sending the low priority one and start sending the higher priority packet. To facilitate this SpaceWire packets are chopped up into smaller data units called frames. When the high priority packet requests to be sent, the current frame of the low priority packet is allowed to complete transmission, and then the frames of the high priority packet are sent. When all the frames of the high priority packet have been sent, the remaining frames of the low priority packet can be sent.

Each frame has to be identified as belonging to a particular data flow so that the stream of packets can be reconstructed at the other end of the link. Low priority packets belong to one data stream and high priority packets belong to another data stream.

Each independent data stream allowed to flow over a data link is referred to as a virtual channel (VC). Virtual channels are unidirectional and have a QoS attribute, e.g. priority. At each end of a virtual channel is a virtual channel buffer (VCB), which buffers the data from and to the application. An output VCB takes data from the application and buffers it prior to sending it across the data link. An input VCB receives data from the data link and buffers it prior to passing it to the receiving application.

There can be several output virtual channels connected to a single data link, which compete for sending information over the link. A medium access controller determines which output virtual channel is allowed to send the next data frame. When an output VCB has a frame of data ready to send and the corresponding input VCB at the other end of the link has room for a full data frame, the output VCB requests the medium access controller to send a frame. The medium access controller arbitrates between all the output VCBs requesting to send a frame. It uses the QoS attribute of each of the requesting VCBs to determine which one will be allowed to send the next data frame.

Priority is one example of a QoS attribute. Other types of QoS are considered in the subsequent sections.

B. Precedence

For the medium access controller to be able to compare QoS attributes from different output VCBs, it is essential that they are all using a common measure that can be compared. The name given to this measure is precedence. The competing output VCB with the highest precedence will be allowed to send the next frame.

C. Bandwidth Reservation

When connecting an instrument via a network to a mass memory, what the systems engineer needs to know is “how much bandwidth do I have to transfer data from the instrument to the mass memory?” Once the network bandwidth allocated to a particular instrument has been specified, it should not be possible for another instrument to impose on the bandwidth allocated to that instrument. A priority mechanism is not suitable for this application. If an instrument with high priority has data to send it will hog the network until all its data has been sent. What is needed is a mechanism that allows bandwidth to be reserved for a particular instrument.

Bandwidth reservation calculates the bandwidth used by a particular virtual channel, and compares this to the bandwidth reserved for that virtual channel to calculate the precedence for that virtual channel. If the virtual channel has not used much reserved bandwidth recently, it will have a high precedence. When a data frame is sent by this virtual channel, its precedence will drop. Its precedence will increase again over a period of time. If a virtual channel has used more than its reserved bandwidth recently, it will have a low precedence.

A virtual channel specifies a portion of overall Link Bandwidth that it wishes to reserve and expects to use, i.e. its Expected Bandwidth.

When a frame of data is send by any virtual channel, each virtual channel computes the amount of bandwidth that it would have been permitted to send in the time interval that the last frame was sent. This is known as the Bandwidth Allocation. Bandwidth Allowance is calculated as follows:

$$BandwidthAllowance = Expected \times LastFrameBandwidth$$

Where Expected or Expected Bandwidth Percentage is the portion of overall link bandwidth that a virtual channel wishes to use, and Last Frame Bandwidth is the amount of data sent in the last data frame.

Each virtual channel can use this to determine its Bandwidth Credit, which is effectively the amount of data it can send and still remain within its Expected Bandwidth. Bandwidth Credit is the Bandwidth Allowance less the Bandwidth Used accumulated over time.

Bandwidth Credit is calculated for each virtual channel as follows:

$$BandwidthCredit = \sum_{Frames} \frac{BandwidthAllowance - UsedBandwidth}{Expected}$$

Where Used Bandwidth is the amount of data sent by a particular virtual channel in the last data frame, which is zero except for all virtual channels except for the one that sent the last frame.

The Bandwidth Credit is updated every time a data frame for any virtual channel has been sent. A Bandwidth Credit value close to zero indicates nominal use of bandwidth by the virtual channel. A negative value indicates that the virtual channel is using more than its expected amount of link bandwidth. A positive value indicates that the virtual channel is using less than its expected amount of link bandwidth.

To simplify the hardware required to calculate the Bandwidth Credit it is allowed to saturate at plus or minus a Bandwidth Credit Limit, i.e. if the Bandwidth Credit reaches a Bandwidth Credit Limit it is set to the value of the Bandwidth Credit Limit.

When the Bandwidth Credit for a virtual channel reaches the negative Bandwidth Credit Limit it indicates that the virtual channel is using more bandwidth than expected. This may be recorded in a status register and used to indicate a possible error condition. A network management application is able to use this information to check correct utilisation of link bandwidth by its various virtual channels.

For a virtual channel supporting bandwidth reserved QoS, the value of the bandwidth counter provides the precedence value for that virtual channel.

The operation of a bandwidth credit counter is illustrated in Figure 2.

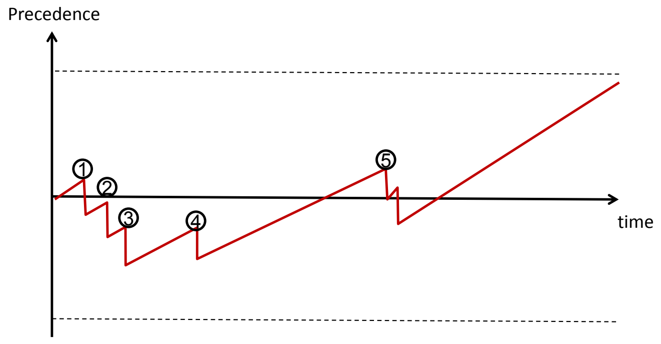


Figure 2 Bandwidth Credit Counter

The bandwidth credit for a particular VC increments gradually. At point (1) a frame is sent from by this VC, resulting in a sudden drop in credit. The size of the drop is amount of data sent in the frame divided by the percentage bandwidth reserved for the VC. This means that the smaller the percentage bandwidth the larger the drop, and hence the longer it takes to regain bandwidth credit.

After the drop at point (1) the bandwidth credit gradually increments until point (2) when another frame is sent by the VC. Further frames are sent at points (3), (4), (5) etc. If the frames sent are full frames then the drop in bandwidth credit every time a frame is sent, will be the same size.

The bandwidth credit counter for another VC is illustrated in Figure 3. This VC has about half the bandwidth of the VC in Figure 2 allocated to it. This means that the drops in bandwidth credit when frames are sent by this VC are about twice the size, as can be seen Figure 3 at points (1), (2) and (3).

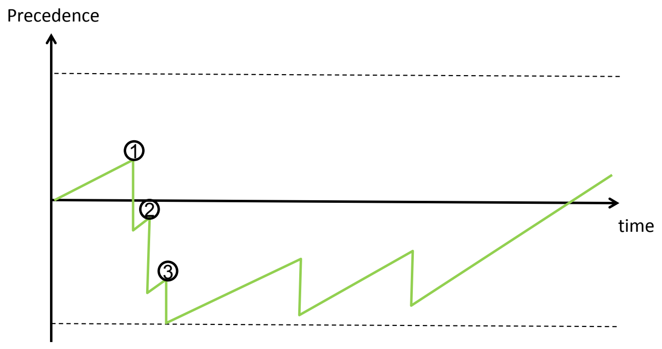


Figure 3 Bandwidth Credit Counter with Smaller Reserved Bandwidth

The bandwidth credit counter of another VC is shown in Figure 4. In this case the bandwidth credit slowly increments and although some frames are sent at points (1), (2) and (3), the bandwidth credit eventually saturates, reaching its maximum permitted value at point (4). Although more bandwidth should be accumulated after point (4) this is effectively ignored since the maximum possible bandwidth credit has been reached. At point (5) a frame is sent once more, resulting in a drop from the maximum bandwidth credit value.

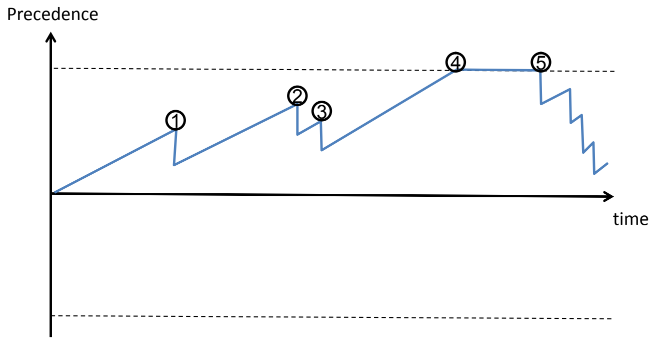


Figure 4 Bandwidth Credit Counter Reaching Saturation

All three VCs are shown together in Figure 5. When a VC has a data frame ready to send and room for a full data frame at the other end of the link, it competes with any other VCs in a similar state, the one with the highest bandwidth credit being allowed to send the next data frame. At points (1), (2) and (3) the red VC has data to send and sends frames. At points (4), (5) and (6) the green VC has data to send and sends a data frame. At point (7) both the blue and the red VCs have data to send. The blue VC wins since it has the highest bandwidth credit count. After this the red VC is allowed to send a further data frame at point (8).

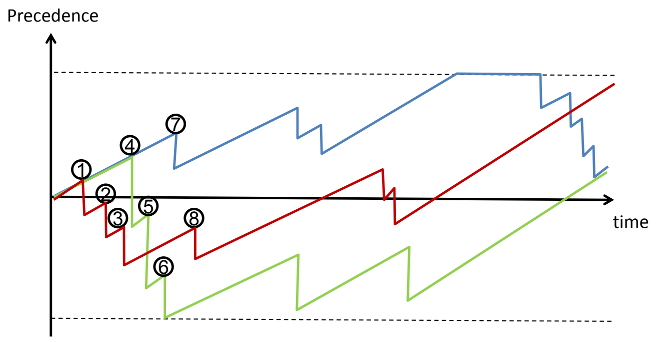


Figure 5 Bandwidth Credit of Competing VCs

If the bandwidth credit counter reaches the minimum possible bandwidth credit value, it indicates that it is using more bandwidth than expected and a possible error may be flagged. This condition may be used to stop the VC sending any more data until it recovers some bandwidth credit, to help with “babbling idiot” protection.

Similarly if the bandwidth credit counter stays at the maximum possible bandwidth credit value for a relatively long period of time, the VC is using less bandwidth than expected and this condition can be flagged to indicate a possible error.

The bandwidth credit value is the precedence used by the medium access controller to determine which VC is permitted to send the next data frame.

D. Priority

The second type of QoS provided by VCs is priority. Each VC is assigned a priority value and the VC with the highest priority (lowest priority number) is allowed to send the next data frame as soon as it is ready. Figure 6 shows three priority levels. SpaceFibre has 16 priority levels.

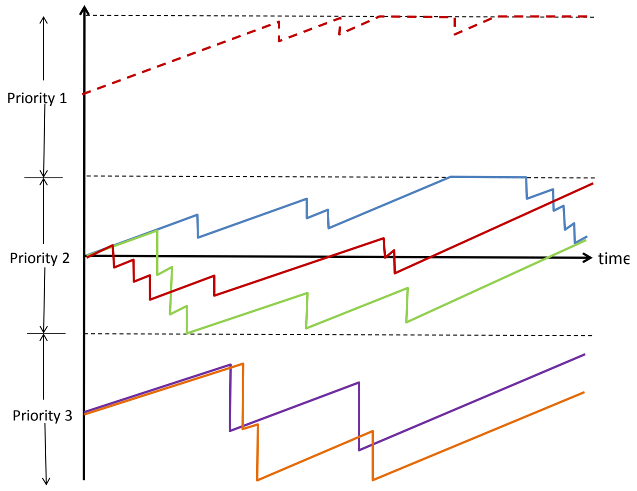


Figure 6 Multi-Layered Precedence Priority QoS

Within any level there can be any number of VCs which compete amongst themselves based on their bandwidth credit. A higher priority VC will always have precedence over a lower priority VC unless its Bandwidth Credit has reached the minimum credit limit in which case it is no longer allowed to send any more data frames. This prevents a high priority VC from consuming all the link bandwidth if it fails and starts babbling. More than one VC can be set to the same priority level in which case those VC's will compete for medium access using bandwidth reservation.

6.5 Scheduled

To provide fully deterministic data delivery it is necessary for the QoS mechanism to ensure that data from specific virtual channels can be sent (and delivered) at particular times. This can be done by chopping time into a series of time-slots, during which a particular VC is permitted to send data frames. This is illustrated in Figure 7.

Time-slot	1	2	3	4	5	6	7	8
VC 1	🟩							
VC 2		🟩				🟩		
VC 3				🟩				
VC 4								
VC 5							🟩	
VC 6			🟩					
VC 7								
VC 8			🟩					

Figure 7 Scheduled Quality of Service

Each VC is allocated one or more time-slots in which it is permitted to send data frames. VC1 is scheduled to send in time-slot 1 and VC2 is scheduled to send in time-slots 2 and 3. The time-slot duration is a system level parameter, typically 100 µs, and there are 256 time-slots.

During a time-slot, if the VC is scheduled to send in that time-slot, it will compete with other VCs also scheduled to send in that time-slot based on precedence (priority and bandwidth credit). A fully deterministic system would have one VC allowed to send in a time-slot.

The schedule is always operating. If a user does not want to use scheduling the schedule table is simply filled completely, allowing any VC to send in any time-slot, competing with precedence.

Scheduling can waste bandwidth if only one VC is allowed to send in a time-slot and that VC is not ready. To avoid this situation, the critical VC can be allocated a time-slot and given high priority. Another VC can be allocated the same time-slot with lower priority. In this way when that time-slot arrives the high priority VC will be allowed to send its data, but if it is not ready the VC with lower priority can send some data. This configuration is illustrated in Figure 7 time-slot 3 and VCs 6 and 8.

Time-slots can be defined using broadcast messages to send start of time-slot signals or to send time information and having a local time counter which determines the start and end of each time-slot. The SpaceFibre broadcast message mechanism support both synchronisation and time distribution.

The SpaceFibre QoS mechanism is simple and efficient to implement and it provides bandwidth reservation, priority and scheduling integrated together, not as separate options. Furthermore SpaceFibre QoS provides a means for detecting "babbling idiots" and for detecting nodes that have ceased sending data when they are expected to be sending information.

IV. SpaceFibre Fault Detection, Isolation and Recovery

SpaceFibre provides automatic fault detection, isolation and recovery. When a fault occurs on a SpaceFibre link, it is detected and the erroneous or missing information resent. SpaceFibre recovers from intermittent faults very rapidly, detecting faults, recovering and resending data faster than SpaceWire disconnects and reconnects a link. The retry mechanism does not depend on time-outs, naturally adapting to different cable delays.

Fault detection is provided by checking each 8B/10B symbol for disparity errors and invalid 8B/10B codes. SpaceFibre has selected the 8B/10B K-codes it uses to have enhanced Hamming distance from data-codes. This means that a single bit error occurring in a data-code cannot result in a valid K-code used by SpaceFibre. In addition each data frame, broadcast frame, FCT, ACK and NACK are protected by a CRC.

Fault isolation is provided at various levels in SpaceFibre. AC coupling is used in the physical layer to prevent damage from faults that cause DC voltages exceeding the maximum permitted to appear on the transmitter outputs or receiver inputs. This feature also enables galvanic isolation to be implemented readily. At the Quality level SpaceFibre provides time containment, containing errors in the data frame in which they occur, and bandwidth containment, containing errors to the virtual channel in which they occur; an error in one VC does not affect data flowing in another VC. Babbling idiots are contained using the QoS mechanism described above.

Fault recovery is provided at the link level using a retry mechanism that resends data frames, broadcast frames and FCTs. The retry is very fast, uses a minimum amount of buffer memory, and adapts automatically to different link lengths. In addition to the retry mechanism the multi-lane functionality includes graceful degradation on lane failure. If a lane fails permanently, so that a retry or re-initialisation does not recover lane operation, a multi-lane system will continue using the remaining lanes available. This reduces the bandwidth available but does not stop the link operating. For critical operations an extra lane can be included and the graceful degradation will then provide automatic replacement of a faulty lane. The bit error rate (BER) of a lane is monitored and a lane reported as faulty if the (BER) is above a level which results in the effective link bandwidth being unusable. This feature allows lanes that can re-initialise successfully but which will not run for very long before having to re-initialise again, to be detected, isolated and replace by a fully functional lane.

V. SpaceFibre Networks

A SpaceFibre network uses similar packet formats, packet addressing and routing concepts to SpaceWire. The main difference is that SpaceFibre includes virtual channels.

A SpaceFibre router is illustrated in Figure 8.

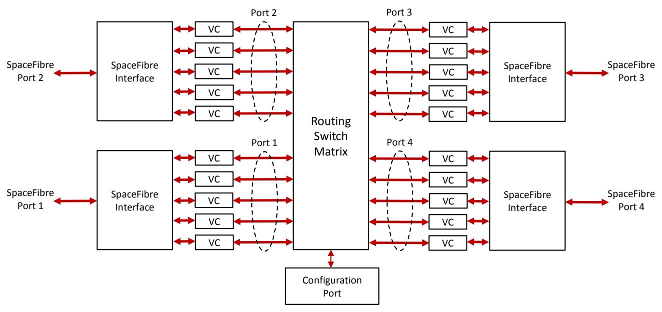


Figure 8 SpaceFibre Router

The SpaceFibre router comprises a number of SpaceFibre interfaces and a routing switch matrix. Each SpaceFibre interface has several virtual channels. The VC number for each virtual channel can be configured, except for VC0 which is a virtual channel used for configuration, control and monitoring of the SpaceFibre network. When a packet arrives on a SpaceFibre interface it is placed in the appropriate virtual channel, i.e. the one with the same VC number as it was transmitted on. The leading data character of the packet determines which port of the routing switch the packet is to be forwarded through using either path or logical addressing. The port that it is to be switched to must have a VC configured with the same number as the VC that the packet arrived on. The packet is then passed through the routing switch matrix and placed frame by frame in the VC of the output port. The packet is then transferred across the SpaceFibre link, competing with other VCs in that port for access to the link medium according to their precedence.

If a packet arrives and the output port that the packet is to be switched to does not have a VC with the same number as that on which it arrived, the packet is spilt and an error recorded.

Virtual channels can be used to construct virtual networks where a single VC number is used for connecting to all or several of the nodes attached to the network. This is illustrated in Figure 9 where VC6 (blue) is used to connect all the nodes on the network. Using VC6 the Control Processor can send commands to Instrument 1 or 2 or the Mass Memory unit, setting their operating mode or reading housekeeping information, etc. This virtual network acts like a SpaceWire network.

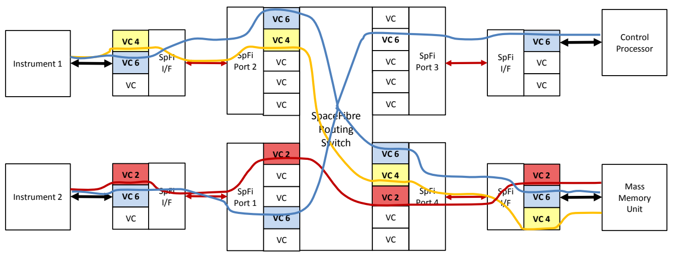


Figure 9 A Simple SpaceFibre Network

Virtual channels can also be used to construct virtual point to point links from one node to another. VC2 and VC4, in Figure 9, are providing virtual point to point links. VC2 provides a virtual point to point link between Instrument 2 and the Mass Memory Unit and VC4 between Instrument 1 and the Mass Memory. These virtual channels can be each allocated the bandwidth they need to send their data to the Mass Memory Unit. Once this bandwidth is allocated other virtual channels or virtual networks will not interfere with their operation.

Figure 10 shows a more realistic onboard network using SpaceFibre which includes a SpaceWire to SpaceFibre Bridge. Two high data-rate instruments (Instruments 1 and 2) have SpaceFibre connections. Four less demanding instruments have SpaceWire connections to the SpaceWire to SpaceFibre Bridge. Each instrument has a virtual point to point connection to the Mass Memory Unit and there is a virtual point to point connection between the Mass Memory and the Downlink Telemetry Unit. The Control Processor has a virtual network for configuring and controlling all devices on the network.

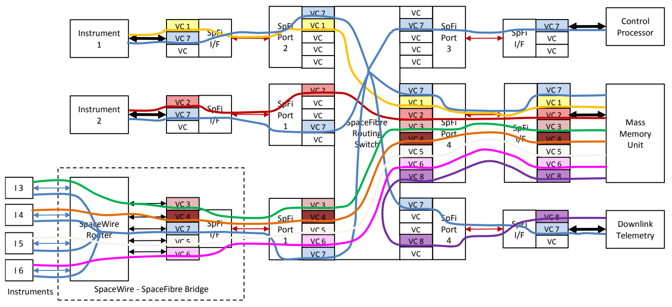


Figure 10 Realistic SpaceFibre Network

Figure 10 is solving a complex communication task with many separate, isolated virtual channels providing point to point links, and a virtual network being used to control the entire system. Figure 11 shows this same network with the virtual channels removed, revealing the simplicity of implementation of a complex communication task when using SpaceFibre.

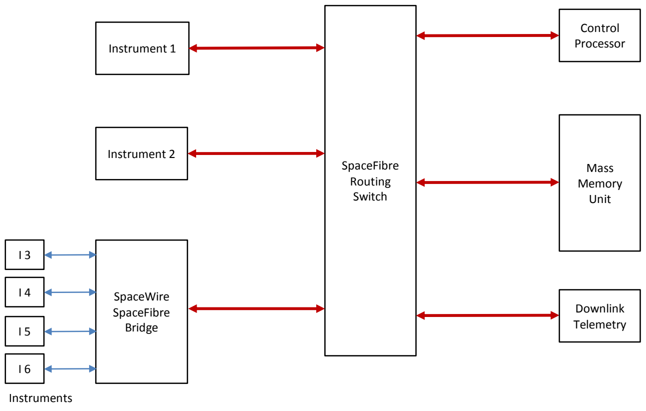


Figure 11 Simple System Architecture with SpaceFibre

VI. SpaceFibre Implementations

The SpaceFibre specification has been written by the University of Dundee for ESA, and has been widely reviewed by the international spacecraft engineering community. It has also been simulated and implemented in several forms. While work remains to be done on the specification the existing draft specification is close to maturity. In this section the current state of SpaceFibre is explored.

A SpaceFibre interface has been designed by University of Dundee and STAR-Dundee for ESA. This VHDL IP core has been used at all stages of the draft specification to validate and prove the concepts being explored. As a consequence the VHDL IP core has gone through as many iterations as the SpaceFibre specification. At present the VHDL IP core implements all layers of the SpaceFibre specification with the exception of the Multi-Lane layer.

To support the testing of SpaceFibre a suitable test platform was required, so STAR-Dundee developed the STAR Fire unit, which has two SpaceFibre interfaces and includes a link diagnostic capability for analysing traffic on a SpaceFibre link. Two STAR Fire units are being used in Figure 12 to help with the testing of radiation tolerant Fibre Optic transceivers for SpaceFibre operating over 100 m of Fibre Optic cable.

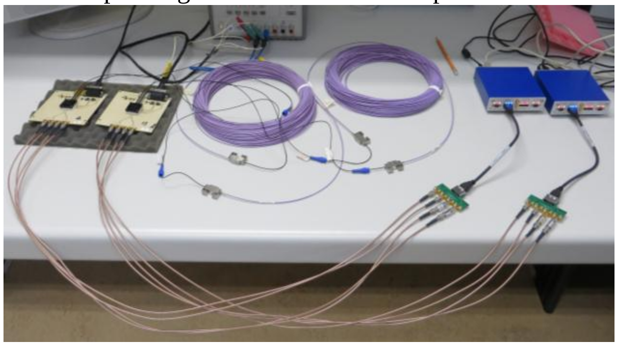


Figure 12 STAR Fire Testing 100m Fibre Optic Cable

A radiation tolerant SpaceFibre interface device (VHiSSI) is being developed by University of Dundee and several partners within a European Union (EU) Framework 7 project [6].

NEC and Melco are both developing SpaceFibre interface devices to the specification produced by the University of Dundee. This work is providing valuable feedback on the specification and implementation of SpaceFibre. Interoperability testing in December 2013 and April 2013 has been successful with various levels of the SpaceFibre protocol stack being implemented and tested.

Research carried out during the SpaceWire-RT EU Framework 7 project resulted in the Quality layer for SpaceFibre being developed by University of Dundee. Within this same project St. Petersburg University of Aerospace Instrumentation (SUAI) modelled and simulated the various layers of SpaceFibre and ELVEES assessed the feasibility of ASIC implementation using a custom designed SerDes.

Several ESA projects are using the Dundee SpaceFibre IP core under a Beta evaluation programme including:

- High Performance COTS Based Computer, Astrium and CGS.
- Leon with Fast Fourier Transform Co-processor, SSBV.
- FPGA Based Generic Module and Dynamic Reconfigurator, Bielefeld University.
- Next Generation Mass Memory, Astrium, IDA and University of Dundee.
- 1 x High Processing Power DSP, Astrium and STAR-Dundee.

Work on the formal European Cooperation for Space Standardization (ECSS) standard for SpaceFibre is schedule to start in early 2014, once the technical specification is complete.

VII. Conclusions

SpaceFibre is a multi-gigabit/s networking technology designed specifically for spaceflight applications. It incorporates a comprehensive quality of service capability providing integrated bandwidth reservation, priority and scheduling. Efficient, effective and rapid fault detection, isolation and recovery mechanisms are included in the SpaceFibre interface, enabling rapid detection and recovery from link level errors.

SpaceFibre is designed to support very high data-rate missions like multi-spectral imagers and synthetic aperture radar. It reduces development time and costs, because of its integrated QoS and FDIR capabilities and because it simplifies previously complex onboard data-handling architectures. SpaceFibre is designed to use the same packet format as SpaceWire enabling straightforward upgrading of spacecraft networks to include the improved QoS, FDIR and bandwidth of SpaceFibre while being able to operate with existing SpaceWire equipment. SpaceWire units can be readily integrated with SpaceFibre using a SpaceWire to SpaceFibre Bridge.

Acknowledgment

The research leading to these results has received funding the European Space Agency under ESA contract numbers 4000102641 and from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 263148 and 284389. We would also like to thank Martin Suess the ESA project manager for the SpaceFibre related activities for his help, advice and guidance.

References

[1] S. Parkes, A. Ferrer, A. Gonzalez, & C. McClements, “SpaceFibre Standard Draft E1”, University of Dundee, 28th September 2012.

[2] S. Parkes, A. Ferrer, A. Gonzalez, & C. McClements, “SpaceFibre: Multiple Gbits/s Network Technology with QoS, FDIR and SpaceWire Packet Transfer Capabilities”, International SpaceWire Conference, Gothenburg, June 2013.

[3] S. Parkes, “Never Mind the Quality, Feel the Bandwidth: Quality of Service Drivers for Future Onboard Communication Networks”, paper no. IAC-10.B2.6.6, 61st International Astronautical Congress, Prague 2010.

[4] S. Parkes, C. McClements and M. Suess, “SpaceFibre”, International SpaceWire Conference, St Petersburg, Russia, 2010, ISBN 978-0-9557196-2-2, pp 41-45.

[5] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire, Links, Nodes, Routers and Networks”, Issue 1, European Cooperation for Space Data Standardization, July 2008.

[6] S. Parkes, A. Ferrer, A. Gonzalez, C. McClements, R. Ginosar, T Liran, G Sokolov, G Burdo, N Blatt, P Rastetter, M Krstic, A Crescenzio, “A Radiation Tolerant SpaceFibre Interface Device”, International SpaceWire Conference, Gothenburg, 2013.

SpaceFibre Quality of Service and Network Routing

SpaceWire Standardisation, Long Paper

Clifford Kimmery and Stephen Belvin
Honeywell International
Space Electronic Systems
Clearwater, FL, USA
clifford.kimmery@honeywell.com, stephen.belvin@honeywell.com

***Abstract*— The available SpaceFibre literature views each SpaceFibre virtual channel as logically equivalent to a SpaceWire physical link. This view can be interpreted as allowing SpaceWire packet traversal of the SpaceFibre network through any appropriate virtual channel. Because SpaceWire packets do not have associated Quality of Service (QoS) attributes, the SpaceFibre Standard Draft [1] associates QoS attributes with virtual channels. As a result, SpaceFibre routers must make routing decisions using a combination of SpaceWire packet header and virtual channel QoS attributes to provide a consistent QoS from source to destination. Parkes and Suess [2] introduce the concept of a SpaceFibre virtual network composed of virtual channels with identical Virtual Channel Identifier (VCID) and QoS attributes such that packet routing can be performed using the SpaceWire packet header and the VCID.**

While the virtual network concept is beneficial for providing routes with the same QoS attributes through the SpaceFibre network, the mechanisms used should have the flexibility to address the majority of application use cases. In particular, the solution must address the likelihood that some SpaceFibre endpoint implementations will support a small number of virtual channels (for example, one or two). Mapping the resulting collection of virtual networks onto the SpaceFibre network is likely to be a significant problem for larger networks.

We consider the limitations imposed on SpaceFibre network routing by the coupling of VCIDs with virtual networks. We also review possible methods for addressing the limitations identified.

***Index Terms*—SpaceFibre, SpaceWire, network, routing, quality of service, virtual channel, virtual network.**

I. Introduction

The introduction of Virtual Channels and Quality of Service (QoS) in SpaceFibre is a significant advancement for the SpaceWire community. The new capabilities also increase the complexity of networks based on SpaceFibre protocols. The concept of establishing virtual networks to provide customized performance capabilities for each network traffic type introduces an abstraction that makes SpaceFibre network routing easier to contemplate and accomplish. By associating a QoS type (we define a QoS type as a specific set of QoS attributes) with each virtual network (VN), network users can view the physical SpaceFibre network as a collection of virtual SpaceWire networks. Each virtual SpaceWire network offers specific network performance characteristics established by the associated QoS type.

II. Overview of SpaceFibre Virtual Channels and Quality of Service

Virtual channels are fundamental to the SpaceFibre architecture and form the basis for any concept of SpaceFibre networks. A virtual channel is a unidirectional logical link through a physical SpaceFibre link between two nodes. Multiple virtual channels can share a single SpaceFibre physical link. Each virtual channel has independent flow control and behaves like a traditional SpaceWire physical link. Because of the unidirectional nature of SpaceFibre virtual channels, the number of virtual channels operating in one direction can be different than the number operating in the other direction. This capability offers significant flexibility to SpaceFibre network designers.

Each SpaceFibre Encoder/Decoder (CODEC) implementation must provide no less than one virtual channel (up to a maximum of 256). A SpaceFibre CODEC must include a Virtual Channel Buffer (VCB) for each Virtual Channel (VC) supported. The SpaceFibre link provides flow control and other link management support by associating a Virtual Channel Identifier (VCID) with each virtual channel. SpaceFibre frames transmitted by the link encoder are associated with the virtual channel by including the VCID in the frame header. A SpaceFibre data frame is shown in Fig. 1 with the header consisting of COMMA, Start of Data Frame (SDF) and VC characters. The frame trailer consists of an End of Data Frame (EDF), Frame Sequence (FR_SEQ#) and CRC.

0 7	8 15	16 23	24 31
COMMA	SDF	VC	RESERVED
DATA 1 LS	DATA 1	DATA 1	DATA 1 MS
DATA 2 LS	DATA 2	DATA 2	DATA 2 MS
●●●	●●●	●●●	●●●
DATA N LS	DATA N	DATA N	DATA N MS
EDF	FR_SEQ#	CRC_LS	CRC_MS

Fig. 1. SpaceFibre Data Frame Format.

One of the key benefits of SpaceFibre is support for Quality of Service behaviors. Each QoS behavior defines a mechanism for ensuring that a defined set of network performance characteristics are met. Because SpaceWire packets do not have associated Quality of Service attributes, the SpaceFibre Standard Draft associates QoS attributes with virtual channels. SpaceWire packets traversing a SpaceFibre link are prioritized based on the QoS attributes of the virtual channel. The QoS behavior is established for each virtual channel at the SpaceFibre link transmitter while the link receiver treats all virtual channels equally.

The SpaceFibre Standard Draft defines four QoS behaviors: best effort, priority, bandwidth reservation and scheduled. The Medium Access Controller (MAC) in the SpaceFibre CODEC uses the precedence of each virtual channel to prioritize data transmission. The best effort QoS behavior is assigned the lowest precedence and is dependent on the availability of unallocated link bandwidth to allow packets to traverse the network. The priority QoS behavior defines 16 levels of priority and the SpaceFibre Standard Draft assigns a specific precedence to each level. The best effort QoS is equivalent to lowest precedence level of the priority QoS.

The bandwidth reservation and scheduled QoS behaviors are more complex than the best effort and priority behaviors. In order to determine the precedence of a virtual channel with the bandwidth reserved QoS behavior, the MAC must consider the expected bandwidth utilization of the virtual channel, along with its recently utilized bandwidth and the available link bandwidth. The scheduled QoS behavior is based on fixed time periods or slots, where a virtual channel may be configured to transmit frames in one or more time slots. Priority may be combined with bandwidth reservation or scheduled QoS to provide more control over precedence for a virtual channel.

III. SpaceFibre Network Concepts

Much of the existing SpaceFibre literature envisions using SpaceFibre links in point-to-point applications. The focus on single-link applications is appropriate since the SpaceFibre Standard Draft does not address network-level aspects. While some work [2] [3] [4] has considered the application of SpaceFibre to general networks, many aspects remain unresolved.

Earlier work on QoS in SpaceFibre networks [4] has shown the difficulties derived from a lack of QoS attributes associated with SpaceWire packets. Two alternatives for associating QoS attributes with SpaceWire packets traversing SpaceFibre networks were discussed. The first, association of QoS with each virtual channel, is the basis of the method used by the SpaceFibre Standard Draft. The second, including QoS information in the SpaceFibre frame header, offers greater flexibility in network routing at a significant cost in implementation complexity.

The primary disadvantage of associating QoS with the virtual channel is that the number of available VCIDs establishes the upper bound on the number of independent QoS types that can be used in any SpaceFibre link. Recent SpaceFibre work [3] exacerbates this disadvantage by also using the VCID in the SpaceFibre network routing mechanism.

Including QoS attributes within the SpaceFibre frame header has the disadvantage that each SpaceWire packet traversing the virtual channel can dynamically change the QoS attributes of the virtual channel. For the simple QoS types (priority and best effort), this behavior does not have much effect, but for the complex QoS types (bandwidth reservation and scheduled) that are dependent on historical traffic patterns or out-of-band events, such perturbations would be difficult to address.

More recent work [2] [3] defines virtual networks as collections of virtual channels with identical QoS attributes (QoS type). All SpaceWire packet traffic utilizing the virtual network has the same precedence through the physical SpaceFibre network.

In its simplest form, this concept uses virtual SpaceWire networks with topology identical to the physical topology of the host SpaceFibre network. Each virtual network is associated with the QoS attributes needed by one class of SpaceWire traffic used in the system application. For simplicity, each virtual network is allocated a virtual channel in both directions of each SpaceFibre link. In cases where the packet traffic flows in a single direction through the virtual network, the reverse virtual channel is unused.

The concept can be extended by recognizing that the system application is unlikely to use all of the links of the SpaceFibre physical topology in each of the virtual SpaceWire networks. The unused virtual channels do not need to be allocated physical SpaceFibre link resources, freeing those resources for use by other virtual channels.

There are a number of factors that should be considered to make the virtual SpaceWire network design effort sufficiently flexible and convenient. Initial SpaceFibre network implementations are likely to utilize relatively simple topologies containing few links. These initial versions will be composed from newly developed devices designed for the specific applications. The SpaceFibre community should expect to face the issues that have arisen in the SpaceWire community as more capable devices become available and must be integrated into complex networks with older, less-capable devices. The SpaceFibre Standard Draft should be developed with the expectation that long-term use will result in similar experiences as technology and applications evolve.

IV. SpaceFibre Network Considerations

The SpaceFibre Standard Draft [1] does not address network-level topics relevant to SpaceWire packet routing, raising a number of concerns. The recent suggested methods [3] [4] for routing in SpaceFibre networks address some, but not all, of these concerns. We review each concern in this section.

A. Practical Implementation Constraints

When developing SpaceFibre networks composed of elements with differing levels of VC support, SpaceFibre network designers must assess the many possible mappings of QoS to virtual networks to reach an optimum configuration. In relatively complex cases, achieving a satisfactory result may be dependent on the VC capacity of some network elements.

As an example, a network designer can begin by identifying all of the QoS classes needed for the various data flows through the SpaceFibre network. The designer can then define a virtual network for each QoS class by assuming that every network endpoint and router is capable of supporting all of the virtual networks. Figure 2 shows an example on-board data processing system with five SpaceWire VNs identified. The Configuration Network and the Data Network are bi-directional, while the two Instrument Networks and the Uplink Network are unidirectional.

A complete definition of each virtual SpaceWire network requires a detailed inventory of the virtual channels available for each SpaceFibre link. As the number of virtual networks is increased, the ability to optimally provision each VN as desired becomes more difficult. Solutions to such SpaceFibre network optimization problems require the ability to assign virtual channels to an arbitrary virtual network map.

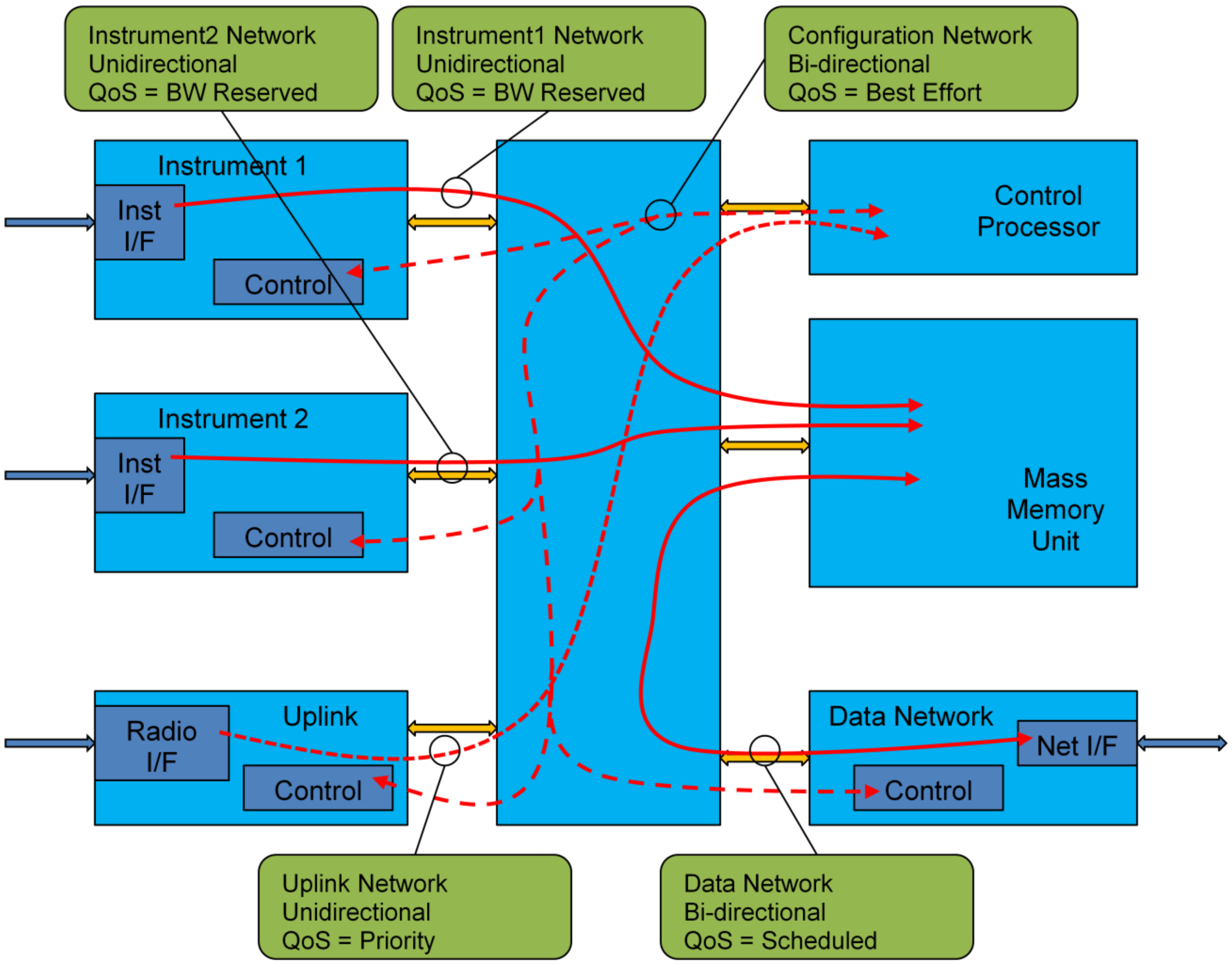


Fig. 2. Virtual Network Identification

B. Interoperability of Resource-Constrained Devices

Practical SpaceFibre endpoint and router implementations are likely to minimize the number of distinct virtual channels supported in order to limit complexity and power dissipation characteristics. Such resource-constrained implementations must be allowed to participate in SpaceFibre networks equally with more capable implementations. The SpaceFibre Standard Draft should not impose requirements that unnecessarily favor one class of device implementation over another.

A SpaceFibre endpoint must provide a virtual channel that supports each QoS type used by the application. Since each application utilizing SpaceFibre endpoints will have unique QoS needs, the number of virtual channels and QoS types supported by any specific SpaceFibre endpoint device will be correspondingly unique. Because of the industry tendency to develop products tailored for a specific application, a wide variety of SpaceFibre endpoint devices with a correspondingly wide mix of capabilities are likely to be developed over time.

C. SpaceFibre Network Configuration

The need to associate identical QoS attributes with a virtual channel of each network link traversed by a SpaceWire packet increases the configuration overhead of SpaceFibre networks significantly compared to SpaceWire networks. Each SpaceFibre router and endpoint must be configured with the QoS characteristics of each virtual network it supports. The time needed to configure a SpaceFibre network will scale linearly with the number of network entities to be configured and, separately, with the number of virtual networks to be utilized.

A SpaceFibre link clearly must support at least one virtual channel in one direction to be useful (at least one virtual channel in each direction is the likely minimum implementation). Support for additional virtual channels is an implementation decision that will balance application needs against complexity and power dissipation constraints.

The SpaceFibre Standard Draft is ambiguous regarding VCID configuration. The likely interpretation is that the VCID can be written only once after the CODEC exits the cold-reset state. SpaceWire Working Group presentations [3] are consistent with that interpretation. The ambiguity is exacerbated by the fact that each VCB configuration defaults to a VCID of zero. There doesn’t appear to be any mechanism to guarantee virtual channel uniqueness (avoid multiple virtual channels with the same VCID, etc.).

The SpaceFibre Standard Draft does not establish mechanisms for negotiating virtual channel use between the ends of a SpaceFibre link. The standard draft apparently presumes that such negotiations are performed using a higher level of the protocol stack. It also does not address the methodology to be followed by compliant implementations when initializing virtual channels. Some concepts [3] assume that VC 0 is always assigned to a Configuration Virtual Network, presumably automatically configured following cold-reset to provide full-duplex best-effort access to SpaceFibre network devices.

We can infer that a Flow Control Token (FCT) issued by the link receiver for a specific VCID indicates that the receiver supports that virtual channel. Whether the link transmitter has any obligation to allocate resources to the virtual channel is unclear. Previous negotiation of support for the virtual channel at each end of the link would clearly be beneficial.

The mechanism for initializing virtual channels other than VCID 0 is likely intended to be implemented using a SpaceWire configuration protocol (SpaceWire PnP or similar). While not addressed by the standard, the SpaceFibre CODEC must provide the configuration protocol with a method for determining the available virtual channel capacity. Additionally, the SpaceFibre CODEC at each end of the link must expose the virtual channel configuration fields and provide a mechanism for enabling the virtual channel after configuration is complete.

V. Virtual Networks Using Identical Virtual Channels

The VCID-based virtual network abstraction [2] [3] offers a conceptually simple method for routing SpaceWire packets over SpaceFibre networks by using the VCID as a substitute for QoS type when making routing decisions. Figure 3 illustrates the concept by showing the use of VCIDs to route SpaceWire traffic through the virtual networks identified in Fig. 2. A virtual channel in every SpaceFibre link used by the virtual network must be assigned the identical VCID regardless of the number of virtual channels supported by the specific SpaceFibre link. Note that routing of SpaceWire packets through the virtual network is performed similarly to traditional SpaceWire routing.

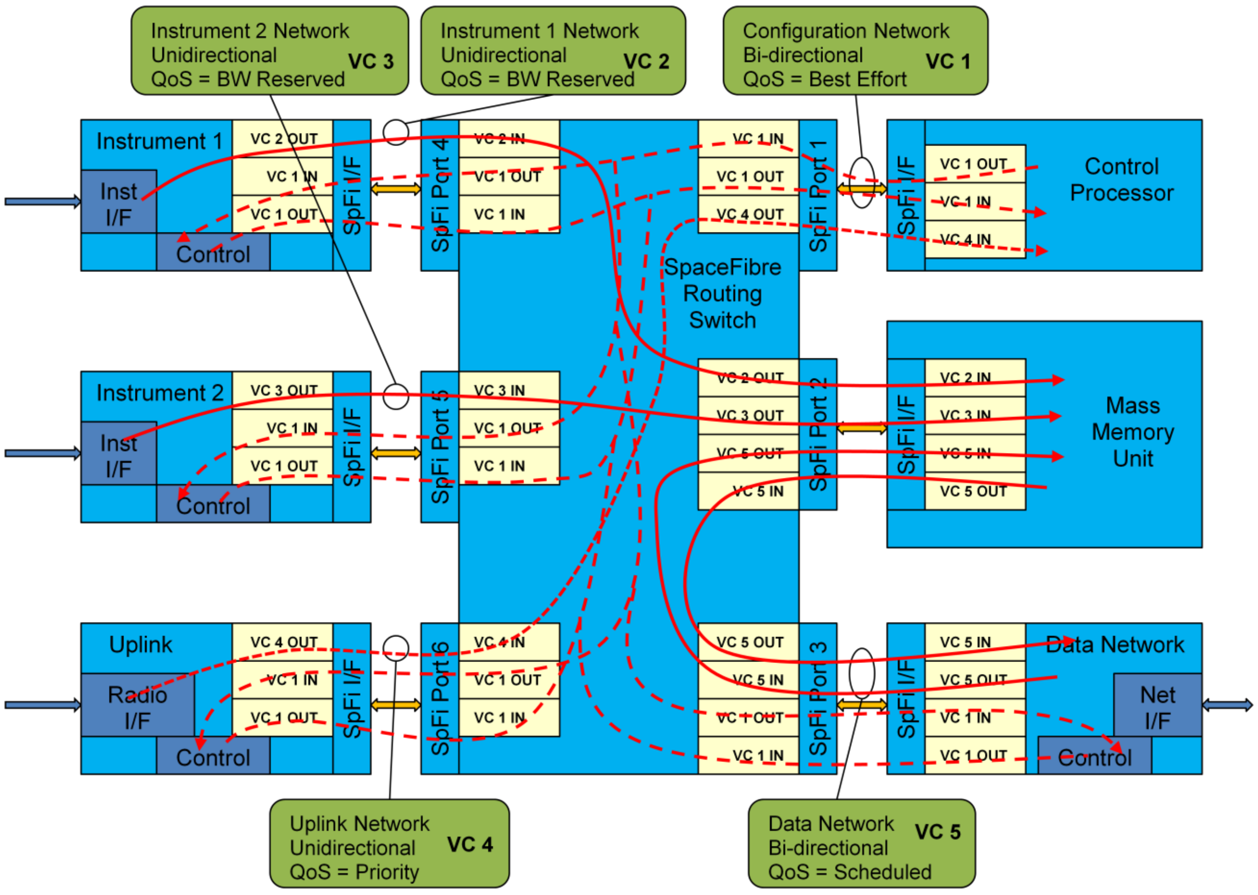


Fig. 3. Packet Routing Based on Virtual Channel Identifier

In the example shown in Fig. 3, the application creating SpaceWire traffic in Instrument 1 sends data to the Mass Memory Unit over the VCID2 virtual network. During network initialization, all of the network endpoints and routers must be configured with identical QoS attributes assigned to every virtual channel with a VCID of 2. Any violation of this rule causes inconsistent QoS behavior within the virtual network. If the application in Instrument 2 were to use the VCID2 virtual network to send packets to the Mass Memory Unit, the router must arbitrate between the two packet streams for access to VC2 of Port 2. When Instrument 2 uses the VCID3 virtual network as shown in Fig. 3, the router simply forwards the packets over the respective virtual channels of Port 2.

A. Limits on the Number of Virtual Networks

As mentioned in previous work [4], using the VCID as a QoS type identifier limits the number of QoS types that can be mapped onto the physical SpaceFibre link. By extension, using the VCID as a virtual network identifier limits the number of virtual SpaceWire networks that can be mapped onto the physical SpaceFibre network. While not a concern for the SpaceFibre networks considered in currently proposed applications, future SpaceFibre applications are likely to impose much more complex scenarios.

The number of virtual networks mapped to a physical network can be increased beyond the limit of 256 VCIDs in cases where a VCID can be used to identify multiple virtual networks. Such cases only arise if certain virtual networks have non-overlapping footprints within the overall physical network. The definition of non-overlapping virtual networks is dependent on the ability to establish isolation boundaries within the physical SpaceFibre fabric. An isolation boundary prevents SpaceWire packets on one virtual network from bleeding into a different virtual network when both virtual networks have the same VCID.

B. Virtual Channel Initialization

The SpaceFibre Standard Draft requires that SpaceFibre CODECs establish the VCID associated with each VCB after cold reset. The QoS parameters associated with a VCID are configurable using the CODEC link management interface. The VCID-based virtual network abstraction requires that assignment of the VCID be accomplished during network configuration.

C. Initial Virtual Channel Identifier Assignment

If simple SpaceFibre CODEC implementations statically assign the VCID of each virtual channel supported (sequential integers starting at zero, for example), the ability to design complex SpaceFibre networks with more than a few virtual networks is significantly limited. To support the VCID-based virtual network abstraction, any individual SpaceFibre endpoint might need to connect to multiple non-contiguous virtual networks within the VCID state space.

D. Virtual Channel Identifier Reassignment

Since a physical SpaceFibre network can be composed of many virtual SpaceWire networks, an endpoint containing a small number of virtual channels would benefit from the ability to dynamically connect and disconnect to any arbitrary virtual SpaceWire network desired. To support the VCID-based virtual network abstraction, such a capability would require reconfiguring a virtual channel to match the QoS attributes and VCID of the desired network. Since the virtual channel reconfiguration must be performed at both ends of the SpaceFibre link, the most appropriate method involves taking the virtual channel offline, reconfiguring each end of the link and bringing the virtual channel back online.

As currently specified in the SpaceFibre Standard Draft, the flow control counters associated with a virtual channel can only be initialized by reset or remote flush (link initialization). These events apparently affect the flow control counters of all virtual channels, making it very difficult to dynamically reassign a single virtual channel to a different virtual network (VCID) without taking the entire SpaceFibre link offline. This limitation is particularly acute for SpaceFibre endpoints with few virtual channels that need to support more QoS classes than the number of virtual channels available.

VI. Virtual Networks Using Arbitrary Virtual Channels

Many of the issues raised regarding virtual networks composed of virtual channels with identical VCIDs can be eliminated by allowing a virtual network to use any virtual channel that has been initialized with the appropriate QoS attributes. Creating virtual networks from arbitrary collections of virtual channels with identical QoS attributes allows much greater network design flexibility.

As previously mentioned, a virtual network is composed of virtual channels with identical QoS attributes. Every virtual channel in the virtual network must be initialized to have the same QoS attributes. The use of virtual channels is greatly simplified by introducing the concept of a Virtual Network Identifier (VNID) that identifies a specific QoS type (combination of QoS attributes).

A. Limits on the Number of Virtual Networks

The VNID concept allows practically unlimited scaling of the number of virtual SpaceWire networks that a SpaceFibre network can support. The limit on the number of virtual networks that can simultaneously use a single SpaceFibre link remains, but has little affect on the number of virtual networks that can be globally defined. An additional benefit is the low complexity associated with virtual channel initialization and VCID assignment.

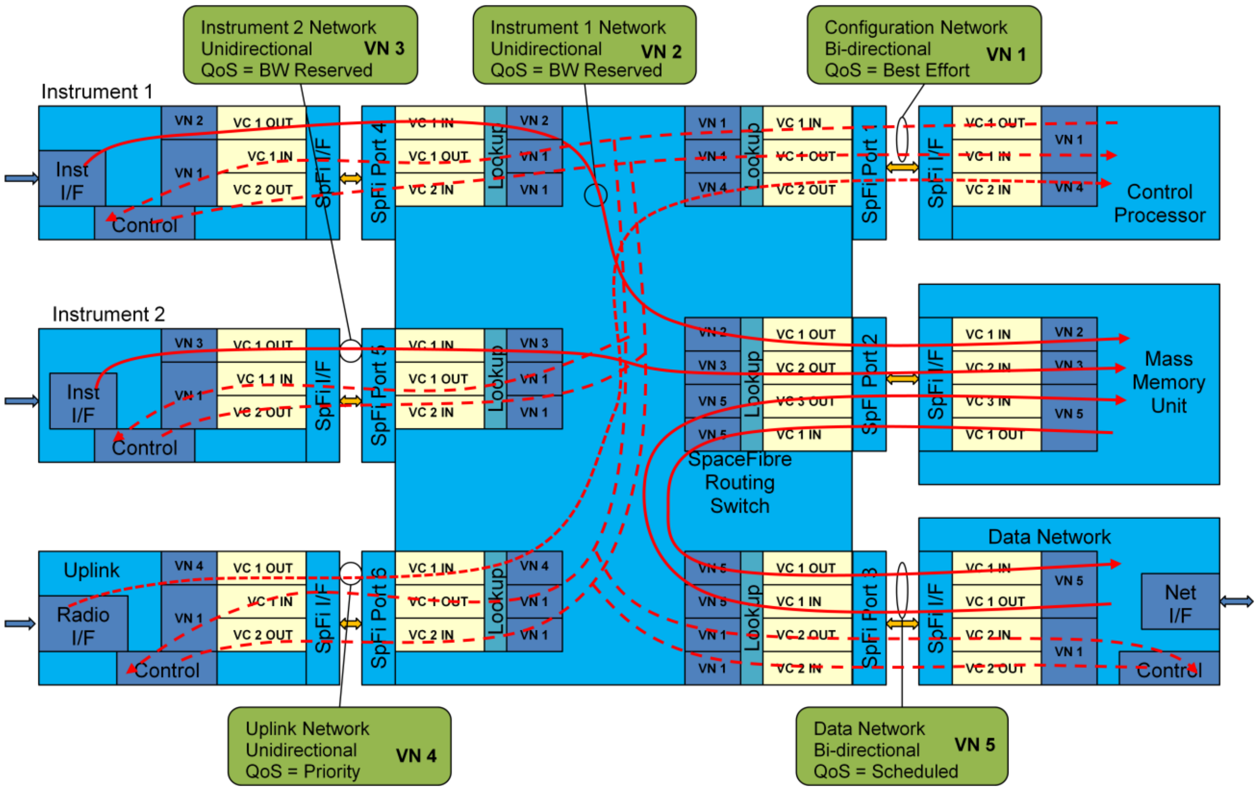
A specific set of QoS attributes is associated with each virtual channel using the corresponding VNID. While a virtual channel must be associated with one VNID, an individual VNID can be associated with more than one VCID on the same link. Note that the number of VNIDs supported by an endpoint device can exceed the number of VCBs available since the QoS attributes of a virtual channel are allowed to change without reinitializing the virtual channel.

Figure 4 illustrates the concept by showing the SpaceWire traffic routed through virtual networks identified by the VNID. Any virtual channel in a SpaceFibre link can be associated with the virtual network since the VCID is not used for network routing. The SpaceFibre router contains the mapping between the VNID and the associated VCID(s) of each router port. SpaceWire packets entering the router through a virtual channel are sent to the virtual SpaceWire network associated with that virtual channel. The virtual network routes the SpaceWire packets to the output port using traditional SpaceWire routing methods. The packets are transmitted over the output virtual channel associated with the virtual network.

In the example shown in Fig. 4, the application creating SpaceWire traffic in Instrument 1 sends data to the Mass Memory Unit over the VNID2 virtual network. During network initialization, all of the network endpoints and routers must be configured with identical QoS attributes assigned to every virtual channel associated with virtual network VNID2. Any violation of this rule causes inconsistent QoS behavior within the virtual SpaceWire network. When the application in Instrument 2 sends packets to the Mass Memory Unit over the VNID3 virtual network, the router sends the packet that entered over Port 5, VCID 1, out over Port 2, VCID 2. In the same vein, packets traversing the VNID1 (Configuration) virtual network are routed over links using a variety of VCIDs as shown in Fig. 4.

B. Initial Virtual Channel Identifier Assignment

Eliminating the use of the VCID as a virtual network identifier returns the VCID to its primary purpose as a link-level bandwidth allocation mechanism. As such, the VCID associated with a virtual channel has no relevance beyond the CODECs at each end of the SpaceFibre link.



Instrument 2 Network Unidirectional VN 3 QoS = BW Reserved
Instrument 1 Network Unidirectional VN 2 QoS = BW Reserved
Configuration Network Bi-directional VN 1 QoS = Best Effort
Instrument 1
Instrument 2
Inst I/F
Control
Uplink
Radio I/F
SpFi I/F
SpFi Port 4
SpFi Port 5
SpFi Port 6
SpFi Port 1
SpFi Port 2
SpFi Port 3
Lookup
SpaceFibre Routing Switch
Control Processor
Mass Memory Unit
Data Network
Net I/F
Uplink Network Unidirectional VN 4 QoS = Priority
Data Network Bi-directional VN 5 QoS = Scheduled

Fig. 4. Packet Routing Based on Virtual Network Identifier.

C. Virtual Channel Identifier Reassignment

Similarly, the need for reassigning VCID values is eliminated when the VCID is no longer used for network routing. An application can easily associate a different VNID with any available virtual channel to join another virtual SpaceWire network at will (the VNID reassignment must be applied to both ends of the SpaceFibre link).

D. Virtual Network Routing

Using VNIDs for routing decisions in SpaceFibre networks allows SpaceWire packets to traverse any virtual channel of the outbound port with the matching VNID. The complexity of routing decisions is not significantly greater than using VCIDs since the one-to-one mapping of incoming VCID to VNID can be viewed as a form of indirect addressing.

A characteristic of VNID-based routing is that VNID values do not need to be identical throughout the SpaceFibre network. The VNID value assignments in each router are independent of the assignments in any other router. As with VCID-based routing, however, the QoS attributes of an individual virtual SpaceWire network must be configured identically in every router. Note that there are practical benefits to treating the VNID values as virtual network identifiers and assigning them consistently throughout the SpaceFibre network.

VII. SpaceFibre Standard Improvements

We have identified an ambiguity in the SpaceFibre Standard Draft with regard to assignment of the initial VCID value to each virtual channel. In addition, the default VCID value of zero raises concern about the possibility of multiple virtual channels with the same VCID value. There is also a lack of clarity regarding the acceptable methods for negotiating virtual channel utilization between opposite ends of the SpaceFibre link.

A. Improvements for Virtual Networks Using Virtual Channel Identifiers

As discussed above, using VCIDs as virtual network identifiers restricts the use of resource-constrained endpoint devices and complicates the virtual network design task for system implementers. In the event that the SpaceFibre community chooses this method for SpaceFibre routing, a number of improvements to the SpaceFibre Standard Draft are appropriate.

1) Virtual Channel Initialization

Clarification of the virtual channel initialization method is important regardless of the SpaceFibre routing mechanism used. It is critical when the virtual channels must be mapped to an arbitrary VCID in order to be associated with a virtual SpaceWire network identified by that VCID.

The SpaceFibre Standard Draft must provide a detailed description of the CODEC features that make virtual channel initialization possible. It should also describe a process for negotiating virtual channel use between the ends of a SpaceFibre link.

2) Virtual Channel Identifier Configuration

Allowing dynamic VCID assignment improves the usability of resource-constrained endpoint devices in complex VCID-based SpaceFibre networks. An application needing to connect to more virtual SpaceWire networks than the attached SpaceFibre link can support should be able to reassign virtual channels to achieve the desired communication flexibility.

To achieve this capability, the SpaceFibre Standard Draft must provide a mechanism for independent reconfiguration of individual virtual channels. This mechanism should include the ability to clear the FCT counters associated with the virtual channel without affecting the operation of other virtual channels.

B. Using Virtual Network Identifiers for Virtual Network Routing

The improvements discussed above are generally unnecessary if a level of indirection is added to the virtual network routing mechanism. By identifying virtual SpaceWire networks using the VNID instead of the VCID, any virtual channel (regardless of the assigned VCID) associated with the VNID is a member of the virtual network. There is no longer a need to assign specific VCIDs to virtual channels, so SpaceFibre CODECs can hardwire the VCID of each supported virtual channel.

Because VNID-based virtual SpaceWire networks don't care about the VCID values used, the virtual channel initialization concerns raised above are less serious. The negotiation of virtual channel utilization between opposite ends of the SpaceFibre link can be significantly simplified.

VIII. SUMMARY

We believe that virtual SpaceWire networks based on VNID routing mechanisms is the simplest and most flexible solution for implementing SpaceFibre routers. The simple indirection method described makes the design of complex SpaceFibre networks much simpler than alternatives. In addition, the number of virtual SpaceWire networks that can be defined is not unnecessarily limited by the maximum number of virtual channels possible in any single SpaceFibre link.

REFERENCES

[1] S. Parkes, A. Ferrer, A. Gonzalez, and C. McClements, "SpaceFibre Standard draft E1," ECSS-E-ST-50-XXX, 28th September 2012

[2] S. Parkes and M. Suess, "Mixed SpaceWire - SpaceFibre networks," Proceedings of the 4th International SpaceWire Conference, November 2011, San Antonio, Texas, p. 144.

[3] S. Parkes, C. McClements, M. Dunstan, A. Ferrer, and A. Gonzalez, "SpaceFibre," SpaceWire Working Group Meeting 19 Session 1 - Revised SpaceFibre specification, October 2012.

[4] C. Kimmery, "SpaceFibre virtual channels and flow-control," Proceedings of the 2nd International SpaceWire Conference, November 2008, Nara, Japan, pp. 37-38.

Standardisation 2 (Long)

GigaSpaceWire – Gigabit Links for SpaceWire Networks

Standardisation, Long Paper

Evgeny Yablokov, Yuriy Sheynin, Elena Suvorova, Alexander Stepanov
Institute of High-Performance Computer and Network Technologies
St. Petersburg State University of Aerospace Instrumentation
St. Petersburg, Russian Federation
Evgeny.Yablokov@guap.ru, Sheynin@aanet.ru, Suvorova@aanet.ru, Alexander.Stepanov@guap.ru

Tatiana Solokhina, Yaroslav Petrichcovitch, Alexander Glushkov, Ilia Alekseev
R&D Center ELVEES Company,
Moscow, Russia
elcore@elvees.com

***Abstract*—SpaceWire network technology is intended to be used for spacecraft on-board communication. Providing low implementation and overhead costs as well as hard real-time communication services, currently SpaceWire fails to meet the latest communication system requirements in the fields of data transmission rate, galvanic isolation, cable length and cable mass raised by world-wide space industry. This paper discusses the new physical layer for SpaceWire called GigaSpaceWire which is aimed to make SpaceWire networks satisfying the requirements.**

***Index Terms*—GigaSpaceWire, gigabit links, galvanic isolation, standardisation.**

I. Introduction

SpaceWire technology becomes a general interconnection technology in national and international missions. As its applications become more and more diverse, the constraints of SpaceWire limit its usage in next generation demanding missions. In accordance with analyses made by representatives from the Russian and European [1] and US space industries [2], the main SpaceWire problems are: 1) lack of galvanic isolation; 2) cable length is limited to 10 m distance; 3) limited data rates, while gigabit rates are demanded for new missions; 4) lack of QoS that is required for real-time control. New developments to overcome these problems are on the way (e.g. SpaceFibre, SpaceWire-RT). However they are in the course of development (and would be for a couple of years, at least, before would be finally fixed) and their great features would be not free. Overheads for them would be reasonable where they are actually needed and would be a burden where not.

Analysis shows that the first three of four main constraints of SpaceWire could be solved just now without considerable problems and overheads.

In order to enhance link characteristics for SpaceWire networks this paper describes the GigaSpaceWire technology that has been developed by St. Petersburg State University of Aerospace Instrumentation and ELVEES Company. GigaSpaceWire provides gigabit link technology with longer distances and galvanic isolation capability for SpaceWire networks. The GigaSpaceWire standard has been previously introduced in paper [3] and its specification has been developed [4] and is considered now as a part of the Russian national SpaceWire-based standard draft.

The core principle of GigaSpaceWire technology is to substitute DS encoding scheme with 8b10b encoding which is currently used in a wide number of communication standards, e.g in such as Fibre Channel [5] and Serial RapidIO [6]. Consequently, galvanic isolation can be implemented, the maximum transmission rate can be raised up to 2.5 Gbit/s (5 Gbit/s in future), the maximum cable length can be increased to 100 m with cable mass significantly reduced.

II. GigaSpaceWire Protocol Stack

The GigaSpaceWire protocol stack is shown in Fig. 1. It contains new character, encoding and PHY layers, the modified SpaceWire exchange level and the conventional SpaceWire packet and network levels.

The PHY layer principal task is to transmit and receive a raw bit sequence over a physical link. In order to perform it, the PHY layer receiver establishes bit synchronization before starting the reception of data from the link. When bit synchronization is achieved, at the receiving side the PHY layer accepts bit stream from the physical link, performs de-serialization and symbol alignment and transmits 10-bit code sequences to the encoding layer. The transmitting side of the PHY layer accepts 10b code sequences from the encoding layer, serializes them and sends bit-wide stream to the physical link.

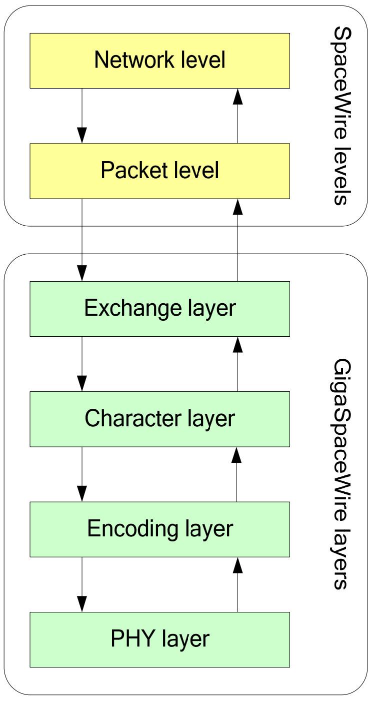


Fig. 1. The GigaSpaceWire protocol stack

The encoding layer performs the 8b10b encoding. At the transmitting side the encoding layer accepts 8b symbols from the character layer and substitutes them with correspondent 10b code sequences. Accordingly, at the receiving side the encoding layer accepts 10b code sequences from the PHY layer and transforms them into 8b symbols.

The exchange layer manages the point-to-point connection over the link. After reset the exchange layer tries to establish bi-directional connection with the exchange layer entity of the remote side. If the connection is acquired, the upper layers are allowed to send SpaceWire packets, Time-codes and Distributed Interrupt codes over the link and the exchange layer performs flow control, data rate adjustment and connection maintenance functions. The data stream through the GigaSpaceWire protocol stack is depicted in Fig. 2.

The SpaceWire packet and network levels are not changed. It provides easy GigaSpaceWire links integration into regular SpaceWire networks.

The purpose of the GigaSpaceWire technology is to substitute DS encoding, which is used in the SpaceWire standard, with 8b10b encoding. In turn, it brings to SpaceWire networks such features as possibility for galvanic isolation and long distance data transmission at the rate of several gigabits. However, introduction of the 8b10b encoding into a SpaceWire link causes considerable changes in such basic elements of SpaceWire technology as the exchange level state machine, the silence exchange procedure, the flow control mechanism and the encoding of SpaceWire characters and codes as well as the complete substitution of the signaling and physical levels with the new GigaSpaceWire PHY layer. However, beyond these unavoidable changes the GigaSpaceWire technology does not attempt to introduce new services that are irrelevant to SpaceWire. The principle SpaceWire features that are not affected by GigaSpaceWire are data and control interfaces that link interface offers to the upper layers.

In the following sections the key principles that have been changed in the GigaSpaceWire technology are discussed in detail.

III. Exchange Layer State Machine

The state machine which is deployed in the GigaSpaceWire exchange layer is given in Fig. 3. The state machine has the same set of states as the SpaceWire exchange level state machine but changes both the rules managing the transmission among the states and operations of transmitter and receiver in particular states.

The most significant difference between the state machines of GigaSpaceWire and SpaceWire consists in the actions that are performed in the ErrorReset, ErrorWait and Ready states. While the PHY layer technology which is incorporated in GigaSpaceWire can require considerable time to make the PHY layer transmitter and receiver ready for communication, the GigaSpaceWire state machine may not disable them each time when the ErrorReset state is entered. Therefore, in order to maintain the bit synchronization and symbol alignment established at the PHY layer, the exchange layer state machine enables the transmitter in all the states. However, in the ErrorReset, ErrorWait and Ready states the transmitter is permitted to send only filler symbols, which are called as IDLE symbols. Simultaneously, the exchange layer receiver is disabled in the ErrorReset state while the PHY layer receiver still operates.

GigaSpaceWire decreases both the 6.4 us and 12.8 us timeouts which manage the transitions from the ErrorReset to the ErrorWait and from the ErrorWait to the Ready states respectively. As in GigaSpaceWire the silence exchange procedure(described below) takes considerably less time and the transmitter and receiver of the PHY layer are never disabled upon entering the ErrorReset state, the durations of both the timeouts are shorted to 5.12 us.

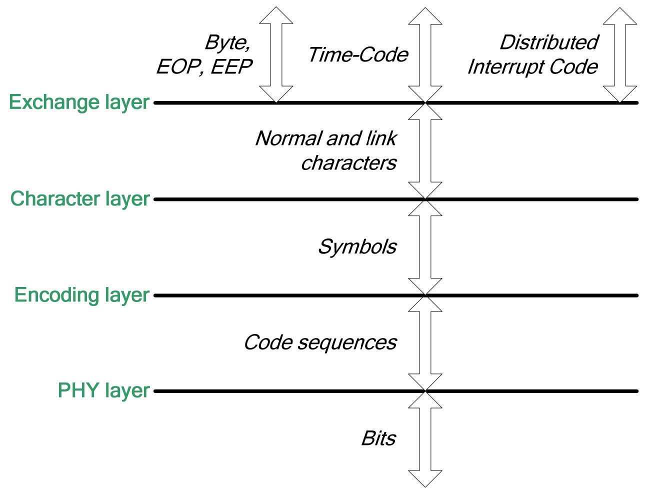


Fig. 2. The GigaSpaceWire data flow

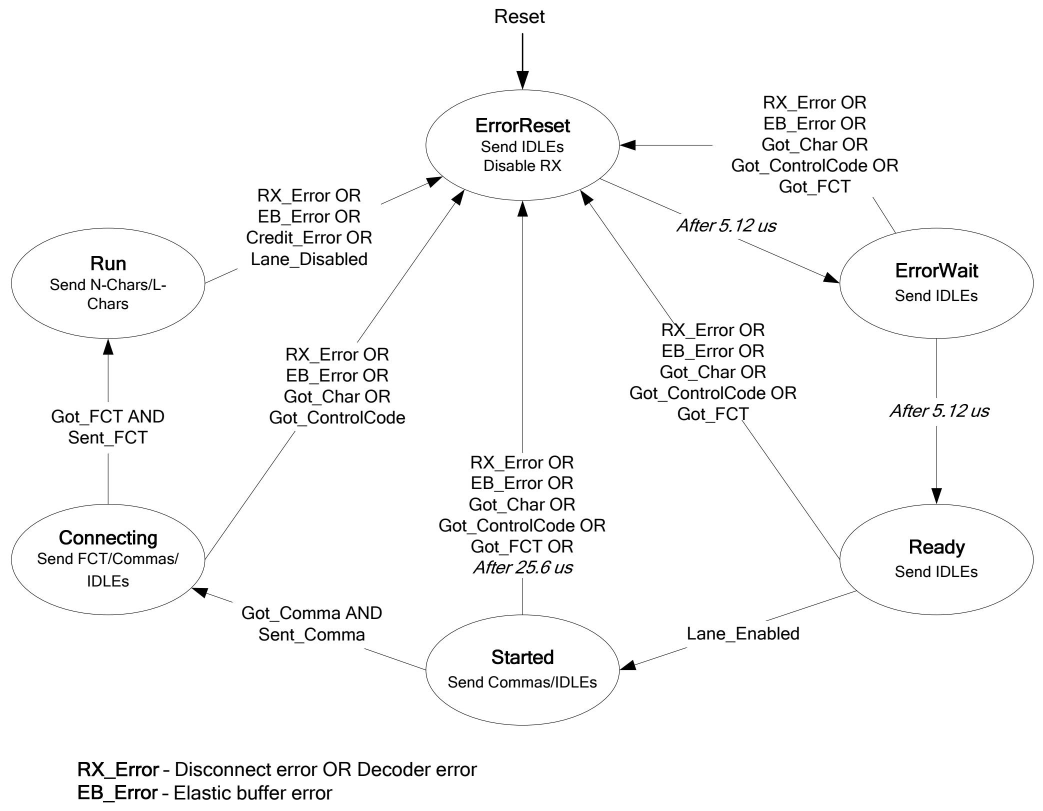


Fig. 3. The GigaSpaceWire exchange layer state machine

Duration of the timeout in the Started state is increased significantly from 12.8 us to 25.6 us. It is assumed (but not required) that the PHY layer and the exchange layer state machine are enabled simultaneously. Therefore, while powering on the 25.6 us timeout in the Started state is dedicated to provide enough time for both ends of the link to lock PLLs and establish bit synchronization. After the connection at the PHY layer is established, the 25.6 us timeout is not expected to be used.

While IDLE characters, which are used in GigaSpaceWire as NULL codes in SpaceWire, must be sent in any state, the Got_IDLE condition becomes meaningless. Consequently, in GigaSpaceWire Comma characters are deployed at the connection establishment procedure instead of NULL codes that are used in SpaceWire. In the ErrorWait, Ready or Started state the reception of the first Comma permits error detection. In the Ready state the reception of the first Comma enables the transition to the Started state if the Autostart mode is set. Also, the transmission from the Started to the Connecting state is conditional on the got first Comma condition and the sent first Comma condition. After the exchange of the initial Comma characters is done and the state machines at the both sides of the link are entered the Connecting state, Comma symbols must be sent periodically to maintain the connection.

The SpaceWire exchange level state machine uses the gotNULL and gotFCT conditions while transitioning from the Started to the Connecting and from the Connecting to the Run state accordingly. However, in the both cases these got conditions, which are not accompanied by the corresponded sent conditions, are not enough for the reliable connection establishment. Especially, this concerns the case of entering the Run state from the Connecting state upon the reception of the first FCT.

Consider a situation when the side A of the link has enough space in the receive buffer to send an FCT but the side B does not. Thus, the side B gets at least one FCT from the side A and, consequently, enters the Run state. However, after entering the Run state the side B must send SpaceWire packets or Time-codes or Interrupt-codes if any requested. At the same time the side A keeps to be in the Connecting state until the reception of an FCT from the side B or the expiration of the 12.8 us timeout. As a result, if the side B starts the transmission of user data before sending at least one FCT, these data will be definitely lost causing the side A to enter the ErrorReset state.

In order to overcome the potential problem in the GigaSpaceWire exchange layer state machine the transitions from the Started to the Connecting and from the Connecting to the Run state must be performed only when the both correspondent got and sent conditions are met. While this keeps the link from the lost of user data as in the case discussed above, the usage of both the Got_FCT and Sent_FCT conditions in the Connecting state also eliminates the need of the 12.8 us timeout. Consequently, if at least one side of a GigaSpaceWire link does not have enough room to send one FCT, the connection establishment will be suspended at the

Connection state until the both side have the necessary space in the buffer or the connection will be closed.

IV. Connection Maintenance and Silence Exchange

GigaSpaceWire connection includes implicitly the PHY layer connection and the exchange layer connection. The exchange layer connection is managed by the correspondent state machine that is discussed in the previous section. While establishing the PHY layer connection the receivers at the both sides of the link must acquire bit synchronization and then symbol alignment before considering the incoming data to be valid.

It is assumed that the establishment of both the exchange layer connection and the PHY layer connection is started simultaneously. However, until the receiver of the local PHY acquires bit synchronization and symbol alignment the exchange layer receiver does not get any valid data from the link. Since the PHY layer connection is established, the exchange layer can receive data sent by the remote side.

The GigaSpaceWire error recovery procedure follows the SpaceWire error recovery procedure. This means that in case of any error, the side which has detected the error must indicate this to the remote side and then both the sides must re-establish the connection. To indicate an error to the remote side a SpaceWire node disables its transceiver at least for 19.2 us that causes a disconnect error at the remote side. However, GigaSpaceWire cannot adopt this approach. While the establishment of bit synchronization and symbol alignment at the PHY layer can take considerably long time, the PHY layer connection must not be broken each time when a link error occurs. Therefore, as the transceiver of GigaSpaceWire link interface must not be disabled upon error detection, the GigaSpaceWire technology has to adopt another approach to implement the silence exchange procedure.

In order to develop its silence exchange procedure in another way GigaSpaceWire introduces new link characters that are called as Comma characters. In the Started, Connecting and Run states the transmitter must periodically insert a Comma character into the outcoming data flow that allows the other side of the link to monitor whether the connection is alive or not.

A. GigaSpaceWire silence exchange

The GigaSpaceWire silence exchange procedure, which is depicted in Fig. 4, is organized as following. When a receiver of the side A of a GigaSpaceWire link detects an error, the exchange layer state machine must leave its current state and enter the ErrorReset state. However, as was discussed above, the GigaSpaceWire transmitter does not cease its operation upon the entrance to the ErrorReset state and continues to transmit filler characters being in the ErrorReset, ErrorWait and Ready states. On the other hand, so as to indicate the detected error to the side B the GigaSpaceWire transmitter terminates the insertion of Comma characters into the outcoming data flow. Meanwhile, when the receiver at the side B has not received a Comma for a predefined time interval called as the disconnection time interval (T_D), a disconnect error must be detected and its exchange layer state machine shall enter the ErrorReset state. As a result, the both sides of the link have entered the ErrorReset state and start to re-establish the connection.

To complete the silence exchange procedure the side A, which has initiated the re-connection, must disable its receiver at the exchange layer until the remote side B detects the disconnect error and enters the ErrorReset state. Otherwise, if the side A left the ErrorReset state when the side B has not ceased sending Comma characters into the link yet, it could received one of these last Comma characters as the first Comma. Eventually, when the side B entered the ErrorReset state, the side A in turn would detect a disconnect error and enter the ErrorReset state again.

To avoid this situation the duration of the ErrorReset state timeout has to be long enough to ensure that the “Comma silence” has propagated over the link in both directions. To define the Comma silence propagation time, consider the worst case when the side A detects an error just after its transmitter has introduced a Comma character into the outcoming data flow. Consequently, the Comma silence propagation time has to include the symbol propagation time (T_{prop} that is discussed below) over the GigaSpaceWire link. Then, it means that for the disconnect interval (T_D) the side B will continue sending Comma characters as it consider the connection to be valid. Only upon the expiration of the T_D timeout the side B will stop sending Comma characters. Thus, the Comma silence propagation time has to include the disconnect interval as well. Finally, considering the worst case possibility that the last Comma has been sent from the side B just before the T_D timeout expiration, the Comma silence propagation time must be incremented by the symbol propagation time again. Formally, this means that the ErrorReset timeout duration (T_{ER}) must be not less than the doubled symbol propagation time T_{prop} plus the disconnect interval T_D:

$$T_{ER} \geq 2 \cdot T_{\mathrm{Pr}op} + T_D \,. \tag{1}$$

In accordance with Eq. 1 the GigaSpaceWire standard sets duration of the ErrorReset timeout to the value of 5.12 us.

B. GigaSpaceWire clock tolerance compensation

GigaSpaceWire also uses Comma characters so as to compensate possible deviations between the transmit clocks at both sides of the GigaSpaceWire link.

Accordingly to the specification the transmit clock accuracy must not exceed +/- 300 parts per million (ppm). Therefore, the worst case clock difference between the transmit and receive clocks of a link occurs when the one side deviation is at +300 ppm and the other side deviation is at -300 ppm, resulting in a 600 ppm difference. A receiver elastic buffer is intended to compensate in the receiver the difference between the transmit and receive clocks.

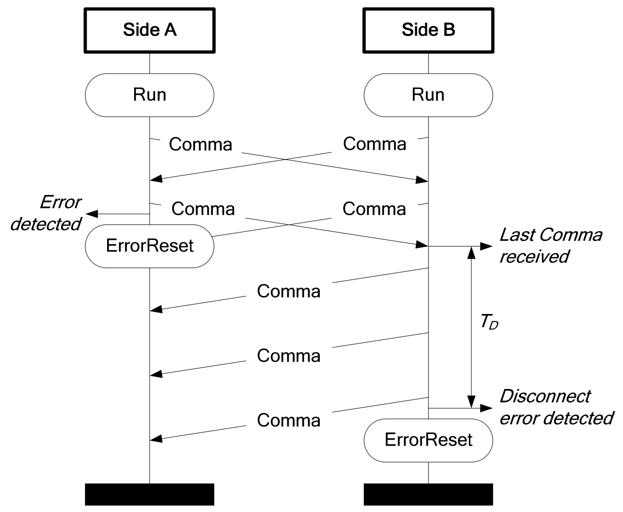


Fig. 4. The GigaSpaceWire silence exchange procedure

Given that in the Started, Connecting and Run states the transmitter periodically inserts Comma characters into the outcoming data flow, the receiver must manages them to keep its elastic buffer in the half-full / half-empty state. When a Comma is received and the elastic buffer is more than half-depth full then the Comma shall be ignored, i.e. not be written into the elastic buffer. When a Comma is read from the elastic buffer and the elastic buffer is empty than half-depth then the Comma shall be read twice from the elastic buffer.

V. Flow Control

The application field of GigaSpaceWire link can be divided into two broad categories: a connection between two usual GigaSpaceWire nodes or a connection between two SpaceWire nodes which need galvanic isolation or long distance cable between them. In the last case each SpaceWire node is connected through SpaceWire interface to a GigaSpaceWire-SpaceWire bridge and these bridges are connected via GigaSpaceWire link as it depicted in Fig. 5.

For these two different use cases GigaSpaceWire provides two distinctive flow control modes: the new GigaSpaceWire flow control mode and the traditional SpaceWire flow control mode. The GigaSpaceWire flow control is dedicated primarily for a direct connection of two GigaSpaceWire nodes and the SpaceWire flow control is recommended to be deployed when a GigaSpaceWire link connects two SpaceWire nodes by means of GigaSpaceWire-SpaceWire bridges.

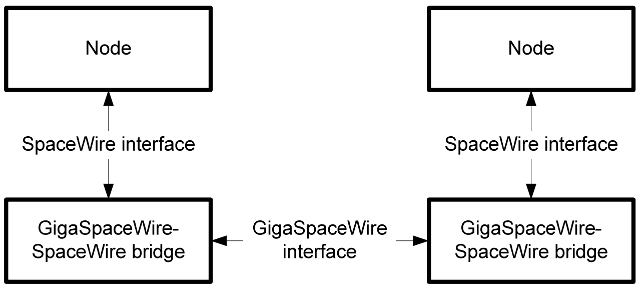


Fig. 5. Connection of two SpaceWire nodes via GigaSpaceWire link

The GigaSpaceWire flow control mode has the same protocol as the flow control, which is defined by the SpaceWire specification, but sets different values for the parameters. In the GigaSpaceWire flow control one Flow Control Token credits transmission of 32 symbols instead of 8 characters as in SpaceWire. Also, in the GigaSpaceWire flow control the maximum permitted value of both the credit counter in the transmitting side and the outstanding counter in the receiving side is 512 when the SpaceWire flow control system restricts the same parameters by the value of 56.

The chosen values were derived from the analysis of the symbol propagation time over GigaSpaceWire link (T_{prop}). Such symbol propagation time consists of the symbol propagation time via the transmiter logics, the transmitter SerDes, the cable, the receiver SerDes, the elastic buffer and, finally, via the receiver logic:

$$T_{\mathrm{Pr}op} = T_{Logic} + T_{ElasticBuf} + T_{SerDes} + T_{Cable}, \qquad (2)$$

where T_{Logic} is the sum of the propagation times over the transmitter and receiver logic, T_{SerDes} is the sum of the propagation times over the transmitter and receiver SerDes, $T_{ElasticBuf}$ is the elastic buffer delay and T_{Cable} is the propagation time over the cable.

In turn, the propagation time over the cable depends on the cable length (L_{cable}), the transmitter frequency (F_{Tx}), the signal propagation delay over the medium (T_S, measured in *ns*) and the ration of the local frequency to the transmitter frequency (k_{Fr}):

$$T_{Cable} = \lceil k_{Fr}(L_{Cable} \cdot F_{Tx} \cdot T_S) \rceil. \qquad (3)$$

The minimum number of credits (*MNC*) is the least value of the credit and outstanding counters that allows utilization of the whole capacity of the given link at the given local and transmitter frequencies for the given implementation of GigaSpaceWire link interface. As a portion of the link capacity is utilized by the link control information (e.g. by Comma and FCT characters) as well as by the SpaceWire Time codes and Distributed Interrupt codes, only the link capacity, which is available for packet data ($C_{User)}$ should be taken into account:

$$\mathrm{MNC} = \lfloor C_{User}(2T_{\mathrm{Pr}op} + T_{FCT} + T_{Tx} + (FCT - 1)) \rfloor, \qquad (4)$$

where T_{FCT} is the FCT generation time, T_{Tx} is expected FCT transmission delay caused by the transmission of other GigaSpaceWire control codes and characters with higher priority (i.e. Comma, Time-codes, etc) and *FCT* is the number of normal characters which can be sent in response to one FCT.

Therefore, in order to utilize the whole capacity of GigaSpaceWire link the maximum permitted value of the credit and outstanding counters must be not less than the minimum number of credits (*MNC*). To calculate the symbol propagation time and the minimum number of credit values close enough to the practical experience, the initial parameters in Eq. 2, 3 and 4 should be not the worst case but a typical case. In accordance with this principle and for the current version of the electrical specification the minimum number of credits is 273 given the FCT value of 32. However, taken into account possible development of the GigaSpaceWire electrical specification (e.g. an incrise in the data transmission rate up to 2.5 Gbit/s or more) the maximum permitted value of the credit and outstanding couters is set to 512. For the implementations intended to satisfy the current version of the GigaSpaceWire standard the recommended depth of the receive buffer is 288 characters.

The GigaSpaceWire flow control system requires considerably more buffer space than the SpaceWire flow control. However, this is obvious that when a GigaSpaceWire connection is used to connect two SpaceWire nodes, the whole gigabit capacity of GigaSpaceWire link cannot be utilized because the corresponded SpaceWire interface supplies the data at the maximum rate of 400 Mbit/s. For this reason GigaSpaceWire adopt the SpaceWire flow control as well. As some devices can be trageted at only bridge functions, they can implement only the SpaceWire flow control system. Therefore, the SpaceWire flow control system is chosen to be obligatory and the GigaSpaceWire flow control system is set to be optional.

VI. Characters Encoding

GigaSpaceWire deploys 8b10b encoding scheme instead of Data-Strobe encoding, which is used in SpaceWire. While not all SpaceWire characters correspond to the 8b symbol format, the GigaSpaceWire character layer is responsible for the representation of SpaceWire characters into proper 8b symbols. This means that all characters that are transmitted over the GigaSpaceWire link have the same length of 10 bit.

Each of the GigaSpaceWire link characters, which comprise Comma, FCT and IDLE characters, as well as the End of Packet and Error End of Packet markers are encoded by single 8b10b K-codes. Each byte of the SpaceWire packet destination address and packet cargo is encoded by a single D-code.

Each SpaceWire Time-code, Interrupt-code or Interrupt_Acknowledge-code is encoded by two consecutive 8b symbols. The first symbol is a dedicated K-code and the second symbol is a D-code, which includes the control code identifier as the two or three most significant bits and the value of the control code as the other six or five bits respectively. Transmission of the two symbols of a Time-code, Interrupt-code or Interrupt_Acknowledge-code must not be preempted by any other character.

VII. Conclusion

The presented GigaSpaceWire communication technology is a practical modification of the low levels of the SpaceWire protocol stack.

So as to improve the SpaceWire physical layer capabilities, GigaSpaceWire technology acquires 8b10b encoding scheme. The GigaSpaceWire protocol stack inherits the packet and network levels of SpaceWire and defines new PHY, endocing, character and exchange layers. The deployment of SpaceWire upper layers guarantees that any application compatible with SpaceWire technology can use GigaSpaceWire links as well. GigaSpaceWire low layers implement 8b10b encoding and ensure transparency of the physical layer technology for the SpaceWire upper layers.

GigaSpaceWire changes only those SpaceWire features which cannot be used in relation with 8b10b encoding.

Bit synchronization and the lock of PLLs, which are performed by the GigaSpaceWire PHY layer, usually cannot be acquired in a relatively short time. For this reason, GigaSpaceWire rejects the SpaceWire error recovery scheme that consists in disabling transmitter in case of an error so as to indicate the error to the remote side. Instead of this, GigaSpaceWire introduces special link control characters, named as Comma characters. These characters are periodically inserted by the transmitter into the outcoming data flow covering the broad field of tasks. Especially, the absence of Comma characters among the incoming data indicates to the receiving side that the remote side of the link has entered the ErrorReset state.

GigaSpaceWire offers two different flow control approaches. The default approach is the SpaceWire flow control and the optional one is a new GigaSpaceWire flow control. The GigaSpaceWire flow control has the same principles as the SpaceWire one, but increases the key parameter values: one FCT credits transmission of 32 normal characters and the maximum number of outstanding credits is 512. The GigaSpaceWire flow control is introduced to utilize the whole transmission capacity of links with gigabit rates. On the other hand, the SpaceWire flow control can be used as well, especially when two SpaceWire nodes communicate via GigaSpaceWire link bridges.

GigaSpaceWire makes changes in a link only and fits in the general SpaceWire network architecture. It could be embedded into a network controller, a routing switch or a simple link type converter (e.g. a GigaSpaceWire-SpaceWire bridge). Integration of GigaSpaceWire into a SpaceWire network is relatively easy: change the links you need for higher throughput, longer distances or galvanic isolation and use the common SpaceWire network infrastructure elsewhere.

By the moment, the GigaSpaceWire technology has been successfully prototyped in FPGA and already implemented in a number of chips “Multiboard” produced by ELVEES. The implementation of the GigaSpaceWire link controller is protected by patents [7].

References

[1] S. Parkes, “D2.1 SpaceWire-RT outline specification,” SpaceWire-RT Consortium, September 2012.

[2] D.A. Gwaltney and J.M. Briscoe, “Comparison of communication architectures for spacecraft modular avionics systems”, Marshall Space Flight Center, Alabama, June 2006.

[3] E. Yablokov, Y. Sheynin, E. Suvorova, T. Slokhina, A. Glushkov and I. Alekseev, “Gigabit links in SpaceWire networks”, Questions on radio electronics, Scientific Research Institute “Electronics”, vol. 1, pp. 24-36, April 2012.

[4] St. Petersburg State University of Aerospace Instrumentation and R&D Center ELVEES Company, “GigaSpaceWire Specification Ver. 2.1”, unpublished.

[5] Fibre Channel Framing and Signaling (FC-FS) Rev. 1.80, INCITS working draft proposed American National Standard for Information Technology, American National Standards Institute, Inc., March 2003.

[6] RapidIO Interconnect Specification Part 6: LP-Serial Physical Layer Specification Rev. 2.2., RapidIO Trade Association, June 2011.

[7] Patent RU 12616 U1, The Russian Federation, “Communication interface for SpaceWire network”, Y. Sheynin, E. Yablokov, E. Suvorova, S. Gorbachev, T. Solokhina, Y. Petrichcovitch, A. Glushkov and I. Alekseev, March 2013.

Distributed Interrupt Signalling for SpaceWire Networks

Standardisation, Long Paper

Sergey Gorbachev, Ludmila Koblyakova, Yuriy Sheynin, Alexander Stepanov, Elena Suvorova
Institute of High-Performance Computer and Network Technologies
St. Petersburg State University of Aerospace Instrumentation
St. Petersburg, Russian Federation
sergvgor@yandex.ru, luda_o@rambler.ru, sheynin@aanet.ru, alexander.stepanov@guap.ru, suvorova@aanet.ru

Martin Suess
ESA / European Space Research and Technology Centre
Noordwijk, the Netherlands
martin.suess@esa.int

***Abstract*— SpaceWire is a standard for spacecraft on-board communication systems for transmission of both payload and control traffic. However, while SpaceWire mostly meets the requirements imposed by data computing and data handling applications, it fails to fully satisfy the requirements raised by on-board control loops. In order to resolve the problem this paper presents the final version of the Distributed Interrupt mechanism aimed at covering the area of hard-real time signal distribution in SpaceWire networks. The mechanism is dedicated primarily for transmission of short and low-frequent alarm messages and critical commands. The described Distributed Interrupt mechanism is intended to be included in the Revision 1 of the SpaceWire standard which is currently being drafted by ECSS.**

***Index Terms*—hard real-time signalling, Distributed Interrupts, Time-codes, standardisation.**

I. Introduction

One of the main advantages of the SpaceWire technology [1] is the ability to be an integrated communication infrastructure for spacecraft on-board networking. SpaceWire-based networks can accommodate different types of traffic that in previous generation of on-board networking used to be implemented by a set of separate interconnections that follow different standards. SpaceWire networks can integrate data stream traffic, packet traffic for distributed processing, command traffic for control, time stamps etc.

It is reasonable to complement the already available SpaceWire features with a hard real-time signal delivery service so as to substitute dedicated signalling lines that are used in onboard systems today and to integrate them in a common networking infrastructure. Hard real-time signalling imposes strict signal delivery constraints and requires high reliability of signal delivery [2]. However, the ECSS-E-50-12C standard SpaceWire services do not address these requirements up to now. The SpaceWire packet transfer service cannot ensure guaranteed low-latency massage delivery in an arbitrary network topology due to the possibility of congestion with other traffic. Though the SpaceWire Time-code service does ensure low-latency delivery of time-stamps, it cannot be used for transmission of a variety of hard-real time signals.

This paper describes the final version of the Distributed Interrupt mechanism, which is already an established approach for hard real-time signalling in SpaceWire networks [3, 4], and its applications. The Distributed Interrupt mechanism key features are ultra low signal delivery latency, simple configuration and high reliability of delivery.

II. Requirements for Distributed Interrupt Signalling

Since the Distributed Interrupt signalling mechanism is dedicated to substitute side-band signal wiring in on-board communication systems, it has the same aims and requirements. The main purpose of the Distributed Interrupt mechanism is transmission of system-critical urgent signals and commands, e.g. alarm signals.

The major requirements to the side-band signalling that are set by Russian and the European space industry cover latency, message transmission rate and reliability [2]. Thus transmission latency shall be less than 1 µs per link. In order to ensure high reliability of the signal delivery, the mechanism shall provide broadcast transmission, automatic acknowledgment of signal delivery and fault detection, isolation and recovery policies.

III. Key Principles

The Distributed Interrupt mechanism uses broadcast distribution of hard real-time signals providing ultra-low delivery latency and high reliability.

The low transmission latency of Distributed Interrupt codes is allowed by low control code size and a high priority level compared to other SpaceWire codes and characters. A Distributed Interrupt code consists of the 4-bit SpaceWire

Escape character followed by a 10-bit SpaceWire data character; the total size of the distributed Interrupt code is 14 bits. Distributed Interrupt codes take priority over SpaceWire FCT characters, data characters and NULL control codes. Therefore, the transmission of Interrupt signals is not affected by data packets flowing through the same links. Moreover, as Distributed Interrupt signalling is not managed by the SpaceWire flow control mechanism, Interrupt distribution can be performed even over links that are blocked by data packets e.g. in case of congestion. The only SpaceWire codes that have the higher priority level than Distributed Interrupt codes are Time-codes. However, transmission of Time-codes should not have significant impact of the Interrupt signals distribution because Time-codes are not expected to be sent often.

As a SpaceWire control code, each 14-bit Interrupt code carries 8-bit data field which, in turn, contain 3-bit code identifier and 5-bit Interrupt identifier. The 3-bit code identifier is used to distinguish Interrupt codes from other SpaceWire control codes (e.g. Time-codes) as well as to determine the type of Interrupt code. There are two types of Distributed Interrupt codes. Each Interrupt request has a particular 5-bit Interrupt identifier that is used to distinguish this request from other Interrupt requests in the network. Therefore, in a network there may be up to 32 different Interrupt requests with identifiers from 0 to 31. It is assumed that for any Interrupt request in the network there is at least one node that is assigned to receive and process the code. Such node is called an Interrupt handler. When an Interrupt handler receives an Interrupt request which this handler is assigned to process, it may issue a confirmation code that is called an Interrupt acknowledgment, which is another type of Interrupt code. Each Interrupt acknowledgment has the same Interrupt identifier as the correspondent Interrupt request.

Broadcast distribution of Interrupt codes allows simple configuration of the network that does not require routing tables in switches. In case of hardware redundancy in the network, broadcast distribution leads to higher reliability of Interrupt code delivery. However, broadcast distribution in networks with circular connections may lead to repeated propagation of Interrupt codes. So as to overcome the problem each SpaceWire switch or node, which supports the Distributed Interrupt mechanism, has a 32-bit Interrupt Source Register (ISR). Each *i*-th ISR bit corresponds to the Interrupt identifier with the same number. When a node issues an Interrupt request or a node or a switch receives an Interrupt request, the correspondent bit of the ISR must be checked. If the bit is already set to ‘1’, it means that the incoming Interrupt request is invalid and must not be forwarded or processed. Otherwise, if the bit is ‘0’, it is switched to ‘1’ and the correspondent Interrupt request is considered to be valid for processing in the node and forwarding to all the output ports of the switch. On the contrary, an incoming Interrupt acknowledgment code is assumed to be valid if the correspondent ISR bit is ‘1’ and invalid otherwise.

IV. Operation Modes

Distribution of an Interrupt codes with a particular Interrupt identifier may be organized in one of two modes: either in the *Acknowledged Mode* or in the *Unacknowledged Mode*. If an Interrupt handler has accepted an Interrupt request for processing which is distributed in the Acknowledged Mode, this handler must generate and send the correspondent Interrupt acknowledgment. If the Interrupt request is distributed in the Unacknowledged Mode, the handler must not send the Interrupt acknowledgment.

Both operation modes which can be used concurrently in a network are discussed in the following subsections in more details.

A. Acknowledged mode

The key advantages of the Acknowledged Mode is that an Interrupt source gets a confirmation that the issued Interrupt request has successfully propagated over the network and reached at least one Interrupt handler which is assigned to process such Interrupt requests. If either the Interrupt request or the Interrupt acknowledgment has been lost, the Interrupt source is informed about it by the error recovery mechanism. Also, propagation of the Interrupt acknowledgment clears correspondent ISRs’ bits in the network switches and nodes on its rout allowing distribution of next Interrupt request with the same Interrupt identifier.

The Acknowledged Mode imposes a set of constraints in the Interrupt codes distribution procedure. An Interrupt source must not send out next Interrupt request until the expiration of a special time interval, *Tg*, that is started at the reception of the Interrupt acknowledgment with the same Interrupt identifier. Similarly, an Interrupt handler must not send an Interrupt acknowledgment until the expiration of a time interval *Th* that is started at the reception Interrupt request with the correspondent Interrupt identifier. Both requirements are aimed to ensure that the Interrupt request and Interrupt acknowledgment with the same Interrupt identifier will not collide in the network. Finally, in the Acknowledged mode in a network there must be not more than one Interrupt source for Interrupt requests with a particular identifier.

In the Acknowledged Mode distribution of Interrupt acknowledgments is the primary way to clear the ISRs and prepare the network for transmission of the next Interrupt request. However, either an Interrupt request or the correspondent Interrupt acknowledgment can be lost while propagating over the network. To deal with the problem each bit in the ISR is associated with a reset timer. The reset timer is started when an Interrupt request is received in a switch (or sent from a node) and the correspondent ISR bit is set to ‘1’, and stopped when an Interrupt acknowledgment is received and the correspondent bit is reset to ‘0’. If the correspondent Interrupt acknowledgment has not been received and, consequently, the ISR bit has not been reset to ‘0’, the expiration of the reset timer causes reset of the ISR bit. Therefore, the reset timer mechanism ensures network recovery from either loss of Interrupt request or loss of Interrupt acknowledgment.

As an example of the Acknowledged Mode utilisation considers a use case of a satellite attitude control system orienting to the Sun. Let the system consist of two subsystems: an orientation module and a central computer. Normally, the orientation module monitors whether the satellite orientation is correct and transmits attitude information to the central computer. The central computer accepts the information from the module and commands to take a particular action in response to a change in the attitude. However, in order to ensure reliability, the orientation module can operate in an automatic mode in case of emergency. When the module detects that the attitude has changed into incorrect one, it sends a correspondent Interrupt request to the central computer and waits for the Interrupt acknowledgment. If the acknowledgement has not been received, the module assumes that the central computer does not operate now and enters the automatic mode. In this mode the module is permitted to perform the attitude recovery without the command from the computer.

B. Unacknowledged mode

In the Unacknowledged Mode the reset timers are the only way to clean the network ISRs after the propagation of an Interrupt request. As in the Acknowledged Mode the reset timer is started at the reception of the correspondent valid Interrupt request. However, since in the Unacknowledged Mode the correspondent Interrupt acknowledgment would not be received, the reset timer always expires clearing the ISR bit.

The absence of Interrupt acknowledgments leads to the subsequent differences between the modes. In the Unacknowledged Mode an Interrupt source cannot get the automatic confirmation that the issued Interrupt request has reached an Interrupt handler. Since in the Unacknowledged Mode Interrupt handlers does not send Interrupt acknowledgments, the *Th* time interval is not used. Also, in Interrupt sources the next Interrupt request may be sent immediately after the expiration of the correspondent ISR reset timer without the *Tg* delay.

In the Unacknowledged Mode multiple Interrupt sources may simultaneously issue Interrupt requests with the same Interrupt identifier. However, it should be noted that in the case of several Interrupt sources which simultaneously issue Interrupt requests with the same Interrupt identifier only one of the requests will reach a particular Interrupt handler and it cannot be distinguished which was the source of the particular request. Several Interrupt handlers do not cause problems in this mode.

The Unacknowledged Mode is recommended to be used for applications which either do not need acknowledgments of correct Interrupt request delivery or require considerably high level of performance. For example, it may be a network manager which sends an Interrupt request prohibiting all the nodes (or several nodes of a particular type) from transmission of data packets to clear the network. In this case it is not necessary for nodes to response with the Interrupt acknowledgment on the reception of the command, because only one acknowledgement will reach the manager. Thus, the manager will not be informed whether the command has reached all the target nodes.

V. Reliability

To increase the reliability of Interrupt codes delivery the mechanism offers several approaches for error tolerance and recovery. The main tools to implement reliability are broadcast distribution of Interrupt codes, reset timers and ISR change timers.

In networks with hardware redundancy broadcast distribution guarantees that a loss of an Interrupt code in a link will not stop the propagation of the code over the network.

Reset timers recover the network for the distribution of subsequent Interrupt codes. While in Acknowledged and Unacknowledged Modes reset timers are dedicated for different purposes, in both modes reset timers allow recovery after a loss of an Interrupt code.

ISRs and ISR change timers can be used to ensure a protection from occurrence of unexpected Interrupt codes. There are two primarily sources of unexpected Interrupt codes. Firstly, an unexpected Interrupt code can be caused by network malfunction producing a false Interrupt code (e.g by bits inversion in a link due to a noise). Secondly, an unexpected Interrupt code can be issued by an incorrectly operating node, a “babbling idiot” node. In both cases an unexpected Interrupt code occurrence can lead to either an Interrupt handler will receive and process a false Interrupt request, or infinite looping Interrupt code could occur in a network with circular connection. Particularly, the last case can happen if a false Interrupt request appears when the correspondent Interrupt acknowledgment is propagating over the network and vice versa.

ISRs are implicitly used to stop propagation of unexpected Interrupt codes. When a received unexpected Interrupt code dose not correlate with the value of the correspondent ISR, this Interrupt code is ignored and not transmitted further. However, this mechanism cannot terminate the propagation of Interrupts which do correlate with the ISR bit state.

ISR change timers are dedicated to overcome the problem in the Acknowledged Mode. An ISR change timer defines the minimum allowed time between any two consecutive changes of the ISR bit. This means that the ISR bit value may not be changed before the correspondent ISR timer expired. In case of an attempt to change the state of an ISR bit while the correspondent timer has not expired yet, the bit value should not be changed and the received Interrupt code should be ignored. Thus, the ISR change timers mechanism partially protect the network from distribution of unexpected Interrupt requests when the ISR bit is ‘0’ and from the distribution of unexpected Interrupt acknowledgments when the ISR bit is ‘1’ (in the Acknowledged Mode).

VI. Time Issues

Calculation of the timeout values is important to ensure the correctness of the operation of the Distributed Interrupt mechanism. This section discuss some calculation rules that should be taken into account while implementing SpaceWire devices and configuring the network.

A. Estimation of the Interrupt code propagation time

Before discussing timeouts calculation principles it is necessary to define the worst case propagation time of an Interrupt code over the longest path in the network. The value of the worst case propagation time ($T_{IP\max}$) should be calculated as the worst case queuing time plus the transmission time of Interrupt code over all the switches and links in the longest path:

$$T_{IP\max} = L_{Queue} \cdot T_{CC} + (P_{Len} - 1) \cdot T_{wtc} + P_{Len} \cdot T_{CC}, \tag{1}$$

where L_{Queue} is the worst case queue length, P_{Len} is the number of links in the longest path, T_{wtc} is the switch propagation time and T_{CC} is the transmission time of an Interrupt code over one link. The queue length should be defined as the maximum number of Interrupt identifiers, actually used in the network, i.e. L_{Queue} has the maximum value of 32. The transmission time of an Interrupt code over one link may vary from implementation to implementation but typically stands between 0.3 and 1.5 µs depending on the link bit rate.

The estimation of the worst case queuing time is based on the fact that an Interrupt code propagating through a network can be delayed by any other Interrupt code only once. After this delay happened for the first time both Interrupt codes will be sent in sequence to the next switch. Therefore, any Interrupt code may be delayed by $L_{Queue} - 1$ Interrupt codes. Also, as it is assumed that Time-codes are not expected to be sent often, the worst case propagation time estimation takes into account that an Interrupt code may be delayed by no more that one Time-code.

The correctness of Eq. 1 can be proved by means of simulation. Firstly, consider a network with mesh topology and given that L_{Queue} has the maximum possible value of 32. The resulted value of the worst case propagation time estimated in accordance with Eq. 1 is depicted in Fig. 1 as $T_{\max}$. It is noticeable that as the value of the T_{CC} transmission time depends on the link bit rate, the worst case propagation time is a function of the bit rate as well. Then, we analysed the worst case propagation time in the same network through simulation, when all 32 types of Interrupt codes are transmitted with the maximum possible rate. The worst case propagation time yielded by this simulation is given at the Fig. 1 as $T'_{\max}$. As a result, at any transmission rate the theoretically estimated value of the worst case propagation time does not contradict the practically calculated value. Moreover, both results are relatively close to each other thus proving the precision of the proposed estimation approach.

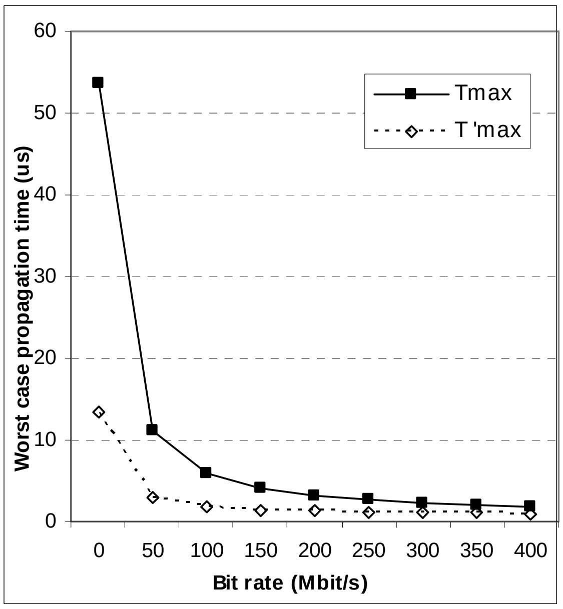


Fig. 1. The worst case propagation time: theoretical and practical calculation results

The worst case propagation time can be decreased at the expense of generality. While the specification of the Distributed Interrupt mechanism does not define the arbitration scheme for Interrupt code of the same type (i.e. among different Interrupt requests), implementations are permitted to deploy any appropriate arbitration principle. Thus, a solution to decrease the worst case propagation time of particular Interrupt codes is to use the priority-based arbitration approach when targeted Interrupt identifiers are assigned to the highest priority levels. As a result, the worst case queue length for such high priority level Interrupt codes would be considerably less than the maximum number of Interrupt identifiers in the network. In turn, this leads to relatively low worst case propagation time for these Interrupt codes.

B. Calculation of the Tg minimum time interval

In the Acknowledged Mode the *Tg* minimum time interval delays the transmission of an Interrupt request for a period of time that is required for the network to become ready to propagate this Interrupt request. The network is not ready to accept an Interrupt request if the Interrupt acknowledgment on the previous Interrupt request is still propagating over the network in such a way that a collision between these Interrupt request and Interrupt acknowledgment is possible. Thus, the duration of the *Tg* minimum time interval depends basically on the network topology and the number of handlers. Below Three different cases of the interval calculation are discussed.

Consider a network with only one handler for an Interrupt identifier and no cycles in the topology. When the Interrupt source receives an Interrupt acknowledgment on the issued Interrupt request, it means that this Interrupt acknowledgment has been propagated via the only path between the source and the handler. Although it is possible that the propagation of the received Interrupt acknowledgment over the whole network has not been completed yet, the next Interrupt request will not collide with this acknowledgment on its way to the handler. Thus, in this case the next Interrupt request may be issued as soon as possible, i.e. the *Tg* minimum time interval may be zeroed. Also, it is noticeable that a collision between an Interrupt request and the correspondent Interrupt acknowledgment can occur somewhere in the network, but this will not affect the operation of the mechanism.

An example of the discussed case is shown at Fig. 2. In the part A of the figure an Interrupt acknowledgment is distributed over the network and reaches the source. Then, in the part B the source issues the next Interrupt request when the Interrupt acknowledgment has not been propagated over the whole network due to some reasons. While this Interrupt request reaches the handler without problems, it catches up the Interrupt acknowledgment at its another propagating path in the Switch N-1. As a result, the distribution of both the Interrupt request and the Interrupt acknowledgment is terminated. However, this situation cannot affect the distribution of the next Interrupt requests and Interrupt acknowledgments of this particular type as the Switch N-1 does not belong to the only path between the source and the handler.

In the second case consider a network with only one handler for an Interrupts with a particular identifier and cycles in the topology. When the source of Interrupts with the Interrupt identifier receives an Interrupt acknowledgment on the issued Interrupt request, it does not guarantee that the Interrupt acknowledgment has been propagated already over all the possible paths between the source and the handler. Therefore, the source must delay transmission of the next Interrupt request for the worst propagation time of the Interrupt acknowledgment via the longest cycle in the network. While it ensures that the next Interrupt request will not collide with the Interrupt acknowledgment on its path to the handler, the time interval does not protect the network from a collision in some part of the network that does not belong to the path between the Interrupt source and Interrupt handler, so it isn't critical for this Interrupt correct distribution (as was discussed earlier for the first case).

The lower bound of the *Tg* minimum time interval defined as the worst propagation time of an Interrupt acknowledgment via the longest cycle in the network may be decreased at the expense of generality. For example, if there are cycles in a network, but a particular source and handler are connected by only one path (i.e. no cycles between them), the *Tg* minimum time interval in the source may be zeroed as in the first case discussed above. However, if another source and handler are connected by several paths (i.e. there are cycles between them), the correspondent minimum time interval needs to be bounded by the worst propagation time of an Interrupt acknowledgment via the longest path. Such approach increases complexity of Distributed Interrupts configuration in a network, but improves its operation time indexes.

Also, the lower bound of the *Tg* minimum time interval may be decreased at the expense of reliability. Consider a network with mesh topology, with the mesh size $N \times N$, where N is a relatively high number. In such a network the length of the longest cycle is $4N$. However, in the absence of errors it is impossible that an Interrupt code can propagate over the whole longest cycle because it will propagate over shortest cycles firstly. Thus, the actual value of the *Tg* minimum time interval may be calculated on the basis of a cycle shorter than $4N$. However, it could decrease reliability of the mechanism: if several faults stop the propagation of an Interrupt code over all the shortest cycles, the mechanism will not be able to handle the situation correctly.

The last case for the calculation of the *Tg* minimum time interval is a network with several Interrupt handlers for one Interrupt identifier. Regardless of the network topology, in such case it is recommended that the duration of the interval depends on the worst propagation time of an Interrupt code over the longest path in the network and should be calculated as:

$$Tg = k \cdot T_{IP\max}, \qquad (2)$$

where k is a reliability coefficient which should be equal or more than 1. The coefficient k is introduced to tolerate a situation when the distribution of an Interrupt acknowledgments exceed the worst Interrupt code propagation time due to unexpected errors. The recommended value of the coefficient is 1.2.

Equation 2 introduces the worst case value for the *Tg* minimum time interval and may be used in any case, to simplify the network calculation and configuration process.

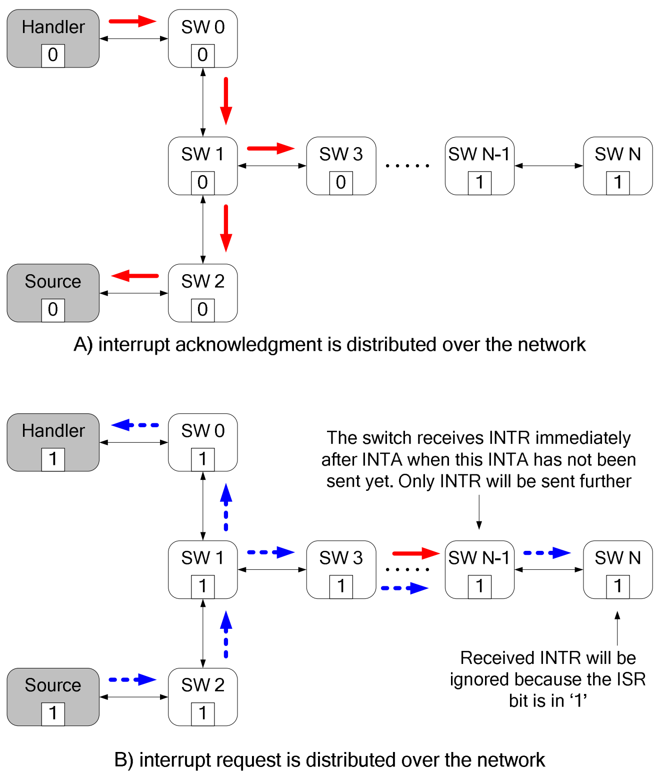

Handler
0
SW 0
0
SW 1
0
SW 3
0
SW N-1
1
SW N
1
Source
0
SW 2
0
A) interrupt acknowledgment is distributed over the network
Handler
1
SW 0
1
The switch receives INTR immediately after INTA when this INTA has not been sent yet. Only INTR will be sent further
SW 1
1
SW 3
1
SW N-1
1
SW N
1
Received INTR will be ignored because the ISR bit is in '1'
Source
1
SW 2
1
B) interrupt request is distributed over the network

Fig. 2. Distribution of Interrupt signals in a network with one handler and no cycles

C. Calculation of the Th time interval

The *Th* time interval delays transmission of an Interrupt acknowledgment for a period of time, which is needed to ensure that the handler will not receive the acknowledged Interrupt request once more. It can happen if the acknowledged Interrupt request is still propagating over a part of the network from where it can reach the handler again, i.e. over a cycle. Three different cases of the *Th* interval calculation are discussed below.

The first and second cases are appropriate when there is only one handler for an Interrupt source with some identifier or there could be several handlers, but it is permitted that Interrupt requests are transmitted not to all Interrupt handlers. As the first case, consider a network with no cycles in the topology. When a handler receives an Interrupt request which this handler is responsible to process, it means that this Interrupt request has already propagated via the only path between the source and this handler. Though it is possible that the distribution of the Interrupt request over the whole network has not been completed yet, the Interrupt acknowledgment will not collide with the Interrupt request on its way to the source. Thus, in this case the Interrupt acknowledgment may be issued as soon as possible, i.e. the *Th* time interval may be zeroed. Again, as in the case of the *Tg* minimum time interval, zeroing the *Th* time interval makes possible a collision between the Interrupt request and the Interrupt acknowledgment somewhere in the network, but not in the path between the source and the handler. As it is discussed above, such collision would not affect correct delivery of the Interrupt codes.

As the second case consider a network with cycles in the topology. When the handler receives an Interrupt request, it does not guarantee that the Interrupt request has been propagated over all possible paths between the source and the handler. Therefore, the handler must delay transmission of the correspondent Interrupt acknowledgment for the worst propagation time of the Interrupt request delivery via the longest path in the network. As in the same case for the *Tg*, the minimum time interval value can be decreased at the expense of reliability and/or generality.

The third case is appropriate when there are several Interrupt handlers in the network and it is required that Interrupt requests are delivered to all the handlers. To address the requirement each handler must delay transmission of the Interrupt acknowledgment for the worst propagation time of the Interrupt requests propagation over the whole network regardless of the network topology:

$$Th = k \cdot T_{IP\max}, \tag{3}$$

where *k* is a reliability coefficient which should be equal or more than 1. Again, the recommended value for the coefficient is 1.2.

D. Reset timers

Generally, reset timers are used in both the Acknowledged and Unacknowledged Modes as a tool to clear the ISRs in network switches and nodes and prepare the network for distribution of the consecutive Interrupt codes. However, the modes deploy reset timers in considerable different ways.

As in the Acknowledged Mode the primary way to recover the network after propagation of an Interrupt request is to send the Interrupt acknowledgment, reset timers should be configured so that they are expired only when it is definitely true that the Interrupt request or the Interrupt acknowledgment is lost. For this reason, the timeout for a reset timer in the Acknowledged Mode should be not less than the doubled Interrupt code worst case propagation time plus the *Th* time interval duration:

$$T_{\mathrm{Re}set} = 2 \cdot T_{IP\max} + Th, \tag{4}$$

On the contrary, in the Unacknowledged mode reset timers are the only way to clear ISRs after the propagation of an Interrupt request. Due to the fact that a reset timer definitely expires each time when it is set, the timeouts of reset timers clearly affect the rate at which Interrupt requests of a particular type can be sent to the network. To ensure that any issued Interrupt request will reach an Interrupt handler (in the absence of errors) and will not collide with the previous Interrupt request with the same identifier, the reset timer timeout should be not less than the worst case Interrupt code propagation time, which is calculated in accordance with Eq. 1:

$$T_{\mathrm{Re}set} = T_{IP\max}, \tag{5}$$

In both Acknowledged and Unacknowledged Modes it is highly recommended that the duration of reset timeouts in nodes is not less than the duration of reset timeouts in intermediate switches.

E. ISR change timer

ISR change timers are intended to protect the networks with circular connections from infinite looping caused by occurrence of unexpected Interrupt codes in the Acknowledged Mode. The duration of ISR change timeout must not exceed the minimum of the *Tg* and *Th* time intervals. Otherwise, the ISR change timer mechanism would corrupt the distribution of Interrupt codes.

VII. CONCLUSION

The SpaceWire Distributed Interrupt mechanism described in this paper is intended for transmission of hard real-time signals over SpaceWire networks.

The core principle of the Distributed Interrupt mechanism operation consists in broadcast distribution of low size Interrupt signals (14 bits), which have a high priority level among SpaceWire characters and control codes. The broadcast distribution allows simple configuration without specific routing tables in switches, and high level of fault tolerance in networks with hardware redundancy. The low size and high priority of Interrupt codes ensure ultra-low propagation time, some microseconds, in SpaceWire networks of any practical size.

The mechanism offers two distinctive operation modes targeted at different applications. In the Acknowledged Mode an Interrupt handler that has received an Interrupt request generates correspondent Interrupt acknowledgment and the Interrupt source will be informed that the issued request has reached at least one handler node and has been accepted for processing. The Unacknowledged Mode does not provide the service of automatic acknowledgments, but takes advantage in higher transmission rate of Interrupt request and smaller configuration efforts.

To increase the reliability of signal delivery the mechanism offers several approaches for error protection and recovery. Broadcast distribution guarantees that a loss of an Interrupt signal will not stop the propagation of the signal in networks with redundant paths. Reset timers recover the network for the distribution of subsequent Interrupt signals.

Calculation of the Distributed Interrupt mechanism timeouts is a critical task to ensure the proper operation of the mechanism. While almost all the timeouts are functions of the Interrupt code worst case propagation time, this parameter should be estimated as close to its realistic value as possible. The computation of the guards times *Tg* and *Th* for sources and handlers respectively can be made by taking into account the network topology and other specific factors. It leads to higher performance level, but has relatively high complexity of calculation and configuration. Another approach - one-size-fits-all calculations, has lower configuration complexity, but could limit the timing characteristics of the Distributed Interrupts operation in a network, first of all – duty cycle of Interrupt requests.

It can be concluded that the Distributed Interrupt mechanism satisfies the basic latency and reliability requirements and allows integrating critical system control traffic into SpaceWire on-board networks. To increase its flexibility the mechanism provides a number of different modes and configuration methods that cover a wide range of use cases. The Distributed Interrupt mechanism has been prototyped and investigated, proved its consistency and efficiency in implementation and operation. The mechanism is implemented in a set of chips “Multiboard” produced by R&D Center ELVEES Company and is under implementation in several independent design companies now.

REFERENCES

[1] Standard ECSS-E-50-12C, “SpaceWire, Links, Nodes, Routers and Networks,” European Cooperation for Space Standardization, July 2008.

[2] S, Parkes, “D2.1 SpaceWire-RT Outline Specification,” SpaceWire-RT Consortium, September 2012.

[3] Y. Sheynin, S. Gorbachev, L. Onishchenko, “Real-time signalling in SpaceWire networks,” International SpaceWire Conference Proceedings, Dundee, pp. 205 – 208, September 2007.

[4] L. Onishchenko, A. Eganyan, I. Lavrovskaya, “Distributed Interrupts mechanism verification and investigation by modelling on SDL and SystemC,” International SpaceWire Conference Proceedings, Nara, pp. 151 – 154, November 2008.

Running Disparity Management for DC-Balancing a 10-bit Code Set

SpaceWire Standardisation, Long Paper

Clifford Kimmery

Honeywell International Space Electronic Systems
Clearwater, FL, USA
clifford.kimmery@honeywell.com

***Abstract*— Providing bandwidth efficiency similar to standard SpaceWire while using Direct Current (DC)-balanced Data-Strobe encoding requires use of 10-bit codes that match the size of SpaceWire characters. Because of the limited quantity of 10-bit codes that meet the needs of DC-balanced Data-Strobe encoding, appropriate running disparity management is critical for achieving the one-zero ratio necessary for successful Alternating Current (AC)-coupled operation. The running disparity management is complicated by the need to balance both the Data and Strobe signals simultaneously.**

The primary goal of the running disparity management is to minimize the running disparity of each signal without minimizing one at the expense of the other. The characteristics and limitations of the available 10-bit codes significantly affect the complexity of the methods considered. Two methods have been shown to consistently keep the running disparity within bounds.

One method (Dynamic Priority) attempts to minimize the running disparity of the signal (Data or Strobe) with the largest running disparity magnitude (the priority signal). If the running disparity magnitude of the priority signal cannot be reduced, the Dynamic Priority method attempts to minimize the running disparity of the non-priority signal. A second method (Minimum Sum of Magnitudes) attempts to minimize the running disparity of both signals simultaneously by choosing the Data-Strobe code pair that produces the smallest sum of running disparity magnitudes. If the sums of magnitudes for both code pairs are equal, the method chooses the code pair that minimizes the running disparity of the signal with the largest running disparity magnitude.

This paper discusses the characteristics and limitations of the available 10-bit codes and describes candidate running disparity management methods. Simulation results for the candidate methods are also presented.

***Index Terms*—SpaceWire, Data-Strobe, DC-balance, disparity, encoding, decoding.**

I. DC-Balanced Data-Strobe Encoding

Direct Current (DC)-balanced Data-Strobe encoding [1] is a practical character-level encoding method for applications requiring galvanic isolation between link endpoints. It offers an alternative to standard SpaceWire character encoding that supports galvanic isolation using ANSI/TIA/EIA - 644 LVDS devices and conventional Alternating Current (AC)-coupling circuits.

The terminology used in this paper is derived from that defined by the SpaceWire standard [2]. The term character is defined by the SpaceWire standard and includes data characters and control characters. The term code is defined as a binary value used to represent a character when transmitted on the SpaceWire link. In standard SpaceWire, a character and the corresponding code are identical. DC-balanced encoding represents each character with one or more different code values.

DC-balanced Data-Strobe encoding uses a class of codes that simultaneously DC-balance both the Data and Strobe bit streams. The code size used directly establishes the resulting link overhead. Link overhead is the ratio of the code size to the datum size (standard SpaceWire has a link overhead ratio of 10/8).

The code size also indirectly affects two characteristics of the encoded bit stream: the maximum running disparity and the maximum run length. The running disparity is the difference in the number of ones and zeroes measured on a continuous basis and is a relative indicator of the DC-balance of the signal. The maximum run length is the largest number of ones or zeroes in a row and establishes the lowest frequency of the encoded bit stream (the highest frequency of the encoded bit stream is always one-half the baud rate). Standard SpaceWire data characters have a maximum run length of nine bits.

A smaller code size generally offers fewer values that meet the needs of DC-balanced Data-Strobe encoding. As a result, it is difficult to select values with the small disparity values needed to limit the running disparity and the shorter run lengths needed to minimize frequency bandwidth.

Of the available DC-balanced codes, the 10-bit code offers the smallest link overhead in comparison to standard SpaceWire (within 5%) [1]. The small size of the 10-bit code set severely restricts the ability to optimize the disparity and run-length characteristics. Because of the limited number of

10-bit code values, the only opportunity available for optimizing the bit streams is to manage the running disparity.

The 10-bit DC-balanced code set contains codes with zero disparity (an even number of ones and zeros), positive disparity (more ones than zeros) and negative disparity (more zeros than ones). Unfortunately, there are no 10-bit DC-balanced codes that produce zero disparity on both the Data and Strobe signals simultaneously. As a result, each SpaceWire character must be assigned two 10-bit codes with opposite disparity so that the running disparity can be maintained close to zero. Establishing the two 10-bit codes with opposite disparity is a trivial exercise since the bitwise inverse of any binary value has opposite disparity. The term base code is defined as the binary value directly mapped to a SpaceWire character. The inverted base code is the code with opposite disparity mapped to the same SpaceWire character.

II. Significant Properties of DC-Balanced Codes

The SpaceWire Strobe encoding function uses the Data code bit stream to produce the corresponding Strobe code bit stream by Exclusive-OR with an alternating-one-zero pattern (clock). This behavior is inherent in Data-Strobe encoding and is unchanged when the Data code bit stream is composed of DC-balanced codes.

A 10-bit DC-balanced code value assigned to a SpaceWire character (the base code) is paired with an alternate code value (the inverted base code) to manage running disparity. Any code value can be inverted (ones-complement) to produce a corresponding code value with the opposite disparity characteristic. The Data-Strobe encoding function can also be viewed as pairing each Data code value with an alternate Strobe code value.

Since the Exclusive-OR and Inversion operators are associative, there is a defined relationship between the 10-bit code values used to represent a specific SpaceWire character in the Data and Strobe cases. Figure 1 exhibits these relationships.

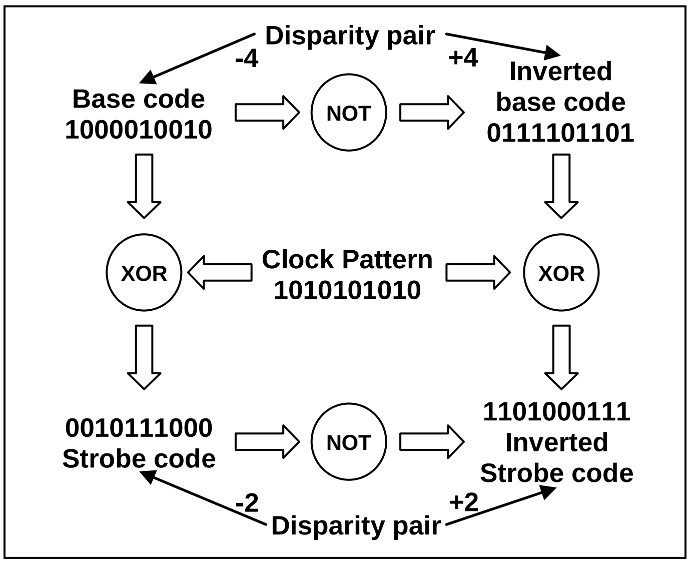


Fig. 1. Code Pair Relationship Example

These properties allow a one-to-one assignment of each SpaceWire character with a single base-code value, simplifying the character encoding implementation. The opposite-disparity encoding of the base-code value is easily generated by inverting the base-code value. The base-code value (or its inverse) is transmitted on the Data signal and the conventional Strobe generation mechanism automatically produces the appropriate code for transmission on the Strobe signal.

III. Running Disparity Management

As with 8b10b encoding, DC-balanced Data-Strobe encoding manages running disparity to limit the difference in the number of ones and zeroes in successive codes. Unlike 8b10b encoding, the Data-Strobe running disparity must be managed for both SpaceWire signals (Data and Strobe) simultaneously [1]. The goal is to minimize the running disparity of each signal without minimizing one at the expense of the other.

Several running disparity management methods were considered. While other methods are possible, the Dynamic Priority and the Minimum Sum of Magnitudes methods were chosen for extensive evaluation.

A. Dynamic Priority Method

The Dynamic Priority method attempts to minimize the running disparity of the signal (Data or Strobe) with the largest running disparity magnitude (the priority signal). The running disparity represents the distance of the signal DC-balance from zero and may be positive or negative. By using the magnitude of the running disparity, the method can compare the distance without the complication of adjusting for signed values. If the running disparity magnitude of the priority signal cannot be reduced, the method attempts to minimize the running disparity of the non-priority signal. The set of rules for minimizing the running disparity of both signals with one of the signals having priority is shown in Table I.

B. Minimum Set of Magnitudes Method

The Minimum Sum of Magnitudes method attempts to minimize the running disparity of both signals simultaneously by choosing the Data-Strobe code pair that produces the smallest sum of running disparity magnitudes. If the sums of magnitudes for both disparity code pairs are equal, the method chooses the code pair that minimizes the running disparity of the signal with the largest running disparity magnitude. The terms used in defining the rules for minimizing the running disparity of both signals simultaneously are shown in Table II. The set of rules is shown in Table III.

TABLE I. RULES FOR DYNAMIC PRIORITY METHOD

Precedence	**Rule Condition**	**Rule Conclusion**
First	If the base code disparity makes the priority signal running disparity magnitude greater	Use the inverted base code
Second	If the base code disparity makes the priority signal running disparity magnitude smaller	Use the base code
Third	If the base code disparity makes the non-priority signal running disparity magnitude greater	Use the inverted base code
Fourth	Otherwise	Use the base code

TABLE II. DEFINITIONS OF MINIMUM SUM OF MAGNITUDES TERMS

Term	**Definition**
$RDD_{current}$	The current Data running disparity value
$RDS_{current}$	The current Strobe running disparity value
RDD_{base}	Computed Data running disparity value (base code)
RDS_{base}	Computed Strobe running disparity value (base code)
$RDD_{inverted}$	Computed Data running disparity value (inverted base code)
$RDS_{inverted}$	Computed Strobe running disparity value (inverted base code)
$SumMag_{base}$	$SumMag_{base} = \lvert RDD_{base}\rvert + \lvert RDS_{base}\rvert$
$SumMag_{inverted}$	$SumMag_{inverted} = \lvert RDD_{inverted}\rvert + \lvert RDS_{inverted}\rvert$

TABLE III. RULES FOR MINIMUM SUM OF MAGNITUDES METHOD

Precedence	**Rule Condition**	**Rule Conclusion**
First	If $SumMag_{base}$ is greater than $SumMag_{inverted}$	Use the inverted base code
Second	If $SumMag_{base}$ is equal to $SumMag_{inverted}$, and If [$\lvert RDD_{current}\rvert > \lvert RDS_{current}\rvert$ and $\lvert RDD_{base}\rvert > \lvert RDD_{inverted}\rvert$] or [$\lvert RDD_{current}\rvert < \lvert RDS_{current}\rvert$ and $\lvert RDS_{base}\rvert > \lvert RDS_{inverted}\rvert$]	Use the inverted base code
Third	Otherwise	Use the base code

The running disparity management methods were evaluated by implementing model encoders of each. A 10 million character random data set was used to build a histogram of the running disparity characteristics of each model. The Dynamic Priority method allowed running disparity excursions of ±12 for the test data set as shown in Fig. 2. Figure 3 shows that the Minimum Sum of Magnitudes method is able to maintain the running disparity within ±8 for the same test data set. Note that the running disparity varies in increments of two because a change in the state of a single bit causes the difference between the number of one bits and the number of zero bits to change by two.

Another attribute of running disparity that is important to assessing the management methods is the dwell time at each running disparity value. The histograms in Fig. 4 and Fig. 5 show that the Dynamic Priority method clustered the dwell time around zero with no dwell time greater than fifteen characters. The histograms in Fig. 6 and Fig. 7 show that the Minimum Sum of Magnitudes method performed similarly with no dwell time greater than fourteen characters.

The Minimum Sum of Magnitudes method clearly provides the best performance for the test data set used. Since a random data set is unlikely to represent real-world SpaceWire packet statistics, the possibility exists of a pathological data set that produces a different result.

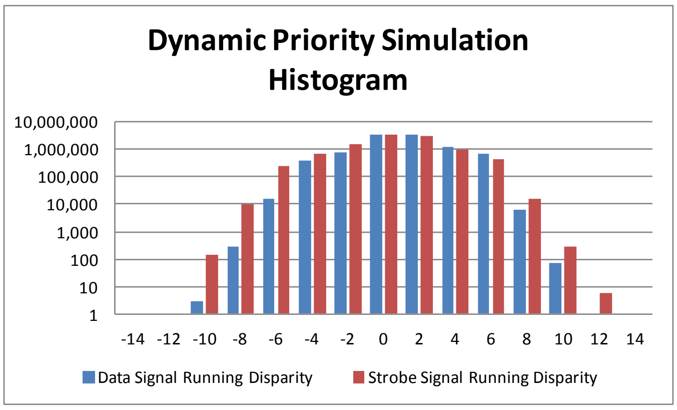


Fig. 2. Dynamic Priority Excursion Histogram

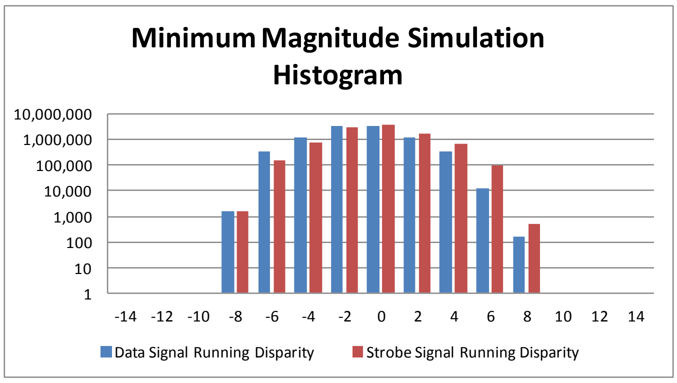


Fig. 3. Minimum Sum of Magnitudes Excursion Histogram

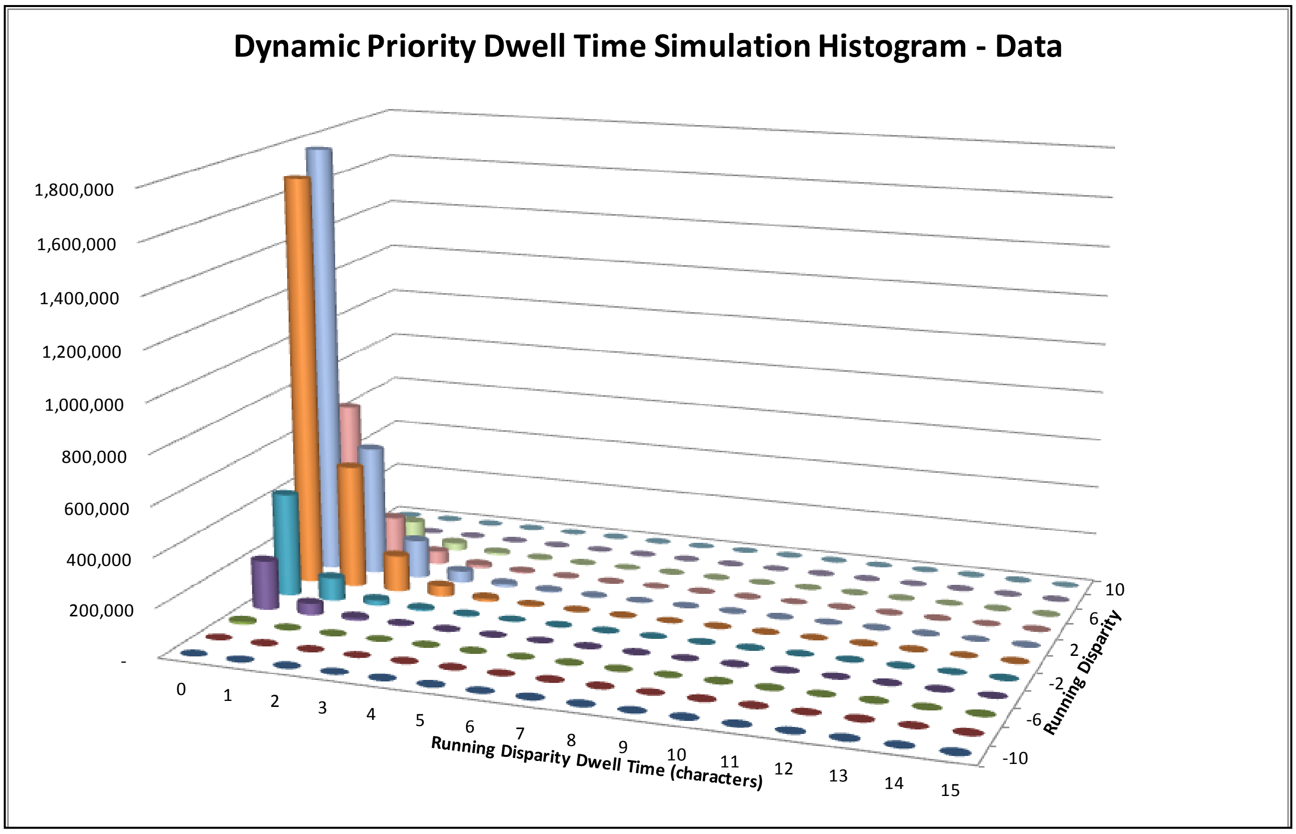


Fig. 4. Dynamic Priority Data Dwell Time Simulation Histogram

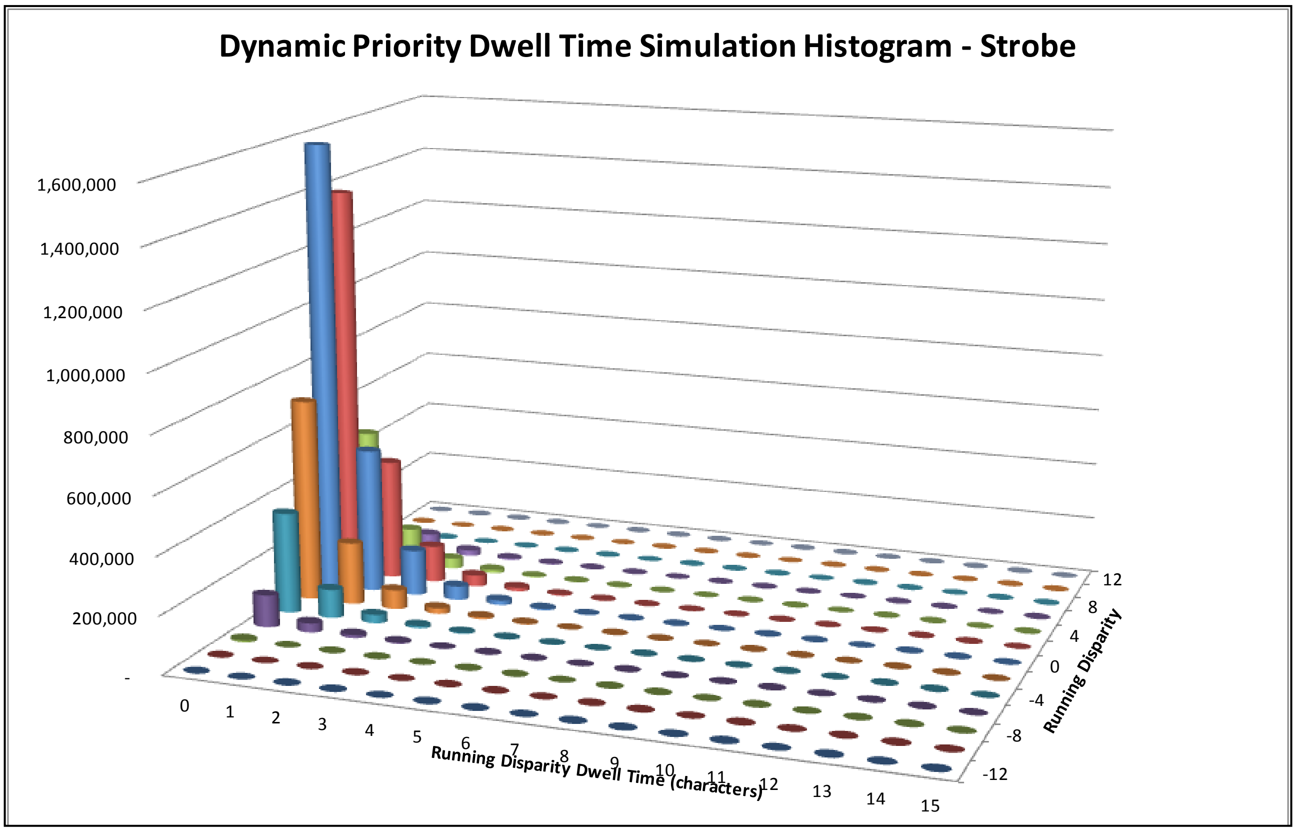


Fig. 5. Dynamic Priority Strobe Dwell Time Simulation Histogram

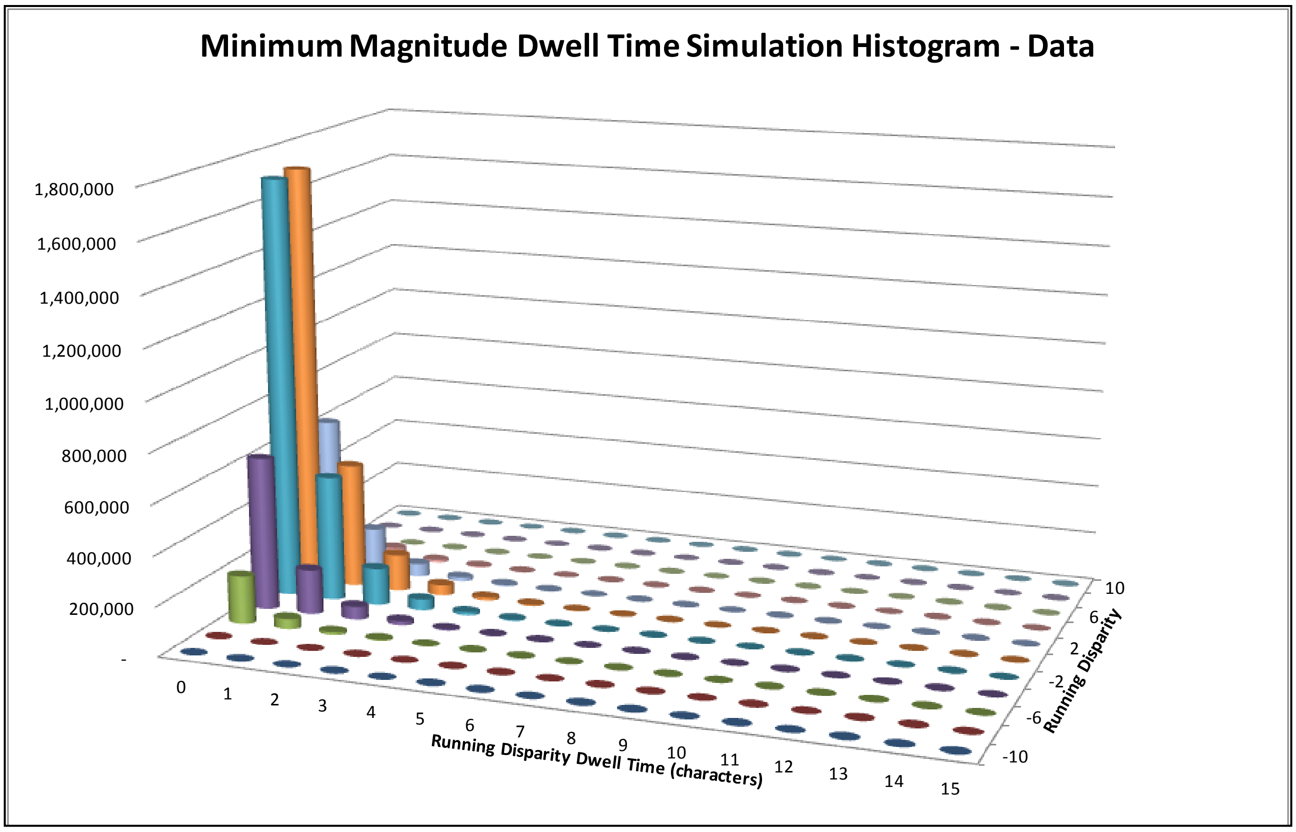


Fig. 6. Sum of Magnitudes Data Dwell Time Simulation Histogram

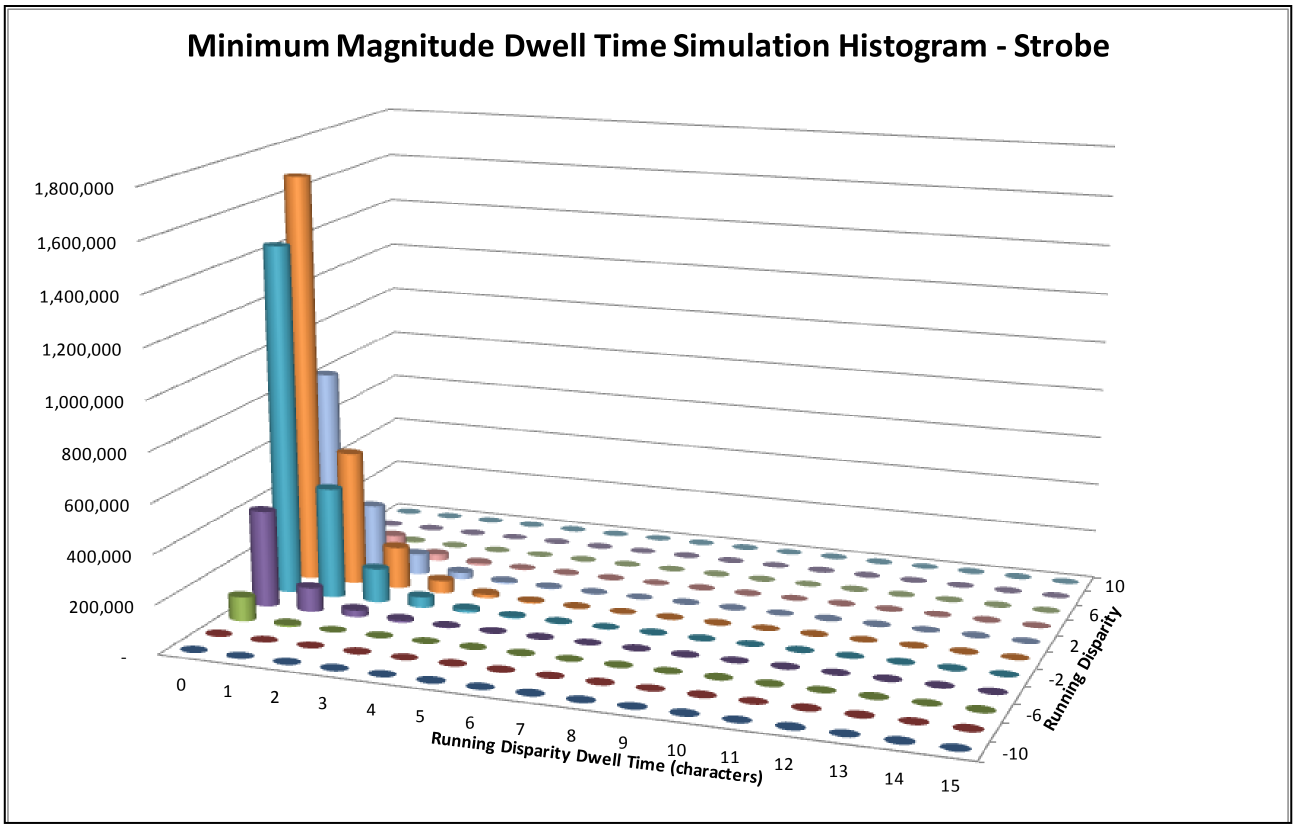


Fig. 7. Sum of Magnitudes Strobe Dwell Time Simulation Histogram

I. Implementation Concepts

Simple concepts for implementing DC-balanced encoders and decoders are shown in Fig. 8 and Fig. 9. The model encoder used to evaluate the running disparity management methods use the encoder concept shown. Responsibility for the running disparity tracking is in the encoder, increasing its complexity. Because of the limited number of 10-bit codes available for DC-balanced encoding, development of an algorithmic method for translating the protocol characters into equivalent base codes is unlikely. The encoding lookup table shown in the Fig. 8 provides for arbitrary mapping of characters to base codes in a straightforward manner.

The decoder lookup table requires more entries because it must decode multiple codes for each protocol character as well as detect illegal code values. Appropriate definition of the mapping between protocol characters and base codes can reduce the number of decoding table entries required.

Note that the dimensions of the encoding and decoding tables are a major factor in establishing the gate count of an implementation. In addition, the complexity of the disparity tracking function can significantly affect the size of the encoder.

II. Summary

The feasibility of using a 10-bit DC-balanced code as an alternative SpaceWire character-level encoding method has been evaluated. The results of that evaluation show that by using an appropriate running disparity management method, the Data and Strobe signals can be balanced within a bounded range.

This assessment of the 10-bit DC-balanced code is based on digital modeling and simulation to determine the performance characteristics. Establishing more complete performance characteristics will require analog modeling and simulation as well as testing of a laboratory implementation.

References

[1] C. Kimmery, “DC-balanced character encoding for SpaceWire,” Proceedings of the 4th International SpaceWire Conference, November 2011, San Antonio, Texas, pp. 269-278.

[2] ECSS Standard ECSS-E-50-12C, “SpaceWire, Links, Nodes, Routers and Networks”, European Cooperation for Space Data Standardization, July 2008

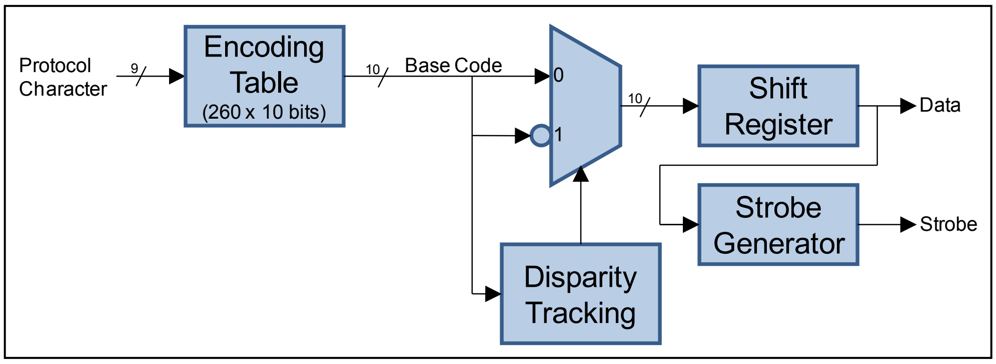


Fig. 8. Conceptual Encoder Diagram

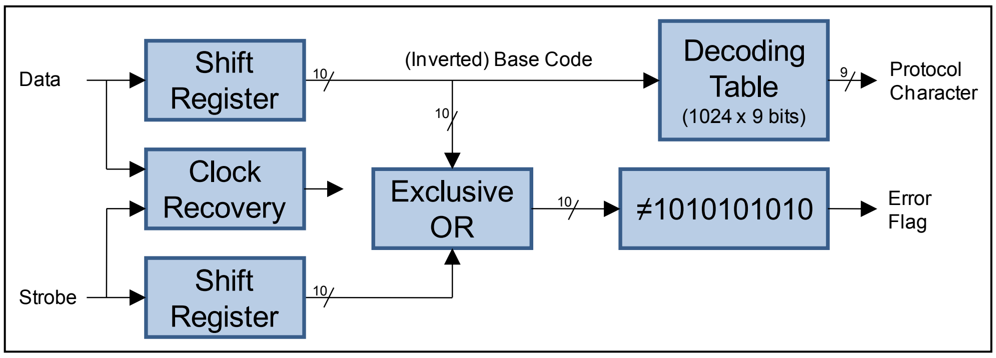


Fig. 9. Conceptual Decoder Diagram

Test & Verification (Long)

Determining the Behaviour of Black-Box SpaceWire Components

SpaceWire test and verification session, Long Paper

Roger Peel, Paul Walker, Barry Cook
4Links Limited
Bletchley Park, United Kingdom
roger@4links.co.uk

David Jameux
On-Board Data Systems Division (TEC-ED)
ESTEC, European Space Agency
Noordwijk, The Netherlands

***Abstract*—This paper outlines how we investigated a rare behaviour that was seen every few hours in the engineering model of a spacecraft data network, which caused a SpaceWire packet to be truncated. This event occurred seemingly at random, a few times per day. The cause turned out to be timing-related, and we identified an interval of a few hundred nanoseconds in which the system was vulnerable. A non-standard feature of the 4Links Multi-link SpaceWire Recorder (MSR) and the standard features of a 4Links Diagnostic SpaceWire Interface (DSI) were used to determine these timings and to reproduce some of them experimentally.**

If one only has access to the links in a SpaceWire network, (i.e. black-box testing), one has to utilise capable test equipment and devise a sensitive investigation strategy to maximise the knowledge that can be discovered from monitoring the network in depth.

We initially built a pipeline of two 10X routers, connected by SpaceWire, which we fed from a DSI port that generated test traffic. A second DSI port was used as the data sink at the other end of the pipeline. We exercised this pipeline using carefully-crafted SpaceWire packets and monitored all of the links using an experimental feature of the 4Links MSR. The recordings allowed us to determine the buffering characteristics of the routers. We also enabled timeouts in the routers, and observed the consequences when the router outputs were stalled for longer than the timeout period.

This paper describes the novel test equipment feature, the techniques used to explore the behaviour of the 10X routers in a laboratory setting, and the precise circumstances of the packet spillage in a spacecraft's data network.

***Index Terms*— SpaceWire, 4Links test equipment, ESA 10X router, black-box testing.**

I. Introduction

When testing a SpaceWire [1] communication network, one might have complete knowledge of the behaviour of all of the components involved, or one might only have access to the external connections of these devices - the SpaceWire links. In the latter case - black-box testing - one has to devise a testing strategy that maximises the knowledge that can be discovered from the external observations.

In the experiments reported in this paper, we determined the buffer sizes at the input and output of the ESA 10X router [3], as well as its behaviour when it spills packets after a timeout.

In addition, we observed timeouts on an ESA 10X router in the engineering model of an ESA spacecraft, and discovered a situation where packet loss occurred.

II. Stand-alone Measurements

We first explored the behaviour of the 10X routers in a laboratory setting. We set up a pipeline from a 4Links Diagnostic SpaceWire Interface (DSI), acting as a traffic generator, through two 10X routers, to another port on the DSI that acted as the traffic sink. Each of the SpaceWire links between these devices was passed through a 4Links Multi-link SpaceWire Recorder (MSR), so that all of the link traffic could be recorded and, more importantly, time-tagged so that the precise time of transmission of each character on the links was known.

The hardware configuration is shown in figure 1.

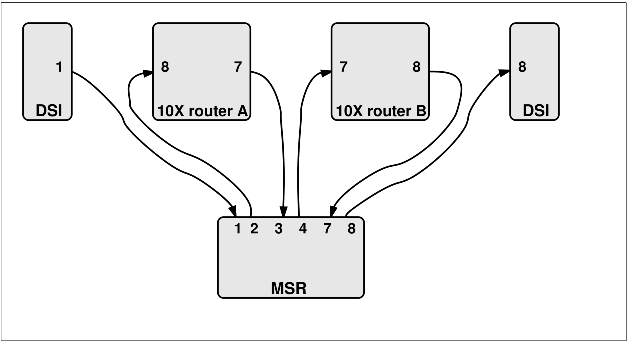


Fig.1. A pipeline through two 10X routers

The 4Links Diagnostic SpaceWire Interface provides the facility to vary its Flow Control Token (FCT) generation algorithm away from the behaviour specified in the SpaceWire standard. Normally, this is done to investigate the flow-control behaviour of a SpaceWire device-under-test. In this instance, the mechanism was used by the receiving DSI to temporarily stop the advertisement of any flow-control credit to the 10X routers. Data characters sent from the DSI into the pipelined routers were therefore eventually blocked as the routers' input and output buffers filled up. The receiving DSI could then be instructed to send individual FCTs, which each allowed a group of eight characters to flow along the pipeline. By recording the precise sequence of events as these groups passed through the routers, the characteristics of the routers could be derived.

The novel aspect of this work is that an experimental mode has been exploited in the 4Links Multi-link SpaceWire Recorder to record time-tags, with a resolution of better than 2ns, for all tokens on the links being monitored (except NULLs). The standard MSR will record time-tags at the start and end of packets, but it does not provide an insight into the gaps that appear within packets if they are blocked within, say, a router. The paper shows output from the prototype analysis tool that was written to visualise this new data.

Our final experiment in the laboratory was to see what would happen when a router timed-out a link because it had become blocked for too long. We enabled this feature on the link between the second router and the DSI. By varying the transmission time of FCTs from the DSI, a timeout could be provoked in a controlled manner, and its effects could be measured.

III. Laboratory Experiments

A. Hardware Arrangement

The work performed in this section of the paper used the following arrangement of equipment:

- Two 10X routers.
- One 4Links Diagnostic SpaceWire Interface (DSI) to act as a data source and a data sink for the routers;
- One 4Links Multi-link SpaceWire Recorder (MSR) to record traffic on three bidirectional SpaceWire links through the routers. A non-standard MSR recording program was used - msr2.
- A laptop computer, running Ubuntu Linux, was used to control the DSI and the MSR.
- A separate desktop PC was used to issue occasional configuration commands to the 10X routers.

B. Observing the Buffering in the 10X Routers

Inspection of the 10X Router's VHDL source code suggests that it has a 32-byte receive buffer. We set out to confirm this.

The only way to obtain a measure of buffering in a pipeline is to stall the sink end of the pipeline, and count the number of bytes that can be sent into the pipelined buffers while no output is consumed. We were able to do this using the SpWIO feature that disables its automatic generation of FCTs in each DSI receiver. After sending the usual full set of seven FCTs during link initialisation, it is possible to command the DSI (using its **`/f 7.`** parameter) not to send any more FCTs, except under subsequent manual control.

Putting all of this together, and sending a stream of 256 data bytes lets us see what blocks in the routers' buffers:

```
java -jar SpWIO.jar /f 7. /d 1 /u 192.168.0.252

@1 7 8 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101
102 103 104 105 106 107 108 109 110 111 112 113 114
115 116 117 118 119 120 121 122 123 124 125 126 127
128 129 130 131 132 133 134 135 136 137 138 139 140
141 142 143 144 145 146 147 148 149 150 151 152 153
154 155 156 157 158 159 160 161 162 163 164 165 166
167 168 169 170 171 172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 187 188 189 190 191 192
193 194 195 196 197 198 199 200 201 202 203 204 205
206 207 208 209 210 211 212 213 214 215 216 217 218
219 220 221 222 223 224 225 226 227 228 229 230 231
232 233 234 235 236 237 238 239 240 241 242 243 244
245 246 247 248 249 250 251 252 253 254 255 eop

@8 fct fct fct
```

Fig.2. Sending packets and FCTs using the SpWIO program

The results of this stimulus, shown in figure 2, may be seen in the following three plots, which illustrate the same network activity on different scales. Notice that activity on the eight ports of the MSR is plotted on eight horizontal rows. Each non-NULL character is represented with the correct width. The links to the DSI connected to the MSR on ports 1 and 8 (see figure 1) were being run at 10Mb/s, while those connected to the 10X routers were being run at 100Mb/s. The data stream of 256 bytes plus two path address bytes was long enough to fill each router buffer completely, and allowed for more data to propagate along the pipeline when the DSI at the sink was used to manually send FCTs.

Figure 3 shows the initial filling of the pipeline with a data stream that was long enough to block all the way back to the output of the DSI.

56 bytes were able to flow right along the pipeline to the receiving DSI port, from where no FCTs were generated in return. 35 bytes were buffered in Router B, and 35 bytes were buffered in Router A. It is impossible to say how many bytes were stored in the routers' receive buffers and how many were in their transmit buffers - but 32 bytes in the receive buffers (c.f. the four FCTs generated on start-up) and up to four bytes (a 32-bit word) in the transmit buffers would be plausible.

Due to the blockage at the sink end of the pipeline, the DSI transmission was stalled after two header bytes plus the first 126 bytes of the packet had been sent.

Figure 4 expands the link's start-up phase. The DSI sent seven FCTs to ports 1 and 8, and the 10X routers on these links sent four in the opposite direction in each case. Remembering that the DSI does not report the FCT that is part of the SpaceWire link start-up state machine, a total of six and three FCTs, respectively, are shown in this plot. The 10X Routers had already been started before this recording, so the flow control tokens between them are not recorded here.

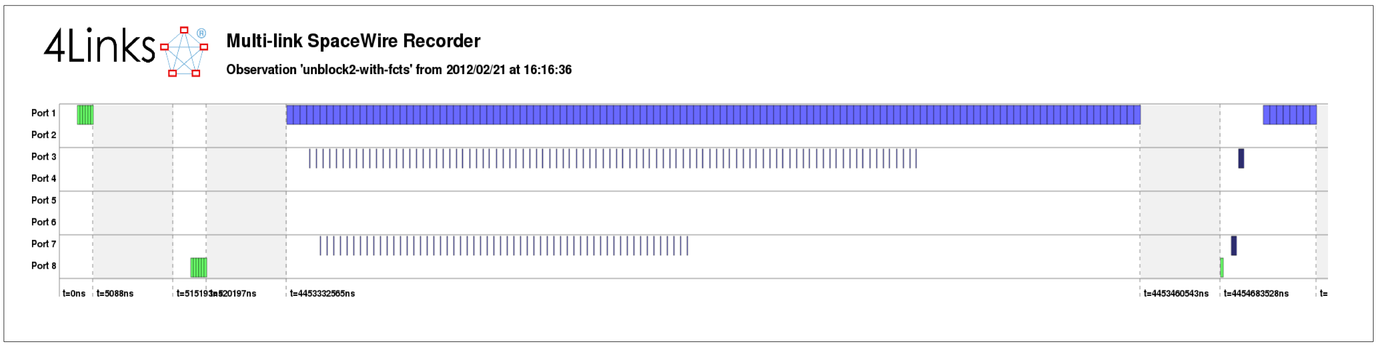


Fig.3. Data bytes filling a blocked pipeline

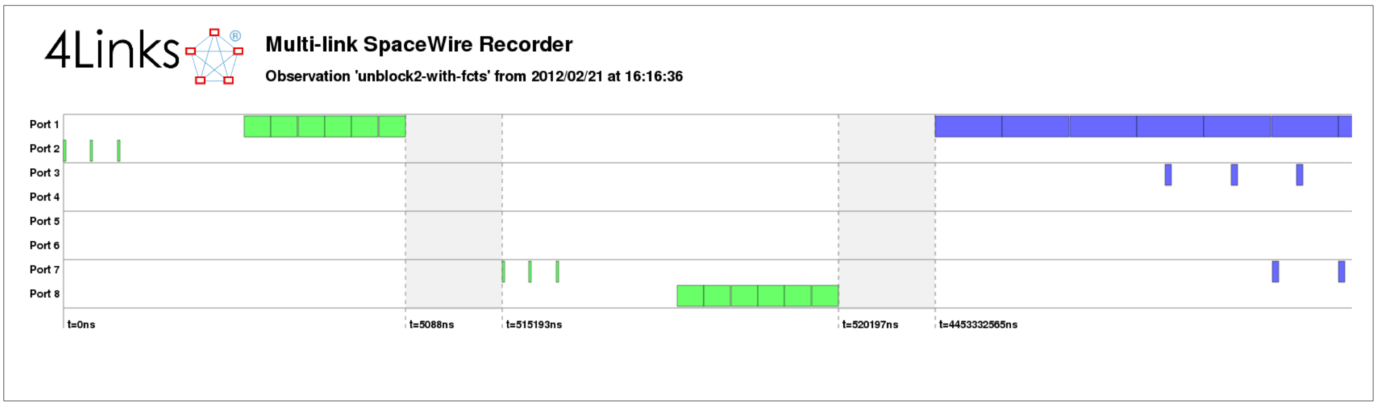


Fig.4. FCTs on link start

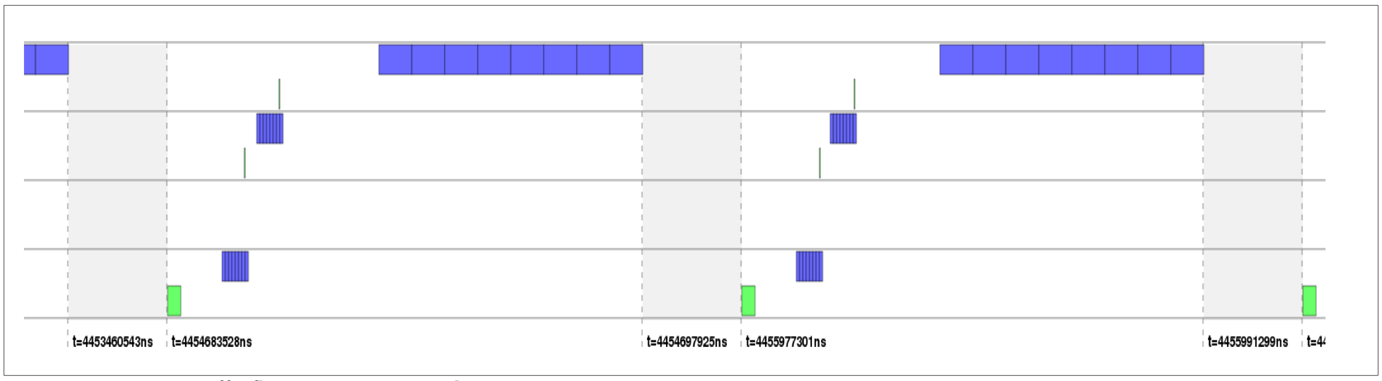


Fig.5. Traffic flow in response to single FCTs

Once the pipeline had become full, the script twice instructed the receiving DSI port to transmit a single FCT to Router B. In turn, these permitted eight more bytes to be sent from Router B to the DSI, and an FCT could also be sent further back along the pipeline from Router B to Router A. Likewise, another FCT was sent from Router A back to the transmitting DSI port. This is shown in figure 5, and illustrates the correct flow control behaviour.

C. Observing a link timeout in a 10X Router

In another experiment, router B was set up to spill blocked packets after a 1.3s timeout. Figure 6 shows that a burst of three FCTs was sent from the DSI back down the pipeline to Router B. Each of these FCTs caused eight further bytes to flow into the DSI, and an extra FCT to be sent back to Router A.

After the initial burst of three FCTs, an extra FCT was sent (by hand) after a delay that was long enough for the link from Router B to time out. It can be seen that five bytes of data were sent out of the router before the remaining data was spilled and the (expected) EEP was transmitted to the DSI.

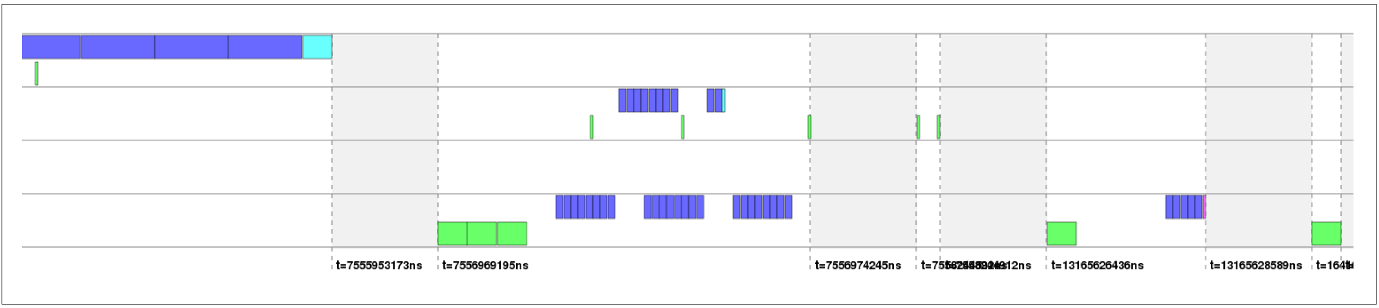


Fig.6. Observing a link timeout

IV. Observing the 10X Router in the Field

We used the information reported above to understand our observations of a 10X router that was running in the engineering model of a spacecraft.

Our goal was to investigate the very occasional loss of an uplinked telecommand, which occurred seemingly randomly, a few times per day. We monitored two SpaceWire links - one that carried the initial command, and the other that connected to the destination instrument - by passing them through a 4Links Multi-link SpaceWire Recorder, as shown in figure 7.

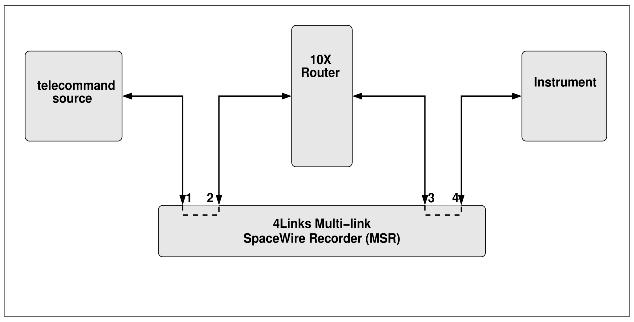


Fig.7. Monitoring SpaceWire links through a 10X Router

The 10X router was configured to disconnect its output links after a period of inactivity (Automatic deactivate driver mode) and to restart these links when they were required again (Start on Request mode).

Traffic within the spacecraft was scheduled on a cyclic basis with the majority of activities taking place every eighth of a second, and with a one second SpaceWire timecode transmission.

Between each of the eighth-second bursts of SpaceWire traffic, the SpaceWire output link from the router to the instrument was allowed to time-out and disconnect. The disconnect timeout was set to 0.01024s, and the 10X data sheet [3] specifies a ±2µs tolerance on this value. The MSR typically reports the time from the last use of the link to its disconnection as 0.01028s, once it has detected the time-out via the receiver state machine. This minor difference illustrates a limitation of black-box testing.

All messages within the part of the spacecraft that we observed were carried in SpaceWire Remote Memory Access Protocol (RMAP) [2] packets.

Simulated telecommands were injected into the system every few seconds.

A standard Multi-link SpaceWire Recorder data capture was performed on these two links over a period of 17.77 hours. This created a logging file of 844.6MB. Analysis of this logging file demonstrated that almost all of the SpaceWire RMAP packets were successfully passed through the router to the instrument and their results were returned successfully.

Various behaviours could be observed:

- An RMAP read or write transaction, issued when the router's output link was active, was always successful.
- An RMAP transaction, started whilst the router's output link was disconnected, was also reliable - the link being restarted link before transmission.
- An RMAP transaction issued immediately after a link had been shut down and disconnected, was also successfully transferred.
- However, over nearly 18 hours of recording, four RMAP packets (corresponding to uplinked telecommands) were observed to be spilled in the router and a truncated EEP packet was generated on the output link of the router. These four command packets were all scheduled for transmission through the SpaceWire network at the same time in the one-second schedule, just over 10ms after the last message and therefore just as the router was timing out and disconnecting its output link.

The time between the presentation of the telecommand RMAP packet to the router and any timeout before onward transmission on the SpaceWire link to the instrument was calculated in each case. There were 13737 telecommand packets in total. Of these:

- The vast majority of these packets arrived at, and were forwarded by, the router before the link to the instrument timed-out and disconnected - see figure 8.
- 170 more telecommand packets arrived after the output link had disconnected, and they were successfully transmitted to the instrument once the link had been restarted - see figure 9.
- The remaining four telecommand packets were four out of the seven packets that arrived very close to the time that the timeout was being triggered - just over 4µs before the MSR time-tagged the link disconnection event.

```
124:23:21:49.254 383 070 4s [-6]  1-->2  Data @0000   90 01 4C 20  4A 10 33 00  00 00 17 D0  00 03 26 A4  (16 bytes)
                                  1-->2  EOP at 124:23:21:49.254 399 071 7s (SOP + 16.001us) ~8.0 Mb/s
124:23:21:49.254 386 342 4s [-6]  3-->4  Data @0000   90 01 4C 20  4A 10 33 00  00 00 17 D0  00 03 26 A4  (16 bytes)
                                  3-->4  EOP at 124:23:21:49.254 402 342 4s (SOP + 16.000us) ~8.0 Mb/s
124:23:21:49.254 406 114 4s [-6]  3<--4  Data @0000   4A 01 0C 00  90 10 33 00  00 03 26 BE  EB 6A 00 00 ... (819 bytes)
                                  3<--4  EOP at 124:23:21:49.255 225 114 4s (SOP + 819.000us) ~8.0 Mb/s
124:23:21:49.254 409 439 7s [-6]  1<--2  Data @0000   4A 01 0C 00  90 10 33 00  00 03 26 BE  EB 6A 00 00 ... (819 bytes)
                                  1<--2  EOP at 124:23:21:49.255 228 447 7s (SOP + 819.008us) ~8.0 Mb/s
124:23:21:49.265 489 621 1s [-6]  1-->2  Data @0000   90 01 7C 20  4A 00 02 00  00 00 29 00  00 00 02 D0   DC BA 5F  (19 bytes)
                                  1-->2  EOP at 124:23:21:49.265 508 622 4s (SOP + 19.001us) ~8.0 Mb/s
124:23:21:49.265 493 251 7s [-6]  3-->4  Data @0000   90 01 7C 20  4A 00 02 00  00 00 29 00  00 00 02 D0   DC BA 5F  (19 bytes)
                                  3-->4  EOP at 124:23:21:49.265 512 251 7s (SOP + 19.000us) ~8.0 Mb/s
124:23:21:49.265 515 959 7s [-6]  3<--4  Data @0000   4A 01 3C 00  90 00 02 7F  (8 bytes)
                                  3<--4  EOP at 124:23:21:49.265 523 959 7s (SOP + 8.000us) ~8.0 Mb/s
124:23:21:49.265 519 349 1s [-6]  1<--2  Data @0000   4A 01 3C 00  90 00 02 7F  (8 bytes)
                                  1<--2  EOP at 124:23:21:49.265 527 349 1s (SOP + 8.000us) ~8.0 Mb/s
124:23:21:49.275 806 678 4s [-6]  3-->4  Disconnect
124:23:21:49.275 808 062 4s [-6]  3<--4  Disconnect
```

Fig.8. The 19-byte command is immediately sent to the router's output port

```
125:03:17:06.629 437 995 7s [-6]  1-->2  Data @0000   90 01 4C 20  4A 10 33 00  00 00 13 20  00 03 20 2D  (16 bytes)
                                  1-->2  EOP at 125:03:17:06.629 453 995 7s (SOP + 16.000us) ~8.0 Mb/s
125:03:17:06.629 441 138 4s [-6]  3-->4  Data @0000   90 01 4C 20  4A 10 33 00  00 00 13 20  00 03 20 2D  (16 bytes)
                                  3-->4  EOP at 125:03:17:06.629 457 138 4s (SOP + 16.000us) ~8.0 Mb/s
125:03:17:06.629 460 891 7s [-6]  3<--4  Data @0000   4A 01 0C 00  90 10 33 00  00 03 20 5A  A4 F1 0F FF ... (813 bytes)
                                  3<--4  EOP at 125:03:17:06.630 273 895 7s (SOP + 813.004us) ~8.0 Mb/s
125:03:17:06.629 464 235 7s [-6]  1<--2  Data @0000   4A 01 0C 00  90 10 33 00  00 03 20 5A  A4 F1 0F FF ... (813 bytes)
                                  1<--2  EOP at 125:03:17:06.630 277 243 7s (SOP + 813.008us) ~8.0 Mb/s
125:03:17:06.640 547 345 1s [-6]  1-->2  Data @0000   90 01 7C 20  4A 00 02 00  00 00 29 01  00 00 02 5C   DC BA 5F  (19 bytes)
                                  1-->2  EOP at 125:03:17:06.640 566 346 4s (SOP + 19.001us) ~8.0 Mb/s
125:03:17:06.640 551 566 4s [-6]  3-->4  Disconnect
125:03:17:06.640 552 998 4s [-6]  3<--4  Disconnect
125:03:17:06.640 576 246 4s [-6]  3-->4  Data @0000   90 01 7C 20  4A 00 02 00  00 00 29 01  00 00 02 5C   DC BA 5F  (19 bytes)
                                  3-->4  EOP at 125:03:17:06.640 595 246 4s (SOP + 19.000us) ~8.0 Mb/s
125:03:17:06.640 599 435 7s [-6]  3<--4  Data @0000   4A 01 3C 00  90 00 02 7F  (8 bytes)
                                  3<--4  EOP at 125:03:17:06.640 607 435 7s (SOP + 8.000us) ~8.0 Mb/s
125:03:17:06.640 602 543 7s [-6]  1<--2  Data @0000   4A 01 3C 00  90 00 02 7F  (8 bytes)
                                  1<--2  EOP at 125:03:17:06.640 610 543 7s (SOP + 8.000us) ~8.0 Mb/s
125:03:17:06.650 883 070 4s [-6]  3-->4  Disconnect
125:03:17:06.650 884 390 4s [-6]  3<--4  Disconnect
```

Fig.9. The 19-byte command is transmitted when the link restarts

```
124:23:21:54.254 473 494 4s [-6]  1-->2  Data @0000   90 01 4C 20  4A 10 33 00  00 00 17 D0  00 03 26 A4  (16 bytes)
                                  1-->2  EOP at 124:23:21:54.254 489 495 7s (SOP + 16.001us) ~8.0 Mb/s
124:23:21:54.254 476 659 7s [-6]  3-->4  Data @0000   90 01 4C 20  4A 10 33 00  00 00 17 D0  00 03 26 A4  (16 bytes)
                                  3-->4  EOP at 124:23:21:54.254 492 659 7s (SOP + 16.000us) ~8.0 Mb/s
124:23:21:54.254 496 453 1s [-6]  3<--4  Data @0000   4A 01 0C 00  90 10 33 00  00 03 26 BE  EB 92 00 00 ... (819 bytes)
                                  3<--4  EOP at 124:23:21:54.255 315 455 7s (SOP + 819.003us) ~8.0 Mb/s
124:23:21:54.254 499 557 1s [-6]  1<--2  Data @0000   4A 01 0C 00  90 10 33 00  00 03 26 BE  EB 92 00 00 ... (819 bytes)
                                  1<--2  EOP at 124:23:21:54.255 318 563 7s (SOP + 819.007us) ~8.0 Mb/s
124:23:21:54.265 595 643 7s [-6]  1-->2  Data @0000   90 01 7C 20  4A 00 02 00  00 00 29 01  00 00 02 5C   DC BA 5F  (19 bytes)
                                  1-->2  EOP at 124:23:21:54.265 614 645 1s (SOP + 19.001us) ~8.0 Mb/s
124:23:21:54.265 599 886 4s [-6]  3-->4  Disconnect
124:23:21:54.265 601 310 4s [-6]  3<--4  Disconnect
124:23:21:54.275 896 467 7s [-6]  3-->4  Data @0000   90 01 7C 20  4A  (5 bytes)
                                  3-->4  EEP at 124:23:21:54.275 901 467 7s
124:23:21:54.286 182 486 4s [-6]  3-->4  Disconnect
124:23:21:54.286 183 886 4s [-6]  3<--4  Disconnect
```

Fig.10. The 19-byte command is spilled as the output link is disconnected

The packets affected had the three longest times and the seventh-longest time between the arrival of the packets and the disconnection of the outgoing link at the router, so why all seven were not affected is not clear. However, the arrival time of these seven packets all fell within 150ns of each other, relative to the following link disconnection. Once the router committed to shutting down its output link, it appears to have treated the arriving packet as if it was blocked, rather than simply pausing while the link restarted again. Thus, the four failing packets were truncated by the router and spilled when it restarted its output link about 10ms after the timeout, and only their five leading bytes, plus an EEP, were eventually sent to the instrument (where they were discarded as incomplete). See figure 10 for these timings. Notice that the normal link restart, shown in figure 9, occurs about 25μs after the packet arrives (as specified in [1], figure 8-2), not 10ms as for the erroneous case.

The five-character output packet with the EEP termination is identical in nature to the truncated packets generated by the 10X router in the laboratory experiments described earlier. It is therefore likely that the 10X router, if it receives a packet at almost exactly the time when it decides to disconnect one of its output links, treats it as blocked rather than buffering it for transmission when the link is restarted. This vulnerable arrival period appears to be around 150ns in duration, or less than two bit intervals at 10Mb/s.

There is not a good correlation in the differences of times from the last use of the outgoing SpaceWire link to the instrument and the arrival time of the next telecommand packet as a predictor for this failing behaviour - the ±2µs tolerance on the timeout blurs this measure.

V. Results

Analysis of our measurements shows that the 10X router appears to have a 32-byte input buffer, and that up to three further bytes are stored in the router's switching fabric and its output buffers when the output link is blocked. Flow control tokens were propagated properly by the routers throughout.

When used in a real system, with link disconnection after an idle period, packet loss was observed if the packet arrived at the 10X router in a predictable very brief interval before a link disconnection. This was sufficiently disruptive that the spacecraft's timing schedule had to be redesigned, calling into question the benefit of this operating mode.

VI. Conclusions

Using test equipment with very fine-grained time-tag recording capabilities allows a precise understanding of the behaviour of SpaceWire devices to be determined. These results were fed into the development stages of an ESA spacecraft mission, where they helped to resolve some problems with the configuration of a 10X router.

VII. References

[1] ECSS-E-ST-50-12C, "SpaceWire - Links, nodes, routers and networks", ESA, 2008

[2] ECSS-E-ST-50-52C, "SpaceWire - Remote memory access protocol", ESA, 2010

[3] C. McClements, S. Parkes and G. Kempf, "SpW-10X SpaceWire Router User Manual", Atmel, 2008, available at http://www.atmel.com/Images/UoD_SpW_10X_UserManual_3_4.pdf

Probabilistic Analysis of SpaceWire Communication Processes

Session: SpaceWire Test and Verification, Long Paper

Yuexing Li, Xiaojuan Li, Rui Wang, Yong Guan
College of Information Engineering,
Capital Normal University
Beijing, China
liyuexingok@163.com, lixj66@gmail.com,
rwang04@163.com, guanyong@mail.cnu.edu.cn

Jie Zhang
College of Information Science & Technology, Beijing
University of Chemical Technology
Beijing, China
jzhang@mail.buct.edu.cn
Xiaoyu Song
Dept.of ECE, Portland State University Portland,
Oregon, USA

Abstract—**SpaceWire is a bus standard for high speed data transmission in aerospace. With the complexity of application it is very important to guarantee the reliability and stability of the transmission system. Since the system exhibits both probabilistic and nondeterministic behavior, this paper applies probabilistic model checking to make quantitative formal analysis for SpaceWire. We model the process of the link initialization and link maintenance of the protocol with probabilistic model checking. Sender model, receiver model and channel model are set up respectively. Owing to that the packets may be lost during transmission, the probability of losing the packet is considered in channel model, which is more close to the reality. The models are encoded as a Markov decision process (MDP) for analysis by the probabilistic model checker PRISM. The paper verifies key properties in Probabilistic Computation Tree Logic (PCTL), including the maximum probability of the successful initialization in both directions within T time, and the maximum probability of maintaining link during data transmission under constraints. A systematic level model is built and the probability of losing packet with constraints is evaluated. The quantitative verification results provide a useful reference for the design, implementation and application of SpaceWire.**

***Index Terms*—SpaceWire, Probabilistic Model checking, Markov Decision process, PRISM**

I. Introduction

The European Space Agency proposes SpaceWire bus standard based on IEEE1355-1995 and IEEE1596.3 (LVDS). SpaceWire provides a unified high-speed data processing infrastructure for connecting together high data-rate sensors and downlink telemetry subsystem [1]. In recent years SpaceWire bus technology is mainly used in aerospace field. It requires the internal communication network is capable of high speed, reliability and resistance to radiation. As the environment of SpaceWire's application is harsh, it is very important to guarantee the reliability. It is significant to verify the correctness of SpaceWire design [2].

Nowdays, most researchers use testing and simulation to analyze SpaceWire. For example, Harbin Institute of Technology simulates every functional module of SpaceWire by constructing test bench [3]. ASA/Goddard Space Flight Center provides the Total Verification System (TVS) to test SpaceWire [4]. Such traditional research methods have some limitations. On the one hand, the traditional verification methods can't cover all paths of execution for a large and complex system. On the other hand, the traditional verification methods are often used to test the known types of fault and difficult to find subtle fault [5]. Formal verification method is proposed to model mathematically, and then using strict mathematical reasoning proves the correctness of the design. There are two fundamental techniques in formal verification: model checking and theorem proving [6]. Li Li-ming etc. verify equivalence of the DS code design and specification of SpaceWire bus physical level by theorem proving [7]. Beijing Engineering Research Center of High Reliable Embedded System etc. propose model checking to verify the SpaceWire error detection mechanism [8]. However, model checking and theorem proving can only have qualitative analysis on SpaceWire.

In order to systematically validate SpaceWire protocol under uncertain environment, this paper proposes probabilistic model checking to build formal model and make verification. Probabilistic model checking, an automatic verification technique for the systems that exhibit random behavior, can be applied to the analysis, design and verification of such protocols. The basic idea is to construct a mathematical model that captures the system's behavior, and then use it to enable a range of exhaustive and quantitative analyses of properties. Like traditional model checking, this technique involves constructing, from a description in some high-level formalism, a finite-state model of a real-life system, but additionally including information about the likelihood and timing of transitions between states occurring [9, 10, 11].

This paper is organized as follows: Section II introduces the exchange level of SpaceWire. Section III is probabilistic model checking method. It contains a brief description of MDP and

PCTL. Section IV describes formal modelling of SpaceWire. Section V presents several properties in PCTL for verification. The last section is a conclusion to the paper.

II. Introduction of The exchange level of SpaceWire

The exchange level is responsible for making a connection across a link and for managing the flow of data across the link. The exchange level contains mainly sending module, receiving module, control module. Sending module is responsible for encoding data and transmitting it using the DS encoding technique. Receiving module is responsible for receiving NULL,FCT and Time-Code and decoding the DS signals (Din and Sin) to produce a sequence of N-Chars(data, EOP, EEP) that are passed on to the host system. The main function of control module is to control conversion between multiple states.

Task of exchange lever is divided into two stages: link initialization and normal operation. Initialization is described in Fig.1. (a), normal operation is described in Fig.1. (b).

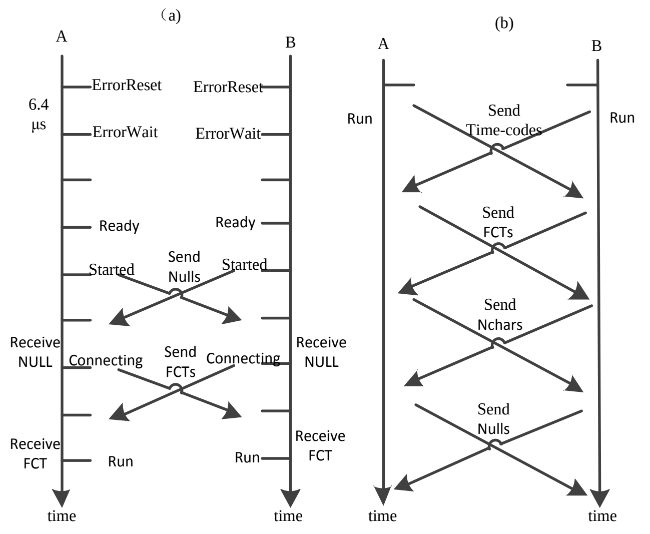


Fig. 1. Link initialization and normal operation

Link initialization: link A and link B remains in the ErrorReset state for approximately 6.4 µs and then moves to the ErrorWait state. Link A and link B can’t send and receive any characters, when they are in the ErrorReset state. In the ErrorWait state link A and B can’t send any characters, but link A and B can receive NULLs. The link interface remains in the ErrorWait state for 12.8 µs and then moves into the Ready state. The link interface moves from the Ready state to the Started state as soon as the link is enabled. In the Started state the link A and B start sending NULLs. It remains in this state until the link A and B detects that a NULL is received over the link or until a connection timeout expires. The connection timeout is set to a nominal 12.8 µs. If a NULL is received then the link interface moves to the Connecting state. If no NULL is received within 12.8 µs it moves to the ErrorReset state. In the latter case the link interface goes through the reset sequence (ErrorReset, ErrorWait, Ready) and attempts to make a connection again a short time later. In the Connecting state the link interface sends some FCTs and waits for the reception of an FCT. If an FCT is received the link interface moves on to the Run state. If an FCT is not received within 12.8 µs then link connection was not made properly, so the link interface returns to the ErrorReset state. The link interface then goes through the reset sequence (ErrorReset, ErrorWait, Ready) and attempts to make a connection again a short time later.

Link normal operation: When the link enters the Run state it starts normal operation, sending and receiving data and control characters. These characters are NULLs, FCTs, Time-Codes, N-chars.

III. Probabilistic Model Checking

A. Probabilistic Model

Probabilistic model checking is based on the construction and analysis of a probabilistic model. It has been applied to a variety of different types of model, such as Markov Decision Processes (MDPs). MDPs are widely used to model systems that exhibit both probabilistic and nondeterministic behavior. Probability is employed to quantify aspects of system behavior where probability distributions are known. Nondeterminism is used to model unknown environments [12]. Since the behavior of SpaceWire is stochastic, we build formal model based on MDP. Properties are then expressed using Probabilistic Computation Tree Logic (PCTL) [13].

Definition1. A labeled Markov decision process is a tuple

$M = (S, \bar{S}, Steps, L)$

- S is the finite set of states
- $\bar{S} \in S$ is the initial state
- $Steps : S \to 2^{Act \times Dist(S)}$ is the transition probability function where Act is a set of actions and Dist(S) is the set of discrete probability distributions over the set S
- $L : S \to 2^{AP}$ is a labelling with atomic propositions

In an MDP, several actions may be available in a given state, each corresponding to a probability distribution. The behavior of an MDP M is as follows. First, a choice between one or more actions is made nondeterministically; secondly, for the chosen action, a successor state is chosen randomly, according to the transition probability function.

B. Probabilistic Computation Tree Logic

Probabilistic Computation Tree Logic (PCTL) is an extension of the non-probabilistic Computation Tree Logic (CTL) [14,15].

Definition2. The syntax of PCTL is given by: let *a* be an atomic proposition, used to identify states of interest, $p \in [0,1]$ is a probability. The symbols *X* and *U* represent the usual operators for next and until. $\sim \in \{\le, \ge, \langle, \rangle\}$, $m \in \mathbb{N}$.

state formulas:

$\phi ::= \text{true} \mid a \mid \neg\phi \mid \phi \wedge \phi \mid p_{\sim p}[\varphi]$

path formulas:

$\varphi ::= X\phi \mid \phi U^{\le m}\phi \mid \phi U \phi$

Definition3. The semantics of PCTL formulae is defined: A state s ∈ S satisfies φ, denoted $s \models \varphi$, if the following holds:

$$s \models p_{\sim p}[\varphi] \Leftrightarrow p_s^{\pi}(\varphi) \sim p$$

$$s \models R_{\sim r}[C^{\leq k}] \Leftrightarrow E_s^{\pi}(X_{C\leq k}) \sim r$$

$$s \models R_{\sim r}[F_{\varphi}] \Leftrightarrow E_s^{\pi}(X_{F\varphi}) \sim r$$

IV. Formal Modelling of SpaceWire

In order to verify systematic properties and analyze the effect of environment on SpaceWire, the paper models the process of the link initialization and link maintenance of the protocol based on MDP. Modelling structure of SpaceWire is shown in figure 2. The models are sending model, receiving model and channel model. SpaceWire provides a full-duplex communication network, so we build two channel models: sending channel and receiving channel.

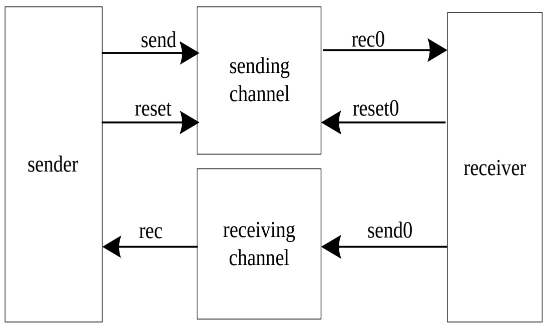


Fig. 2. Model Structure

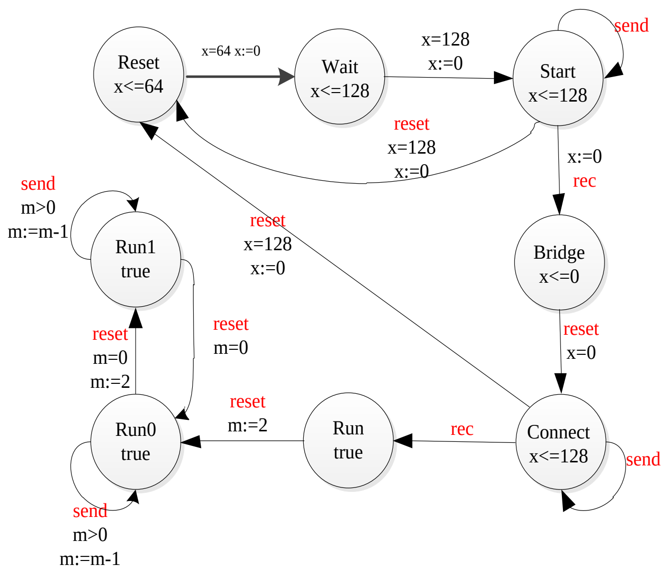


Fig. 3. The model of sender

Figure 3 is sender model and describes the process of the link initialization and normal operation. The model commences in the location *Reset*. *X* of the model is clock variable. The link waits approximately 6.4μs in the location *Reset* before the transitions from *Reset* to *Wait*. After 12.8μs the link of the model moves to the location *Start* directly. In *Start*, the link can send NULLs and remains 12.8μs at most. The transition is labelled with the event *send* to inform the sending channel that the link will send NULLs. The link may receive a NULL by means of the event *rec* within 12.8μs. If the link receives a NULL within 12.8μs, it will move to the location *Bridge*. If the link doesn't receive a NULL after 12.8μs, it will move to the location *Reset* and attempt to make a connection again. The location *Bridge* is a committed location, and therefore must be left immediately. The link sends a *reset* event to inform sending channel to clear the sending buffer from the location *Bridge* to the location *Connect*. In *Connect* the link can send FCTs and NULLs and remains 12.8μs at most. If the link doesn't receive a FCT after 12.8μs, it will move to the location *Reset* to restart a connection. When the link receives a FCT within 12.8μs, it will move to the location *Run*. The link initialization is successful when the receiver model also gets to the location *Run*.

After *Run* the link will reach normal operation. The model describes the process that the link sends packets from the location *Run0* to *Run1*. We make link send 2 FCTs in the location *Run0* so that we can make formal verification for the process of the link normal operation. In order to indicate the link is still active the link shall send NULL in location *Run1*.

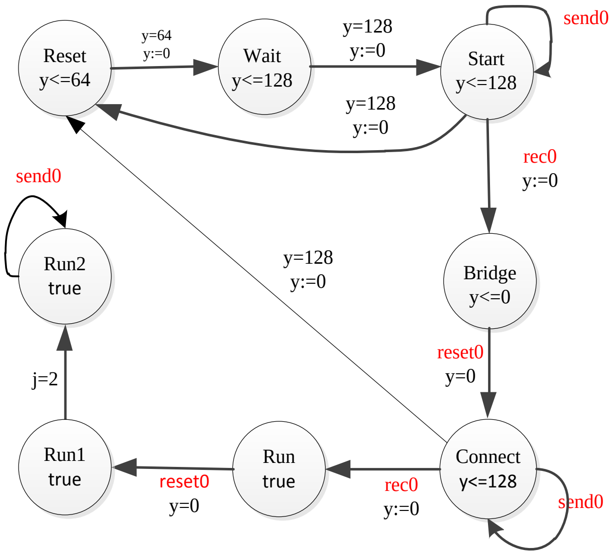


Fig. 4. The model of receiver

Figure 4 is receiver model and describes the process that the process of the link initialization and normal operation. The process of this model is similar to figure 3. Here we don't describe this figure in detail. The *send0* event informs the receiving channel the link will reply NULLs and FCTs in the location *Start* and *Connect*. The link will inform the sending channel to clear receiving buffer by *reset0* event. When the sender model and receiver model gets to the location *Run*, both ends of the link initialization is successful and the link starts normal operation. The model presents the process that the link receives the packets from the location *Run1* to the location *Run2*. We can check the properties of link normal operation by the transition from the location *Run1* to the location *Run2*. The link will move to the location *Run2* from the location *Run1* if the link receives 2 N-Chars. The link shall send NULL to indicate the link is still active in location *Run2*.

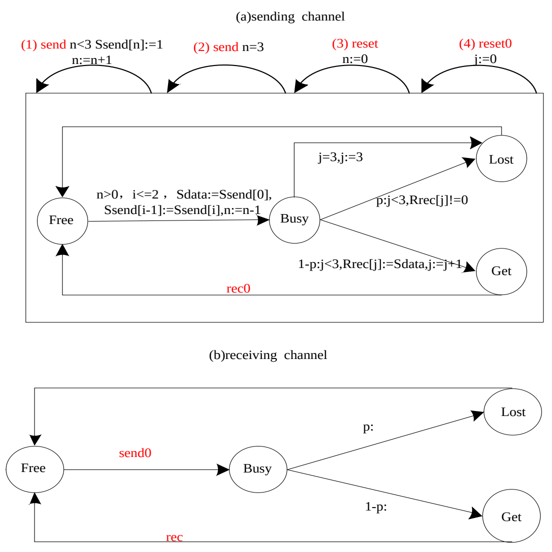


Fig. 5. channel

The packets may be lost during transmission .The channel model sets probability of losing packets .We make the assumption that the probability of losing packets is $p \in [0,1]$. Figure 5(a) describes the process that the sender sends packets to receiver through the sending channel. We create a sending queue and a receiving queue in the sending channel model. The size of the sending and receiving queue is three. The box labelled with four transitions which surrounds the model denotes that these transitions are available in all of the locations of the model. Transition (1) describes that if the sending queue is not full, the sending channel receives a *send* event from sender model and adds the corresponding packet to the sending queue. Transition (2) describes that if the sending queue is full, the sending channel loses the packets. Transition (3) describes that the sending queue is cleared by a *reset* event from the sender model and setting variable *n* to zero. Transition (4) describes that the receiving queue is cleared by a *reset0* event from the receiver model and setting variable *j* to zero.

The sending channel is sending packets to the receiver from the location *Free* to *Busy*. There are three transitions in *Busy*. The receiving queue is full by setting variable *j* to three and the packets are lost. The receiving queue is not full if variable *j* is less than three. The sending channel adds the packet to the receiver with the probability *1-p*. The packet is lost with the probability *p* by the sending channel. The sending channel is free from the location *Lost* and *Get* to *Free*. The sending channel adds the packet to the receiver model by event *rec0* from the location *Get to Free.*

Figure 5(b) describes the process that the receiver sends packets to the sender through the receiving channel. The transition will happen from the location *Free* to *Busy* If the receiving channel receives event *send0* from the receiver. The packet is lost by the receiving channel with the probability *p* from the location *Busy* to *Lost*. The packet is received by the sender with the probability1- *p*. The receiving channel informs the sender that the packet is received by the event *rec* from the location *Get* to *Free*.

V. Probabilistic Analysis and Verification

PRISM [16] provides model checking for several types of probabilistic models: discrete-time Markov chains and Markov decision processes. PRISM uses a uniform modelling language for all the probabilistic models that it supports. This is a textual language [17]. This paper uses PRISM to make formal verification. The properties are represented by formulae in the probabilistic logic PCTL. We now report on the probabilistic model checking results.

Prop.1: When the link initialization is successful at first try, the maximum probability is shown as Fig 6 in which the probability of channel losing packet is different. In PCTL the property is expressed by: *Pmax=? [F s=7&r=7&times=1].*

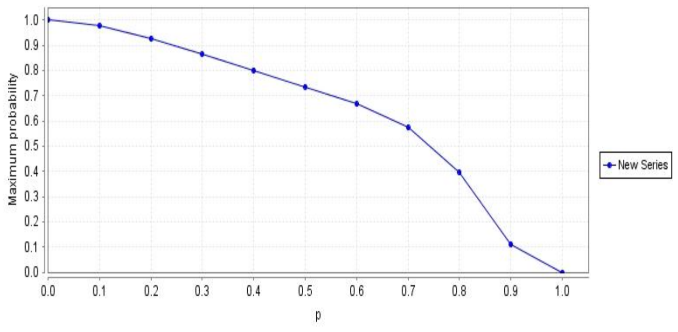
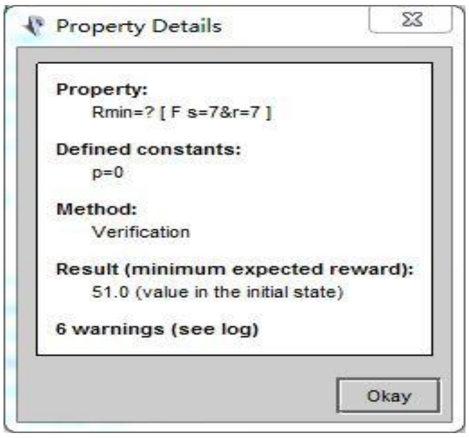


Fig. 6. result of property 1

The horizontal axis represents the probability of channel losing packet. When the link initialization is successful at first try, the maximum probability is represented in vertical axis. It indicates that the higher the probability of losing packet is, the lower the maximum probability is in Fig 6. As the link can continue to send NULLs and FCTs within 12.8μs in the state of Started and Connecting during initialization, the maximum probability is still high when the probability of losing packet is increasing.

Prop.2: The minimum time of link initialization is presented in Fig 7.In PCTL the property is expressed by: *Rmin=? [F s=7&r=7].*


Property Details

Property:
Rmin=? [F s=7&r=7]

Defined constants:
p=0

Method:
Verification

Result (minimum expected reward):
51.0 (value in the initial state)

6 warnings (see log)

Fig. 7. result of property 2

Fig 7 shows the minimum expected reward to initialize successfully. The result shows that it will take at least 51 time units to initialize successfully. The actual time is 4 times as much as the model's time, so it spends at least 204 time units (20.4μs) to initialize. It has 64 time units (6.4μs) and 128 time units (12.8μs) delay in the states of *ErrorReset* and *ErrorWait.* Then it takes 12 time units (12μs) to send a *NULL* and an *FCT*.

Prop.3: When the link initialization is successful, the maximum probability within *T* time is shown in Fig8. In PCTL the property is expressed by: *Pmax=? [F s=7&r=7&t<T]*.

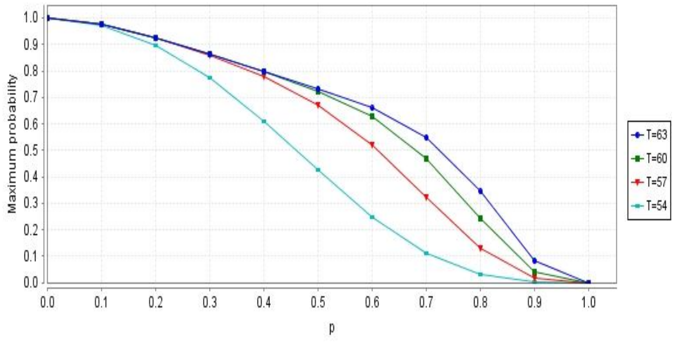


Fig. 8. result of property 3

There are four curves in the graph. The light green curve shows the verification results when *T* equals 54. The red curve is the results of which the time *T* is 57. When the time *T* is 60 and 63, the verification results are green curve and blue curve. For the same probability of losing packet *p*, the result that *T* equals 63 is bigger than the others' result. The reason is that the link can send more NULLs and FCTs when the time *T* increases. According to the result of verification in the graph, we can limit time for the link initialization.

Prop.4: When the link starts normal operation, the maximum probability within T time is shown in Fig 9. In PCTL the property is expressed by: *Rmin=? [F s=10&r=9&t<T]*.

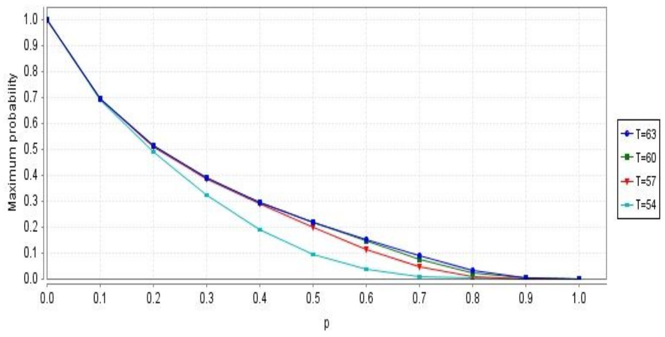


Fig. 9. result of property 4

With the increasing of time T the maximum probability is higher at the same value of *p*. Combining Fig 8 and Fig 9 it shows that the maximum probability of Fig 9 is lower than Fig 8 at the same value of *p* and *T*. The reason is that link initialization is first reached before link normal operation. The blue curve in which *T* is equal to 63 is almost same with the green curve in which *T* equals 60. We can infer that the curve will be stable after the condition in which *T* is more than 63.

VI. Conclusion

This paper applies probabilistic model checking to make quantitative formal verification for the exchange level design of SpaceWire. Firstly, the exchange level of SpaceWire is modeled abstractly on the basis of MDP. Sender model, receiver model and channel model are included. Then the paper verifies key properties of SpaceWire by PCTL. The properties are related to the maximum probability in the condition of which link initialization is successful and the link reaches the normal operation. The quantitative analysis for SpaceWire is carried out by probabilistic model checking and the results of quantitative verification can provide a useful reference for the design, implementation and application of SpaceWire.

Acknowledgment

This research was supported by the BJNSF (4122017), Beijing Education Science and technology development project (KZ201210028036), ISTCP (2011DFG1-3000), and International Science and technology cooperation project(2010DFB10930), The authors are grateful to Prof. Jin Shengzhen for his valuable suggestions.

References

[1] ECSS Standard ECSS-E-ST-50-12C. SpaceWire-Links Nodes, Routers and Networks[S].

[2] Wang Fang, Li Ke, Su Lin and Geng Lihong. Development of onboard solid state recorder for space solar telescope [J]. Acta Electronical Siniea, 2004,32 (3):472-475.

[3] Wang--Na. Design of SpaceWire Node Interface [D].Harbin Institute of Technology, 2007,7.

[4] Damaris L.Guevara, Omar A. Haddad. Using TVS to Verify SpaceWire Designs[C]. SpaceWire-2011 Proceedings of the 4th International SpaceWire Conference. San Antonio,2011: 220-225.

[5] Michael C McFarland. Formal verification of sequential hardware: a tutorial[J].IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems,1993,12(5):633-654.

[6] Meenakshi B. Formal Verificaiton[J].Resonance,2005,10(5):26-38.

[7] LI Li-ming, Guan Yong, WU Min-hua ,Zhang Jie and Shi Zhi-ping. Formal Method for Verifying SpaceWire Encoding Circuit by Applying Theorem Proving [J]. Journal of Chinese Computer Systems, Vol33, No.6, 2012:1372-1376.

[8] Dong Lingling, Guan Yong, Li Xiaojuan, Shi Zhiping, Zhang Jie and Hua Wei. Verification for SpaceWire Error Detection Mechanism by LTL Model Checking[J]. Computer Engineering and Applications,2012,48(22):88-94.

[9] Marta Kwiatkowska, Gethin Norman ,David Parker. Advances and Challenges of Probabilistic Model Checking[C].In Proc.48th Annual Allerton Conference on Communication, Control and Computing, University of Illinois at Urbana-Champaign, 2010:1691-1698.

[10] Nipkow, Tobias, O. Grumberg and B. Hauptmann,eds.Software Safety and Security:Tools for Analysis and Verification[M].IOS Press,2012.

[11] Andrew Hinton, Marta Kwiatkowska, Gethin Norman and David Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems[C]. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06). 2006: 441-444.

[12] David Anthony Parker, Implementation of Symbolic Model Checking for Probabilistic Systems[D].University of Birmingham,2002,8.

[13] Marta Kwiatkowska and David Parker. Advances in Probabilistic Model Checking[C]**.** Proceedings of the 11th international conference on Verification, Model Checking, and Abstract Interpretation. Springer-Verlag, 2010: 25-25.

[14] http://www.prismmodelchecker.org/

[15] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems[C]. In Proc. 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of LNCS, pages 585-591, Springer. July 2011.

[16] Matthias Fruth D, formal methods for the Analysis of Wireless Network Protocols[D].University of Oxford,2011.

[17] Marta Kwiatkowska, Gethin Norman and David Parker. Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach[J]. International Journal on Software Tools for Technology Transfer (STTT), 2004, 6(2):128–142.

Margin Testing of SpaceWire Devices

SpaceWire test and verification, Long Paper

Alan Spark, Pete Scott
STAR-Dundee
Dundee, Scotland
alan.spark@star-dundee.com,
pete.scott@star-dundee.com

Paul Crawford, Steve Parkes
University of Dundee
School of Computing
Dundee, Scotland
psc@sat.dundee.ac.uk,
sparkes@computing.dundee.ac.uk

Abstract— **The SpaceWire Physical Layer Tester (SPLT) is a device designed to support margin testing of the physical level of SpaceWire devices. This paper gives an overview of some useful tests that can be performed with the STAR-Dundee SPLT, whose hardware was introduced at the 2011 SpaceWire Conference [1]. The software used to control the SPLT to conduct margin and production tests is also introduced.**

The SPLT's unique Low Voltage Differential Signalling (LVDS) transmitters can be configured through software to simulate various forms of signal degradation.

The platform software can perform margin testing on individual characteristics of the LVDS transmitters. A start point is set for the desired parameter (for example Voltage swing) and fixed values for all other transmitter characteristics are configured. The test parameter is then progressively degraded by the software until the SpaceWire link disconnects. Performing margin testing on all of the variables will give an envelope outside of which the system will cease to function. Margin test data may be used to define a set of production parameters under which a Unit Under Test (UUT) is expected to operate. This data is fed into a production test feature of the software platform for testing and validation of flight hardware. The production test feature manually, or automatically, explores the UUT's performance at the extremes of the production parameters and prints out a results sheet detailing the tests performed, and any failures that have occurred.

The SPLT also features high speed analogue buffers on either end of the LVDS receiver termination resistors. Measurements are presented of the tests outlined above using an oscilloscope interfaced through these buffers. The eye pattern is observed and verified using masks on oscilloscope software.

***Index Terms*—Physical Layer, Software, Margin, Production, Test, LVDS, Eye Pattern, SpaceWire**

I. Introduction

The SpaceWire Physical Layer Tester is a new test unit from STAR-Dundee that supports margin testing of the physical layer of SpaceWire. It is able to operate in two principal modes: unit test mode and system test mode.

In unit test mode the SPLT is connected to the unit under test and sources the SpaceWire packets used during the testing. The SPLT is able to manipulate the physical level signals, modifying the data to strobe skew, the LVDS positive and negative signal skew, the LVDS differential signal level and the LVDS common mode skew. It can also add controlled jitter to the LVDS signals. The unit under test (UUT) receives the aberrated LVDS signals and will operate normally until they become too severe. At this point the UUT will receive and detect errors on the line and disconnect the SpaceWire link. This disconnection is detected by the SPLT and used to determine the margin of operation

In system test mode the SPLT is placed in the SpaceWire link between two units and both units are tested at the same time. Each unit sends packets to the other unit. The SPLT modifies the physical level signals to assess the margins of the system.

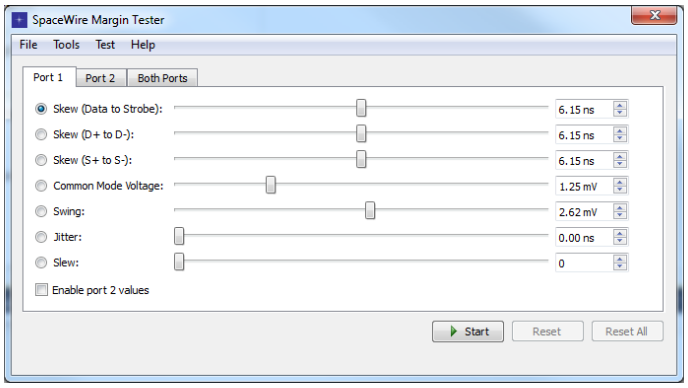
The SPLT has specialised LVDS drivers which are used to measurably control the offset, swing, slew, skew and jitter characteristics of its transmitted SpaceWire signals. A UUT will disconnect the link when the SpaceWire signal received from the SPLT is sufficiently deteriorated to prevent recovery of its bit stream. The envelope of aberrations under which the UUT is able to sustain a SpaceWire link can then be determined, for example the amount of data to strobe skew that can be sustained when all the other LVDS signal parameters are nominal.

Application software provided with the SPLT supports both margin testing and production testing. These separate functions make it easy to determine the operational margins of a UUT and to then define on-going test criteria to ensure that these margins are met during production.

The SPLT software also provides an application programming interface (API) to allow users to integrate SPLT functionality into their own applications.

II. Margin Testing

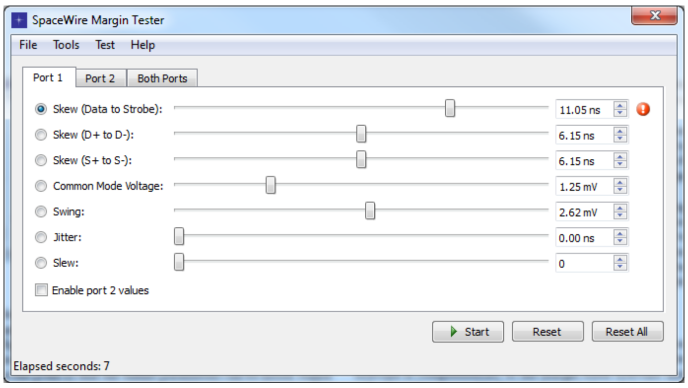
The margin testing software can be used to determine the amount of signal degradation that a UUT can cope with. The user can set the initial amount of degradation to be applied to the signal for various different types of signal aberration. The software will then automatically increase the amount of degradation that is applied to the signal for a chosen type of aberration and the test will stop as soon as a failure is detected.



Skew (Data to Strobe): 6.15 ns
Skew (D+ to D-): 6.15 ns
Skew (S+ to S-): 6.15 ns
Common Mode Voltage: 1.25 mV
Swing: 2.62 mV
Jitter: 0.00 ns
Slew: 0
Enable port 2 values

Fig. 1. Initial SpaceWire Margin Tester screen.

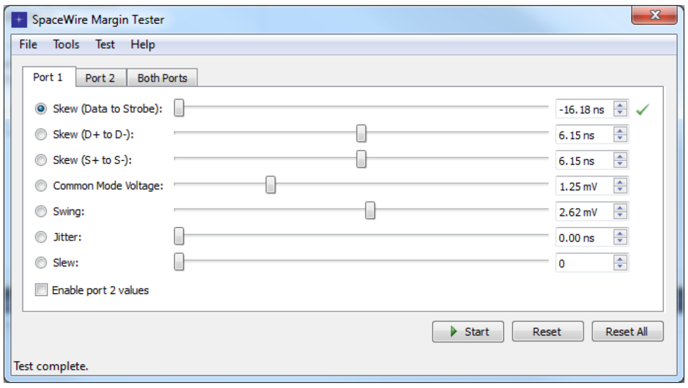
In Fig. 1. the SpaceWire Margin Tester software is shown in its default state where all aberrations are set to their nominal values. The type of aberration that is to be automatically altered is selected using the associated radio button. Meanwhile, the static values of the other aberrations can be set manually. This allows for each aberration to be tested in isolation whilst keeping other variables at a constant level.



Skew (Data to Strobe): 11.05 ns
Skew (D+ to D-): 6.15 ns
Skew (S+ to S-): 6.15 ns
Common Mode Voltage: 1.25 mV
Swing: 2.62 mV
Jitter: 0.00 ns
Slew: 0
Enable port 2 values
Elapsed seconds: 7

Fig. 2. Failure of margin testing on Data to Strobe Skew.

The automatic margin testing feature constantly degrades the signal by increasing the amount of aberration that is applied until either the link fails, or else continues to run under maximum aberration. In the example shown in Fig. 2. the test has failed after the Data to Strobe Skew was increased to 11.05 ns. The testing immediately stops at this point so that the failure parameters can be noted. Failed tests are indicated by a red exclamation mark icon. The aberration value may then be set to the value prior to where the failure last occurred before the other types of aberration are tested.



Skew (Data to Strobe): -16.18 ns ✓
Skew (D+ to D-): 6.15 ns
Skew (S+ to S-): 6.15 ns
Common Mode Voltage: 1.25 mV
Swing: 2.62 mV
Jitter: 0.00 ns
Slew: 0
Enable port 2 values
Test complete.

Fig. 3. The Data to Strobe Skew test completed successfully.

If a link continues to run under maximum aberration (shown in Fig. 3.) a green tick icon is shown alongside the associated slider. Each of the aberrations can be applied one after the other so that the full spectrum is tested and the acceptable margin of operation can be determined.

By default, aberrations will be applied to analogue port 1 on the SPLT. As shown in Fig. 3. this can be changed to port 2 or both ports using the tabbed display.

The SPLT hardware supports various types of Jitter including triangular, square and sawtooth and Gaussian. It is possible to configure the Jitter parameters in the software. The level of which is then controlled by the associated slider in the margin tester screen.

Each SPLT unit is calibrated to the optimal settings during production. However, depending on various factors such as cable length and ambient temperature, the device may need to be recalibrated by the user to ensure that the most accurate levels of aberration are being output by the hardware. The software supports a guided user calibration using a few simple measurements that can be carried out at any time with the use of a high speed oscilloscope.

III. Production Testing

In addition to the margin testing software, production testing software is also being developed for the SPLT. This software is complementary to the margin tester software and will allow for acceptable limits of signal degradation to be defined, giving the user full manual control over the aberrations.

The margin testing software is the first port of call in exploring the physical layer of a SpaceWire system as it can quickly establish the operational limits of a UUT. The results of margin tests can then be used to define production testing criteria.

The production testing software is used to evaluate a production unit against a set of static values. A test against these acceptable limits can be run manually or as part of an automated verification process. The results of the test will then be logged in a report.

IV. Results

The SPLT was set up as shown in Fig. 4. A 50 cm SpaceWire cable was attached between ports 1 and 2 of the device. Aberrations were applied to the output of port 2, with their effect captured by an oscilloscope connected to the buffers on port 1.

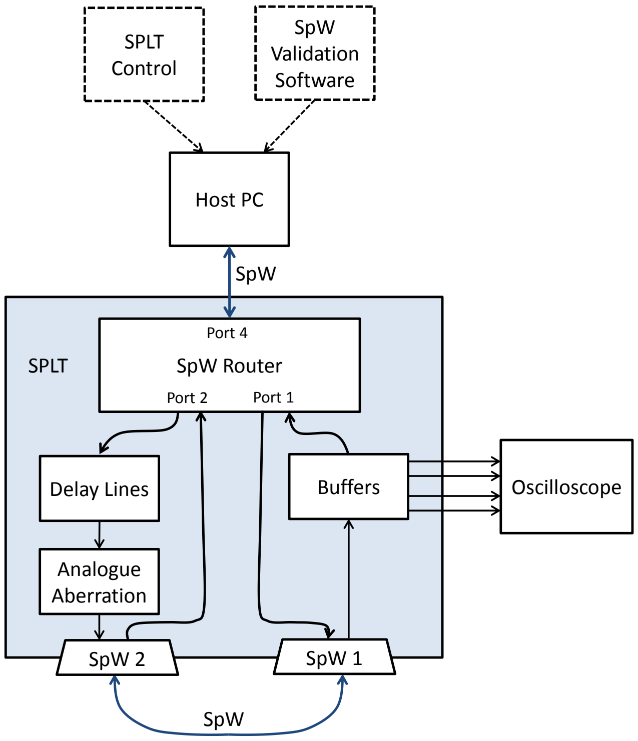


Fig. 4. Setting up the SPLT to test its own SpaceWire ports.

The SpaceWire Router was configured to set port 2 to "start" state, such that the SpaceWire link would automatically attempt to restart following a link disconnect. The tests were then performed with the link running and transmitting Nulls.

In all traces, unless otherwise stated, the timebase is at 10 ns per division, and voltages at 500 mV per division. The traces from top to bottom are D+, D−, S+ and S−

A. Margin testing Data-Strobe skew

The Data-Strobe skew was progressively degraded by progressively delaying the data differential pair from 0ns (no aberration) until link disconnect at 4ns. The results are shown in Fig. 5.

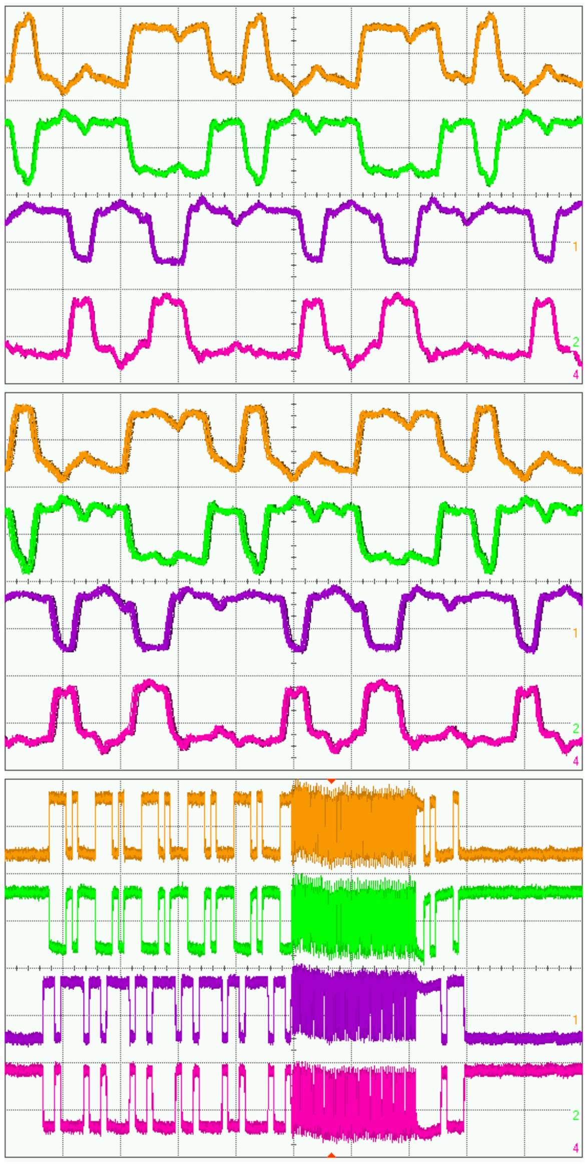


Fig. 5. Data-Strobe skew test at 1, 3 and 4 ns (top to bottom) of delayed data.

The final image in Fig. 5. is zoomed out to show the link trying to re-establish a connection at the 10Mbit/s standard. A connection is briefly made, until a parity error due to the skew aberration causes an exchange of silence before a similar pattern repeats itself.

B. Margin testing in-pair skew

The test in section IV.A was repeated, but this time only the positive end of the Data signal was delayed, whilst the Strobe was unchanged. The results are shown in Fig. 6.

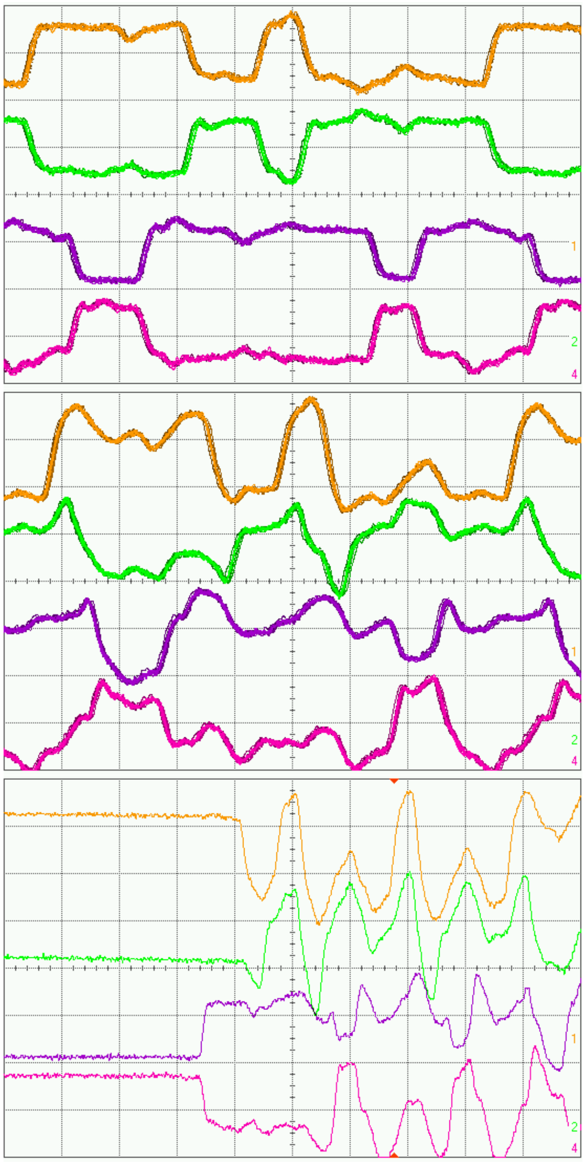


Fig. 6. In-pair skew test at 0, 2 and 4 ns (top to bottom) of delayed Data+.

The link in Fig. 6. eventually disconnects at 4 ns of skew. A single trace is shown where the link speed switches from the 10 Mbit/s startup speed to the 200Mbit/s configured speed. The link soon fails, and reconnection is repeatedly attempted.

C. Margin testing jitter

A triangular pattern was applied to the delay lines such that the stobe delay ramped up to a maximum value in 10 ps steps, before ramping back down to a minimum value in 10 ps steps. This gives a jitter characteristic of a burst of “short” bits, followed by a burst of “long” bits, which are 10 ps shorter and longer than the link transmit rate, respectively. The major effect of this is that the skew rapidly transitions linearly between the two endpoints. The results are shown in Fig. 7.

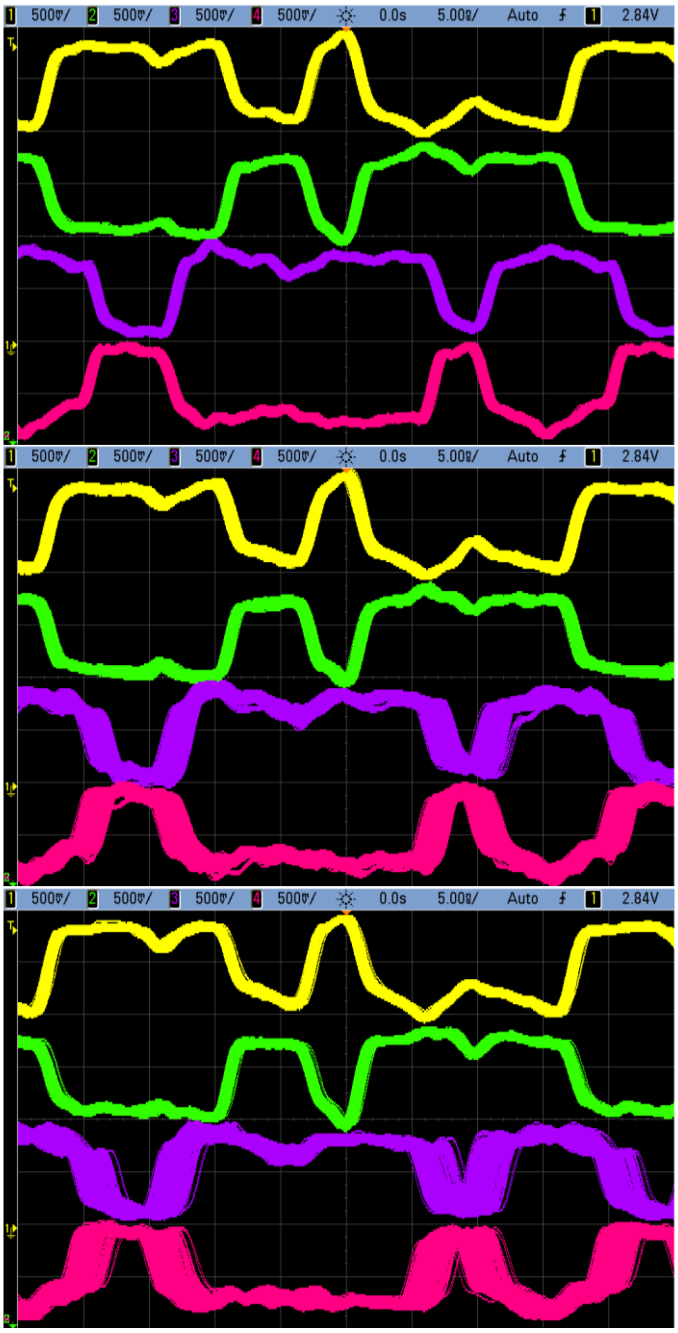


Fig. 7. Jitter triangular pattern on the strobe with 0, 2 and 5 ns of jitter applied from top to bottom.

Fig. 7. Shows the signal’s progressive deterioration as the increasing jitter causes the eye of the signal to narrow. The eye is only just visible at 500 ns as the link is pushed to the edge of its margins. The eye eventually closed to cause the link to disconnect at 6 ns of triangular jitter.

The SPLT was then configured to drive 3 ns of Sawtooth Jitter. In this mode, the Strobe was configured to count up from zero to 3 ns of delay synchronously to the transmit clock.

The Data was configured to a constant skew of 1.5 ns. Traces of this setup are shown in Fig. 8. and Fig. 9.

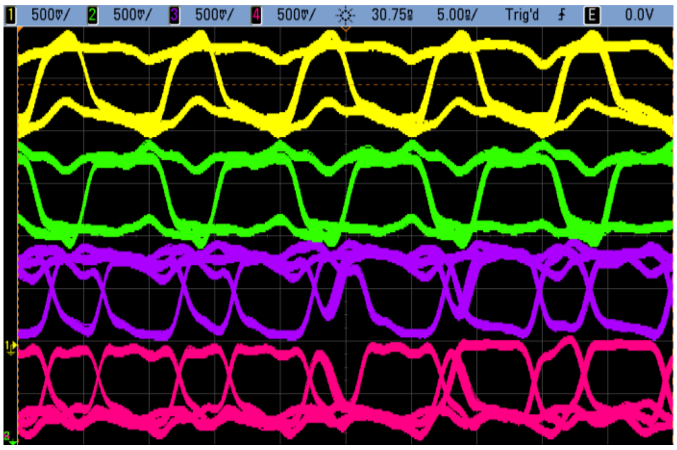


Fig. 8. 3ns Sawtooth Jitter with persistence on.

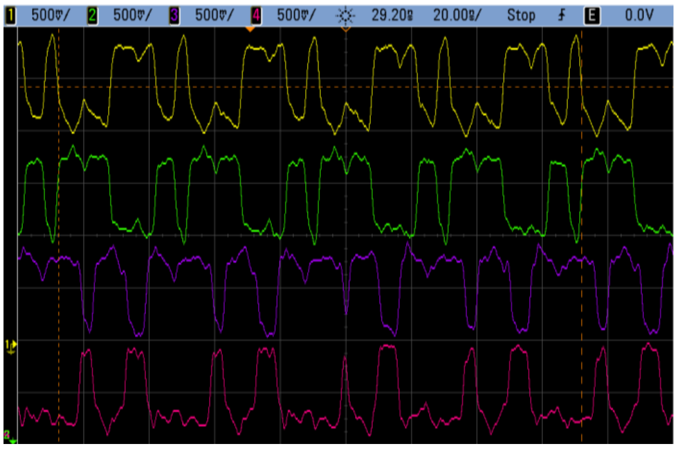


Fig. 9. Single oscilloscope trace of 3ns Sawtooth Jitter.

In Sawtooth jitter mode with the parameters configured in this test, the Strobe moves from 1.5 ns leading the Data to 1.5 ns lagging the Data. It does this by increasing the bit period by 10 ps compared to the Data bit period. When the Strobe reaches this limit, a single bit of period 3ns shorter than the Data bit period is injected to return the Strobe to lead the Data signal by 1.5 ns. A trigger on this short bit period was used to show the shortened bit in the middle of the traces in Fig. 8. and Fig. 9.

The SpaceWire link was seen to periodically disconnect throughout this test every few seconds as the link was pushed to the limit of its margins at 300 ns of sawtooth jitter. Comparison of this value to the 500 ns of triangular jitter that the link was able to sustain in Fig. 7. shows that sawtooth jitter is more detrimental to the link operation than triangular jitter.

D. Margin testing swing

The swing on the Data signal was reduced in steps down to a value of 35 mV, where the link disconnected. The progressively aberrated waveforms are shown in Fig. 10.

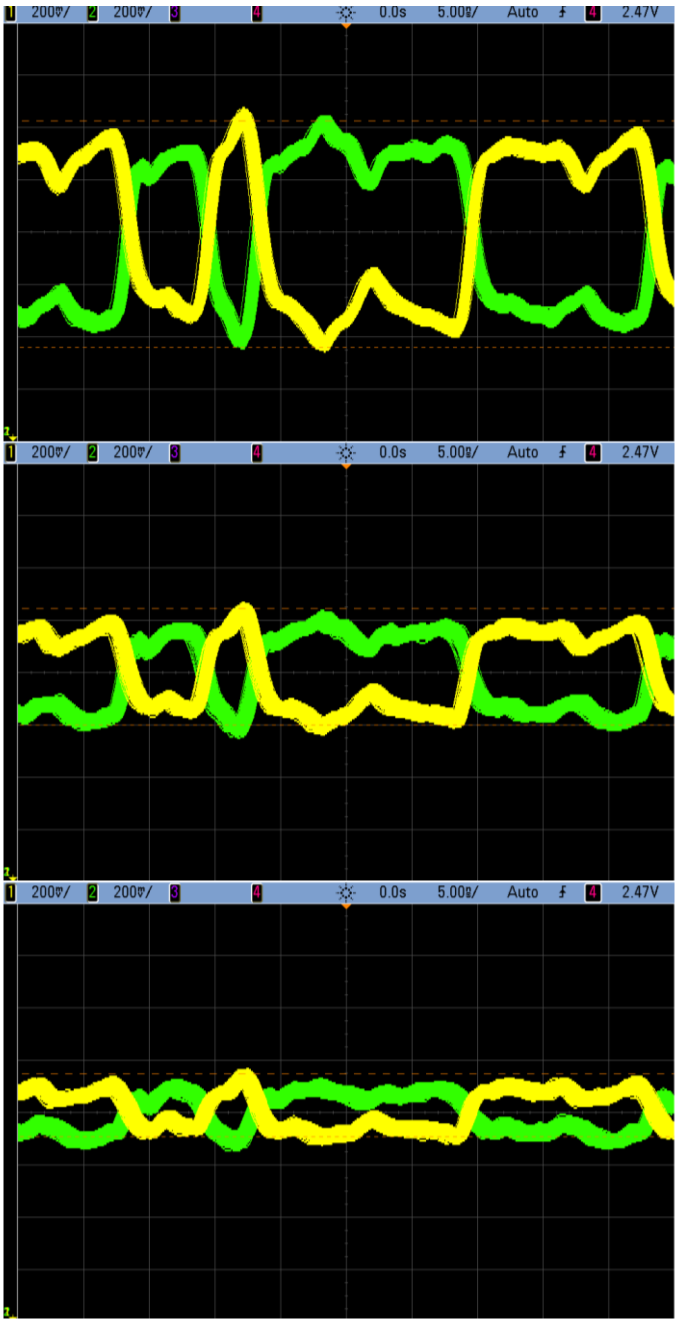


Fig. 10. Reducing swing through 620, 310 and 155 mV from top to bottom.

The eye of the received waveform closes as the swing is reduced. Once the swing is reduced below 150 mV, it becomes very close to the 100 mV minimum swing specified in the LVDS receivers' specifications. Link disconnects start to become increasingly regular as this value is approached.

E. Margin testing offset

The offset of the Data signal was reduced from 2 Volts down to zero volts. Oscilloscope screen captures of this test are shown in Fig. 11.

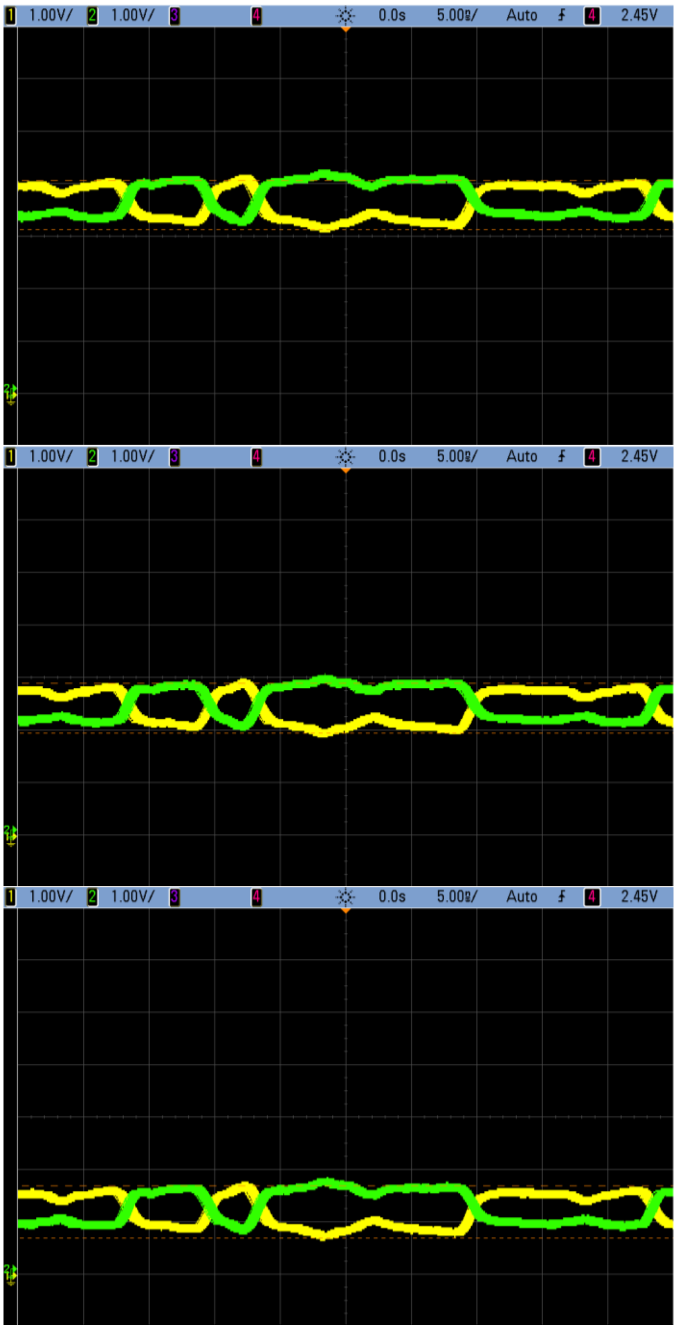


Fig. 11. Reducing the common mode through 1.83, 1.22 and 0.61 Volts.

The link was sustained without any disconnects across this range of tests. The LVDS receivers' specifications stated an input common mode range of 0.3 to 2.35 Volts. This is still a useful test to perform as it confirms the validity of the datasheets' quoted values.

F. Margin testing slew

The SPLT features three different sizes of capacitor which can be independently switched onto each of the drivers to achieve different slew rates depending on the combination of capacitors selected. A test was run with slew loads ranging from 0 pF up to the full 116 pF from all 3 capacitors switched in. For the cable used in this test, this caused the 90% - 10% rise and fall times to increase from approximately 1 ns up to 6 ns. The screenshots of this test are shown in Fig. 12.

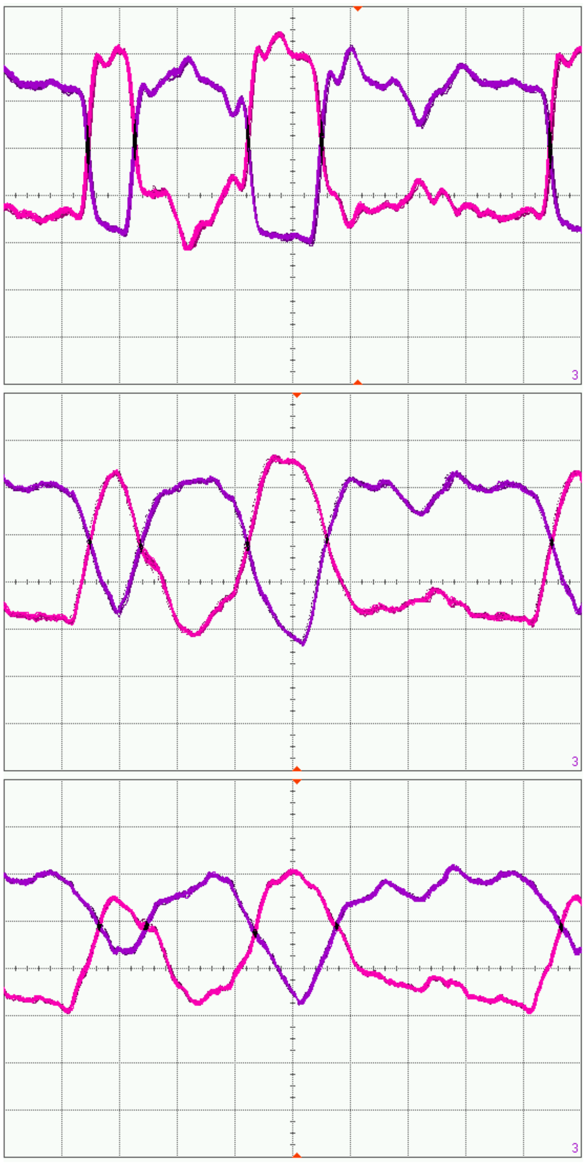


Fig. 12. Increasing the slew capacitance through 0, 48 and 116 pF, shown from top to bottom. The vertical scale is set to 100 mV / division and the horizontal scale to 5 ns / division.

No disconnections occurred during throughout this test. It can be clearly seen from Fig. 12. that, whilst the signal has been progressively degraded at the corners, there is still a clear eye opening allowing for the bit stream to be successfully received.

Slew was then progressively introduced in-pair to the negative end of the Strobe line, whilst the positive end was maintained at minimum slew (all slew capacitors disabled).

A number of oscilloscope screenshots from this test are shown in Fig. 13.

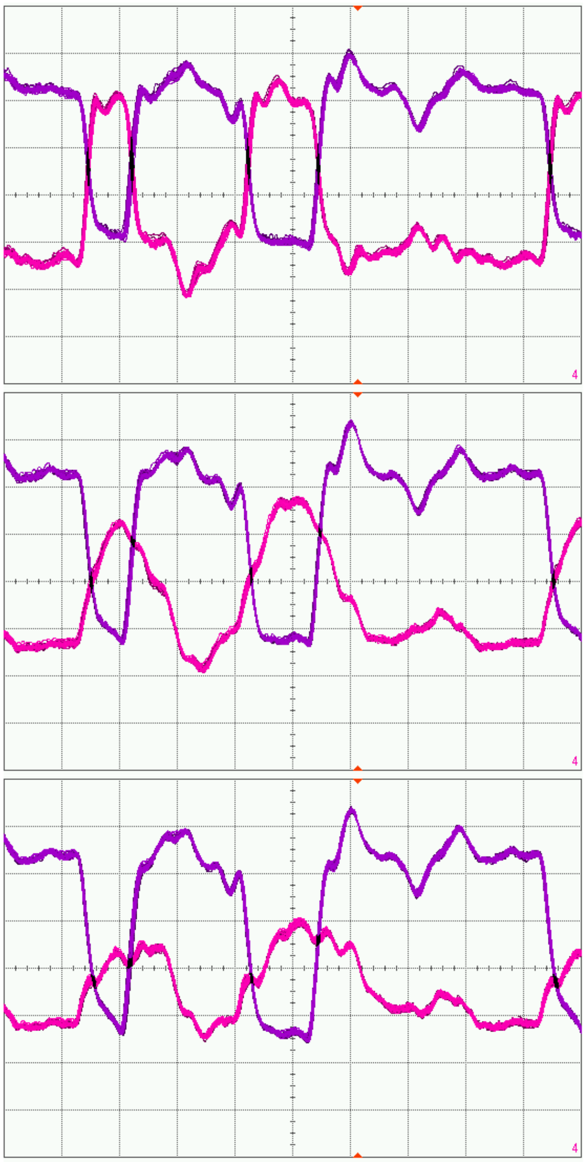


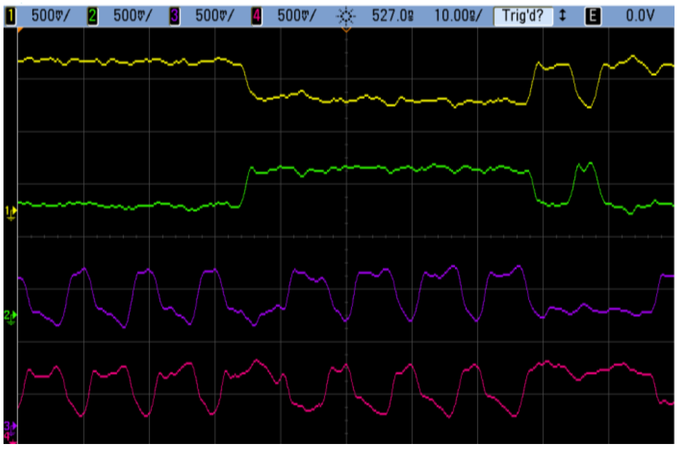
Fig. 13. Increasing the in-pair slew capacitance through 0, 48 and 116 pF, shown from top to bottom. The vertical scale is set to 100 mV / division and the horizontal scale to 5 ns / division.

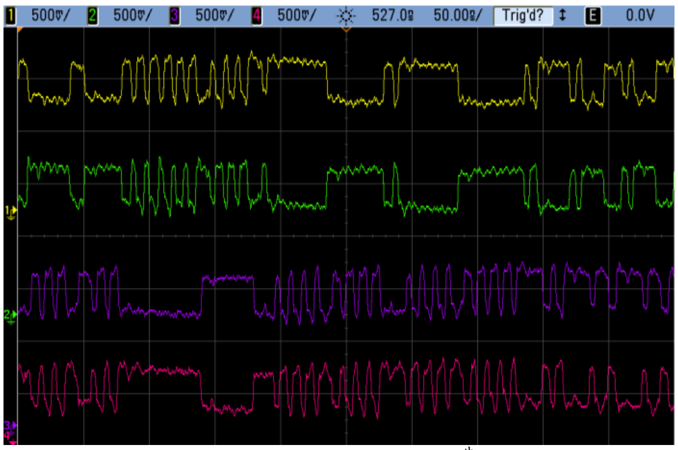
As with the previous slew test, no disconnections occurred at maximum slew settings. The eye of the signal is still open, allowing for the bit stream to be successfully recovered.

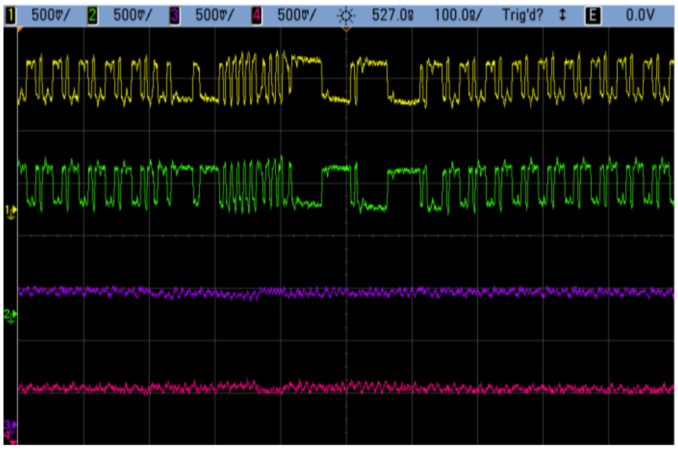
Whilst no disconnects were observed with maximum slew in this test, slew can cause a link to fail when combined with aberrations which would have previously caused no link disconnects.

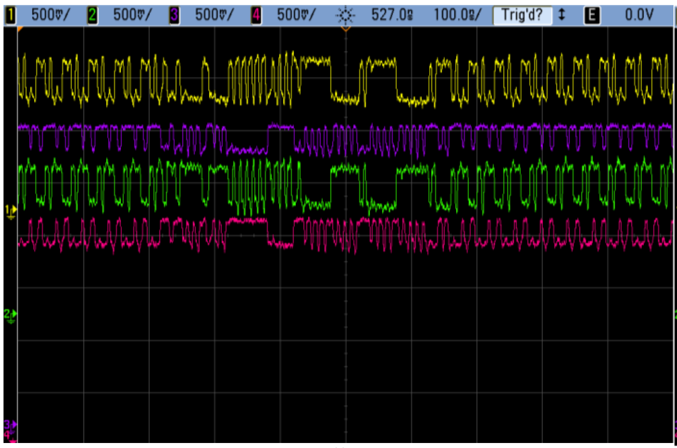
G. Tests with Data

A number of test data packets were transmitted at 200 Mbit/s, with high levels of aberration configured in each test. The following oscilloscope screenshots show the results of these tests.


1 500mV/ 2 500mV/ 3 500mV/ 4 500mV/ 527.0ns 10.00ns/ Trig'd? E 0.0V
Fig. 14. Transmitting a data packet with 3ns skew caused by Strobe delay. Notice the almost simultaneous transition of the Data and Strobe lines occurring three divisions from the centre of the trace.

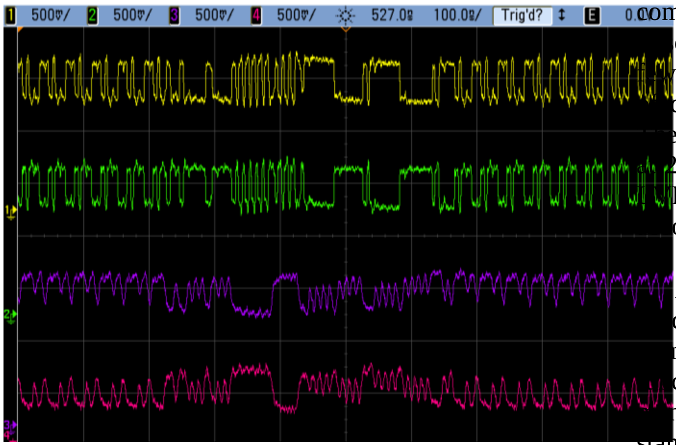

1 500mV/ 2 500mV/ 3 500mV/ 4 500mV/ 527.0ns 50.00ns/ Trig'd? E 0.0V
Fig. 15. Sawtooth Jitter causes 2ns shorter bit every 5th bit on strobe. The strobe bit pattern therefore consists of four bits extended by 0.5 ns and one bit shortened by 2 ns.


1 500mV/ 2 500mV/ 3 500mV/ 4 500mV/ 527.0ns 100.0ns/ Trig'd? E 0.0V
Fig. 16. Transmitting data with the Strobe swing reduced to 75 mV. This link continues to run despite the 100 mV minimum swing specified in the LVDS receiver's datasheet.



1 500mV/ 2 500mV/ 3 500mV/ 4 500mV/ 527.0ns 100.0ns/ Trig'd? E 0.0V

Fig. 17. Transmitting data with the Strobe offset to 2700 mV. Notice how the signal is clipped at such high offsets; the strobe's swing was configured to the same level as the data. The link continues to run, despite the 2350 mV operating limit specified in the datasheet.



1 500mV/ 2 500mV/ 3 500mV/ 4 500mV/ 527.0ns 100.0ns/ Trig'd? E 0.0V

Fig. 18. Transmitting data with full 116 pF of slew

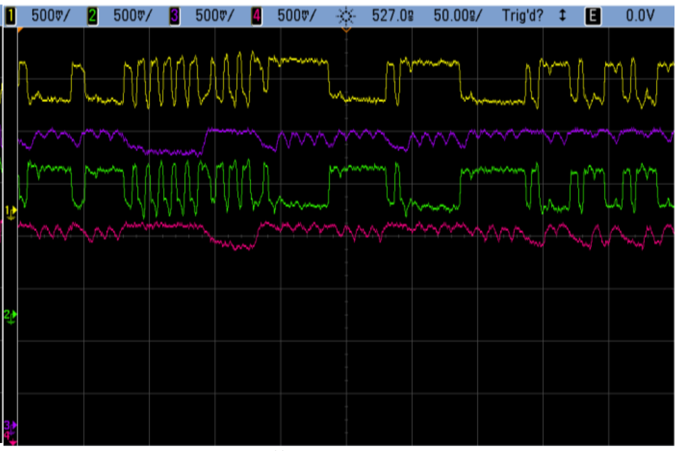
In all of the above tests, the link was left to run for several minutes, an no disconnects were observed.

H. Combined aberrations

Any number of the above aberrations can be combined in any combination to define an envelope in which a link can operate. The following aberrations were configured on the SPLT whilst the link was transmitting data:

• Swing	100 mV
• Offset	2700 mV
• Slew	116 pF

An oscilloscope screenshot of these conditions is shown in Fig. 19.



1 500mV/ 2 500mV/ 3 500mV/ 4 500mV/ 527.0ns 50.00ns/ Trig'd? E 0.0V

Fig. 19. Combining swing, offset and slew aberrations onto the Strobe signal with a SpaceWire link transmitting data.

Fig. 19. shows the oscilloscope measurement of these combined aberrations. The offset is set so high, that the signal omewhat clipped at the top. The link disconnected every seconds under these aberrations; but a significant amount data could still be transmitted between these disconnects. disconnects are largely attributed to the high offset, which 2700 mV, is 350 mV higher than the operating limit of the DS receivers' specification (but still well within the olute maximum!).

A number of combined aberrations could be used for duction testing to validate that a link can operate under a nber of harsh corners of the envelope which a fully ctional production unit could be expected to withstand. A nufacturing defect which might not be detectable under a standard LVDS test could then cause the link to fail when subjected to such an envelope of aberrations from the SPLT.

V. Conclusion

This paper has described the SPLT hardware and introduced the application software that has been developed to simplify the process of physical layer testing.

A number of tests performed in the resuts section have yielded fully operating SpaceWire links; but analysis of the oscilloscope screenshots show how poor the LVDS signal is. This demonstrates how a SpaceWire device can appear to be fully functional when it is powered up and successfully transferring data with other SpaceWire components. Using the Physical Layer Tester's analogue buffers, it is possible to unobtrusively measure the received waveform and validate that a piece of equipment is operating correctly at the Physical layer.

Likewise, when receiving SpaceWire data, a link can seem perfectly functional. The SPLT can be used to subject the SpaceWire link receiver to an envelope of harsh aberrations that should be able to sustain an active link, based on measurements from previous margin testing of equipment. A manufacturing defect, or damage to the unit that does not disrupt a link under standard LVDS parameters could cause the link to fail under these harsh tests.

In Summary, the SPLT is an essential quality control tool for manufacturers of SpaceWire equipment. It is also a powerful validation tool for end users of SpaceWire equipment.

REFERENCES

[1] P. Scott, P. Crawford, S. Parkes and J. Ilstad, "Testing SpaceWire systems across the full range of protocol levels with the SpaceWire Physical Layer Tester", International SpaceWire Conference 2011, San Antonio, 8th – 10th November 2011.

Modeling and Verification of SpaceWire Interface by Timed Automata

SpaceWire Test and Verification, Long Paper

Ping Luo, Yong Guan, Xiaojuan Li, Rui Wang,
College of Information Engineering,
Capital Normal University,
Beijing, 100048, China
{luoping0718@126.com, guanyong@mail.cnu.edu.cn,
lixj66@gmail.com, rwang04@gmail.com}

Jie Zhang
College of Information Science & Technology, Beijing University of Chemical Technology,
Beijing, 100029, China
jzhang@mail.buct.edu.cn
Xiaoyu Song
Dept.of ECE, Portland State University Portland, Oregon, US

***Abstract*—SpaceWire is a serial link Standard for on-board network applications. A design of the standard at the exchange level was implemented in the previous work by our research group. Due to the special demand of higher reliability, it is necessary to test and verify its correctness. This paper presents a formal modeling and verification method of the SpaceWire implementation resorting to Uppaal, a model checker based on timed automata theory. In this work, we focus on the connection process across a link, which is one of the primary missions of the exchange level. We extract the timed automata models for the two ends of the link. Each end of the link is formalized as a network of timed automata including the LocalHost, the Timer, the StateMachine, the Transmitter and the Receiver. In the paper, the SpaceWire safety requirements are specified as computational tree logic (CTL) properties. We find out some detail errors of the original implementation by checking if the link interface is deadlock-free or not. It is also verified that the link connection can be made successfully conforming to the specification. The experimental results demonstrate the effectiveness of the approach presented here.**

***Key Words*—SpaceWire, Formal Verification, Model Checking, Timed Automata, Uppaal.**

I. Introduction

SpaceWire [1] is a serial link standard for onboard network applications, which is put forward by the European Space Agency (ESC for short) in 2003. It provides a full-duplex, bidirectional, serial point-to-point, high speed data link. Data can be transferred at a different data signaling rate in the both directions, which ranges from 2 to 400 Mb/s [2]. Due to the broad prospects of space applications, it has been attracting more and more attention from the space agencies and companies all over the world. Lots of work has been done and variable designs and implementations come forward. Considering the special demand of higher reliability and the possible misunderstanding of designers or programmers, it is quite necessary to test and verify the correctness of SpaceWire designs and implementations.

Formal verification [3] shows great prospects, compared to the incompletion of traditional verification methods, like test and simulation. It is done by proving a formal proof on an abstract mathematical model of the target system. Many powerful mathematical objects are used to model systems, such as finite state machine, Petri nets, timed automata, hybrid automata, and Hoare logic and so on [4]. Model checking [5] [6], one of the formal verification approaches, is put forward by Edmund Melson Clarke, Ernest Allen Emerson, etc., who together won the 2007 Turing Award. The main idea of model checking is: through extracting the model of the target system, to exhaustively and automatically check whether it meets some given specifications. It has been increasingly applied to both hardware and software. Many tools are developed for model checking, like SPIN [7], SMV [8], Uppaal [9], etc. Uppaal will be introduced here, which is jointly developed by Uppsala University and Aalborg University. It is a model checker based on the theory of timed automata. In view of its special advantage to models involved with time, it suits modeling and verifications for real-time systems. The timed automata models of a system can be established in the editor of Uppaal GUI. A simulator can help to check whether the models meet the expectation. At last, we can verify in the verifier the requirement specifications, which are expressed in computational tree logic (CTL for short). If the given result of a property turns out to be satisfied, it indicates that the system meets that specification. Otherwise, a counterexample will be fed back [10].

This paper proposes an approach of model checking to verify our design of the SpaceWire exchange level [11], resorting to the tool Uppaal. The timed automata theory in Uppaal is introduced in section II. Section III makes a brief introduction to SpaceWire link. Its models of timed automata are illustrated in section IV, and verification is done in section V. Conclusion comes at last.

II. Timed Automata in Uppaal

Timed automata theory is first introduced by Rajeev Alur and David L. Dill in 1990 [12]. A timed automaton is a finite state Buchi automaton extended with a set of real value variables in Uppaal.

To define its syntax and semantics, the following notations are used: C is a set of clocks and F(C) is the set of conjunctions over simple conditions of the form $x \bowtie n$ or $x - y \bowtie n$, where $x, y \in C, c \in N$ *and* $\bowtie \in \{<, \leq, =, \geq, >\}$. Let R^C be the set of all clock valuations, where a clock valuation is a function $v: C \rightarrow R_{\geq 0}$. Let $v_0(x) = 0$ for all $x \in C$. We adopt the definition of timed automaton in [10][13].

Definition 1 (Timed Automaton)

A timed automaton is a six-tuple $< L, l_0, C, A, E, I >$, where

- L is a set of locations,
- $l_0 \in L$ is the initial location,
- C is the set of clock variables,
- A is a set of actions and co-actions,
- $E \subseteq L \times A \times F(C) \times 2^C \times L$ is a set of edges between locations with an action, a guard and a set of clocks to be reset,
- $I: L \rightarrow B(C)$ assigns invariants to locations.

Definition 2 (Semantics of Timed Automaton)

A timed automaton $\mathcal{A} =< L, l_0, C, A, E, I >$ can be semantically defined as a labeled transition system $< S, s_0, \rightarrow >$, where

- $S \subseteq L \times R^C$ is the set of states,
- $s_0 = (l_0, v_0)$ is the initial state,
- $\rightarrow \subseteq S \times (R_{\geq 0} \cup A) \times S$ is the transition relation such that:
  - $(l, v) \overset{d}{\rightarrow} (l, v + d), for\ \forall d' \in R_{\geq 0},$ if $0 \leq d' \leq d, v + d'$ satisfies $I(l)$
  - $(l, v) \overset{a}{\rightarrow} (l', v'),$ if $\exists e = (l, a, g, r, l') \in E, s.t. v \in g, v' = [r \mapsto 0]v,$ and $v' \in I(l)$, where $[r \mapsto 0]v$ denotes the clock valuation which maps each clock in r to 0 and agrees with v over $C \backslash r$

A network of timed automata is often comprised of several timed automata over a common set of clocks and actions. These timed automata synchronize each other through pairs of channels of the form a! and a?.

III. SPACEWIRE INTERFACE

In a SpaceWire network, units of nodes and routers are interconnected through bidirectional, full-duplex, point-to-point data links. At the exchange level of SpaceWire, the interfaces of both ends across the link are designed for making a connection and managing the flow of data. Our design includes modules of Controller, Transmitter, Receiver, Timer, CreditCounter, BaudrateCounter, Recovery, and ErrorNotification. In this paper, Controller(StateMachine), Transmitter, Receiver, Timer, only the primary ones, will be modeled and analyzed, purposively and for simplicity. The following Fig. 1 displays the simplified SpaceWire link interface block diagram.

The StateMachine, as the center of control, controls the overall operation of the interface. It is designed as a finite state machine consisting of several states. After a system reset signal, it starts to work. It controls the affair states of Transmitter and Receiver via signals, Reset, enableTx, and enableRx. When assterting signals of sendNULL, sendFCT, sendTimeCode, sendNChar at some specific moments, it directs the Transmitter to send corresponding characters. Also the Receiver will inform the StateMachine when receiving a(an) NULL/FCT/TimeCode/NChar. Timer keeps time for StateMachine and provides two periodic signals of After64 and After128.

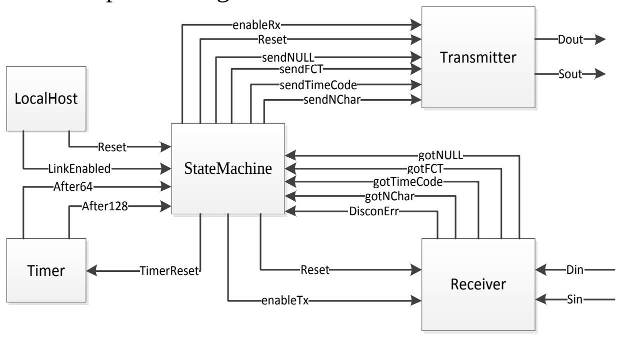


Fig. 1. A simple example of link interface block diagram

Before successfully establishing a connection across a link, both ends of the link will go through ErrorReset state, ErrorWait state, Ready state, Started state, Connecting state, until the state Run. A typical example of initialization sequence is illustrated in Fig. 2. After a delay of 6.4us in ErrorReset and a 12.8us one, the state machine of one end goes into the Started state. Entering that state, its Transmitter becomes enabled and begins to send NULLs to the other end. It will keep sending until the moment an FCT arrives, when it then moves into the Connecting state. In that state, FCTs are allowed to be transmitted. Once an FCT is received in the state Connecting, the state machine shall enter the state Run. So far a NULL handshake and an FCT one have taken place, which make sure of the bidirectional communication. The link initialization succeeds. Further details will be presented in the timed automaton model of StateMachine below.

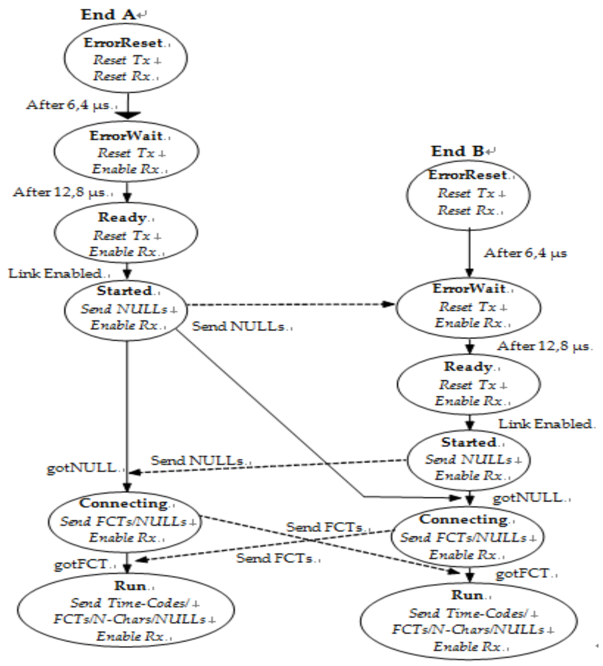


Fig. 2. An example of typical initialization sequence[cite]

IV. Models of Timed Automata

Let id, an integer variable, be the identity of each end. All the models in this paper shall be set the parameter id in order to be identified. We can suppose that end A is marked with id==0 and end B with id==1. The unit of time of 100ns-long is used here. Thus the delays of 6.4us and 12.8us can equal to 64 time units and 128 ones, respectively. Since the Transmitter shall operate at a data signaling rate of (10±1) Mb/s, it takes one time unit to send a bit.

In accordance with the VHDL codes of implementation, the models of timed automata are extracted here including LocalHost, StateMachine, Transmitter, Receiver, and Timer, as shown below.

A. LocalHost

The simplest case of the local system is modeled here in Fig. 3. The system may send a Reset signal to start the link any time. Once a Reset signal beamed, an clock variable x starts to keep track of time, performed by a function TimeStart() as TABLE 1 shows. Also a LinkEnabled signal is assumed to be asserted if needed.

TABLE I. FUNCTIONS DEFINITION

```
//LocalHost
void TimeStart(){
// bool s, true only if at least one end has been reset;
   if(!s)  x=0;
 }
//global declaration
void reset(int id){
   firstNULL_gone[id]=0;
   firstNULL_received[id]=0;
   numberofFCT[id]=0;
}
```

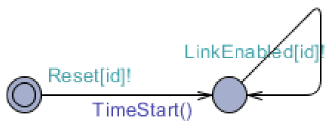


Fig. 3. Timed Automaton of LocalHost

B. Timer

Timer is a quite significant module for SpaceWire. It provides a timer of 64 and one of 128, which control the state transition of link interface, make sure not to wait forever in any state. Figure 4 illustrates the implementation. In the timed automaton, the initial location has an invariant, clk<=64, where clk is a local clock. A bool variable trigger denotes the timer to be triggered. Replying the channels Reset! and TimerReset!, the clock clk is reset to 0, and trigger is updated to be false, signifying the timer of 64 is ready to start. After time of 64 is up, it emits a channel After64! and updates clk to 0 and trigger to 1. 64 time units later, a signal of After128 will be sent out. After that, a new cycle runs.

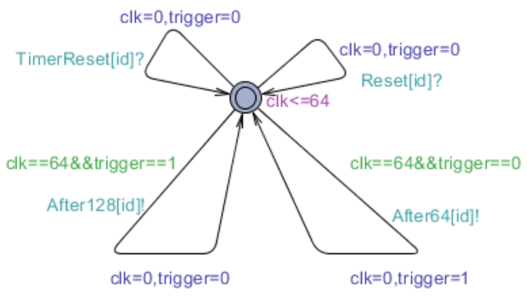


Fig. 4. Timed Automaton of Timer

C. StateMachine

The Fig. 4 illustrates the timed automaton for the StateMachine at a link end. Responding to Reset! from the LocalHost, it stays in ErrorReset for 64 time units. When the delay elapses, it unconditionally enters ErrorWait, with enableRx! sent to Receiver and TimerReset to Timer. When firstNULL_received is true, the StateMachine will go back to ErrorWait from ErrorWait, if any channel, DisconErr?, gotFCT?, gotTimeCode?, gotNChar?, is activated by Receiver. Otherwise, it will enter the state Ready after a delay of 128. As the state ErrorWait does, channels of DisconErr?, gotFCT?, gotTimeCode? and gotNChar? can lead to reset the link. If the link is enabled, it moves to the Started location and commands to keep sending NULLs since the Transmitter is enabled here. Also the Timer gets reset for a trigger of 128 time units. The StateMachine is expecting the reception a NULL, whereas FCTs or TimeCodes or NChars is not allowed to be received unless the first NULL has arrived in Started state. Having received a NULL and with at least a NULL gone, the state Connecting will come, where FCTs and NULLs are both permitted to be transmitted. Once receiving an FCT, the StateMachine leaves for the Run location and the link initialization is made. Normal operations are allowed across the link, unless an error occurs or link is disabled. The characters that are allowed to be transmitted in every state have different priority levels, as TimeCode > FCT > NChar > NULL. Characters of higher priority will be transmitted first. We declare for the priorities: chan priority sendNULL<sendNChar <sendFCT<sendTimeCode.

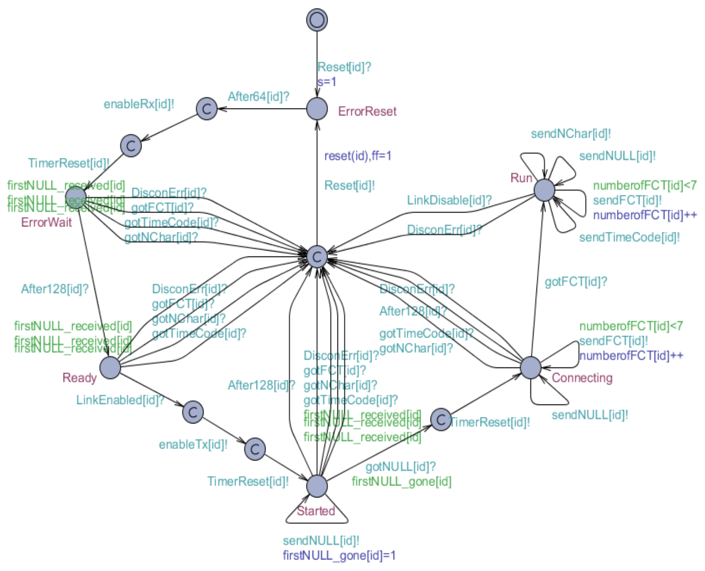


Fig. 5. Timed Automaton of StateMachine

D. Transmitter

We have a high level model of Transmitter as shown in Fig. 6, owing to the specific task of verification discussed in this paper. There are 6 locations: an initial one, an urgent one, and four for sending a type of characters, like TimeCodes, NULLs, FCTs, and NChars. If enabled, the Transmitter sends corresponding characters submitted to the StateMachine's instruction. For instance, while a synchronization channel sendNULL comes, it enters the load_NULL location and commences to send a NULL. A NULL control code contains a ESC of 4 bits and a following FCT of 4bits. Thus a local clock clk is involved. The invariant of clk<=8 is added to the location load_NULL, which indicates that time spent sending a NULL. If clk reaches to 8, it sends a synchronization signal receiveNULL to Receiver at the other end of the link. Here, the propagation of the SpaceWire link is ignored. In the similar way, FCTs/TimeCodes/NChar are transmitted. And that will separately take 4/14/10 time units. A Reset signal can be valid at any location.

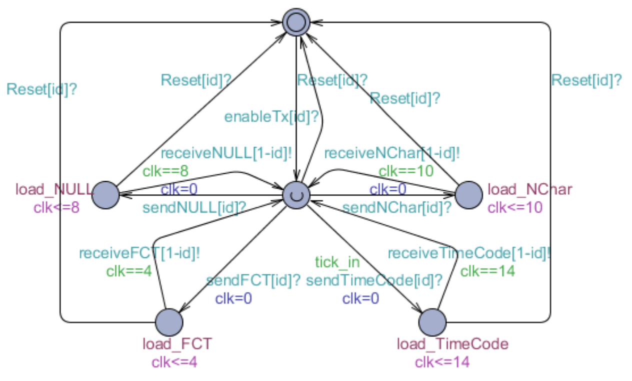


Fig. 6. Timed Automaton of Transmitter

E. Receiver

The Receiver is responsible for decoding the DS signals and passing a sequence of NChars received on to the host system. It also informs the StateMachine of the receptions of characters. As shown in the Fig. 7, there are three prime locations(not committed ones): an initial one, two enabled with any bit received or not, en_nobit, en_bit. In the location en_nobit, a disconnection error is not activated yet, while it will be detected in the location en_bit. On receiving the first bit, it switches into the en_bit location. The signal channel is not introduced alone here. Owing to the need of disconnection error detection, we combine the channel and receiver into a high level extracted model. A particular channel is involved, with three situations: 1) codes may be received normally; 2) codes may be received incompletely; 3) the whole character may get lost in the channel. Due to that three cases, when synchronized by receiveNULL? (receiveFCT?, receiveNChar?, receiveTimeCode?) in the location en_nobit, it may switch into en_bit emitting an edge with gotNULL!, or not because of the part reception of codes. It may also do nothing at all since characters are lost. So does the location en_bit. In addition, it has an invariant clk<9, which is introduced for the detection of disconnection error. If clk>8 (roughly 850ns specified in the Standard) becomes true, a disconnection error occurs, of which will be informed StateMachine.

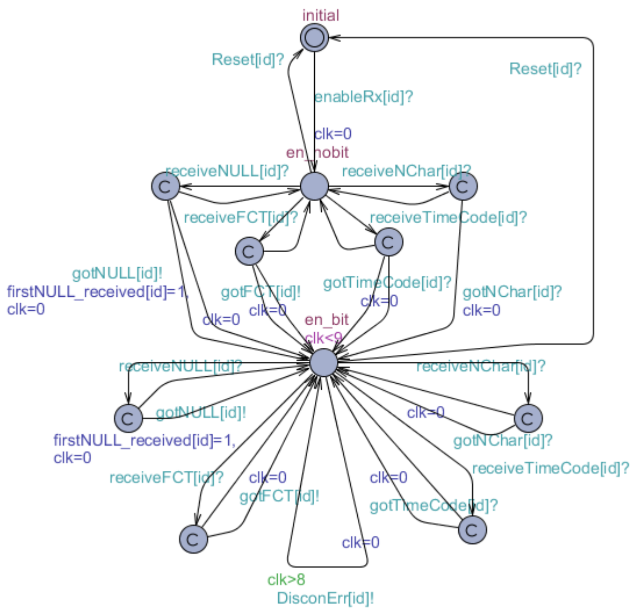


Fig. 7. Timed Automaton of Receiver

V. VERIFICATION

Uppaal uses a simplified version of CTL to express the requirement specification. Path formulae and state formulae are supported.

TABLE II. PROPERTIES IN UPPAAL

property	expression	meaning
Reachability	E<> p	p is satisfied in reachable states
Safety	A[] p , E[] p	something bad never happens
Liveness	A<> p, p --> q	p is eventually satisfied

In this paper, a network of timed automata is established, consisting of such 5 ones belonging to end A as LocalHost[0], Timer[0], StateMachine[0], Transmitter[0], Receiver[0], and 5 to end B, such as LocalHost[1], Timer[1], StateMachine[1], Transmitter[1], Receiver[1]. And several properties are verified below.

1) The system should be deadlock-free.

A[] not deadlock

It turns out that the property is not satisfied. The verifyta in Uppaal presents a counterexample illustrated in Fig. 8. It is shown that end A began to send a NULL entering the Start state while end B still stayed in the ErrorWait state. However, end B didn't receive the intact character. The Receiver at end B transited to the en_bit location where detection of disconnection error was activated. After more than 8 time units, a disconnection error should have occurred.

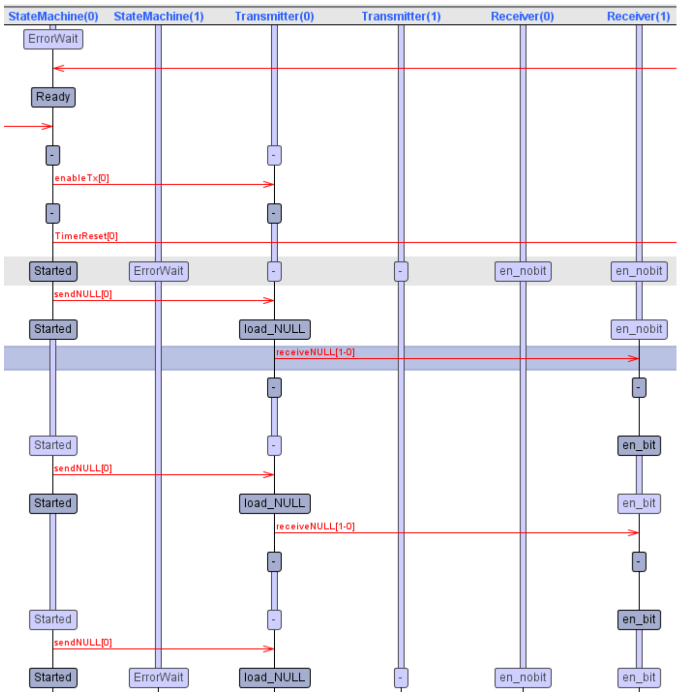


Fig. 8. A counterexample for Property 1).

From the timed automaton of StateMachine in Fig. 5, we can see the edge labeled with the channel DisconErr[1]? is not enabled since the guard firstNULL_received[1] didn't satisfy.

TABLE III shows a small part of original VHDL codes of the StateMachine, where the state transition is described in the ErrorWait state.

TABLE III. PART OF CODES IN VHDL OF STATEMACHINE

```
    CASE CurrentState IS
                              ⋮
    WHEN ErrorWait => IF After128 = '1' THEN
        NextState <= Ready;
    ELSE
        NextState <= ErrorWait;
        IF FirstNULLreceived_internal = '1' THEN
          IF RX_Error = '1' OR RX_GotSomethingWrong =
'1' OR DisconnectionError = '1' THEN
             NextState <= ErrorReset;
          END IF;
        END IF;
    END IF;
                              ⋮
    END CASE;
```

Referring to clause 8.5.2.3 e in [], however, it is said below:

If, while in the *ErrorWait* state, a disconnection error is detected, or if after the gotNULL condition is set, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move back to the *ErrorReset* state.

We can learn that the error results from blocking the disconnection error with the gotNULL condition in the original design. Hence a revised version is displayed as follows. Furthermore, a corresponding timed automaton is modeled to be checked.

TABLE IV. CODES REVISED

```
    CASE CurrentState IS
                              ⋮
    WHEN ErrorWait => IF After128 = '1' THEN
        NextState <= Ready;
    ELSE
        NextState <= ErrorWait;
        IF DisconnectionError = '1' OR
           (FirstNULLreceived_internal = '1' AND (RX_Error
           = '1' OR RX_GotSomethingWrong = '1')) THEN
              NextState <= ErrorReset;
        END IF;
    END IF;
                              ⋮
    END CASE;
```

The same error also occurs in the Ready state and the Started state. Similar modifications are made to the timed automaton model of StateMachine. And the deadlock is resolved in these three states.

2) The link connection can be made successfully.

E<>StateMachine(0).Run&&StateMachine(1).Run

That property is one of the most significant requirements for SpaceWire link. Both ends enter the Run state, which means that link connection has been made and both link characters and normal characters can flow freely in both directions across the link. The property is checked to be satisfied. Uppaal also generates the path to it, which is same as the initialization sequence in Fig. 2. In addition, the time cost in that given path ranges from 204 to 268. The minimum time of 204 can be the time taken to transfer a NULL and an FCT, besides the delay of 64 in the ErrorReset state and 128 in the ErrorWait state during the whole initialization. In fact, 204 is the most perfect minimum time, the ideal one, due to the ignorance of all the delays occurring in the program and in the SpaceWire channel. Even though the transmitters at both ends transmit a NULL of 8 bits at the same time, they will send the second NULL before informed of the reception of a NULL from each other. Thus two NULLs and a FCT at least shall be transferred during initialization in reality, which indicates that at least 212 time units (21.2 us equaled) will be taken.

VI. Conclusion

This paper has proposed an approach of model checking to verify our design of the SpaceWire link interface, resorting to the model checker Uppaal. Both end A and end B across the SpaceWire link are modeled as a network of timed automata. Each end is comprised of LocalHost, Timer, StateMachine, Transmitter and Receiver. By verifying the models extracted from the VHDL codes, a few potential errors are detected, which are caused by the designer's misunderstanding about the constraint of a disconnection error. The results of verification demonstrate the effectiveness of this approach.

Acknowledgment

We would like to acknowledge our group for the work done before and thank Professor S. Z. Jin, Z. P. Shi, and C. N. Zhao for their instructions. This research was supported by the BJNSF (4122017), Beijing Education Science and technology development project (KZ201210028036), ISTCP (2011DFG1-3000), and International Science and technology cooperation project (2010DFB10930).

References

[1] http://www.spacewire.esa.int/content/Home/HomeIntro.php

[2] European Cooperation for Space Standardization, “SpaceWire-Links, nodes, routers and networks”, ECSS-E-ST-50-12C, July 2008, available from http://www.ecss.nl/.

[3] D. Dill, S Tasiran, “Simulation meets formal verification”, slides from a presentation at ICCAD, 1999.

[4] http://en.wikipedia.org/wiki/Formal_verification

[5] E. M. Clarke, O. Grumberg, and D. Peled, “Model Checking”, MIT Press, 2000

[6] Edmund M. Clarke, “The Birth of Model Checking”, Lecture Notes in Computer Science, 2008, Volume 5000/2008.

[7] http://spinroot.com/spin/whatispin.html

[8] A. Cimatti, E. Clarke, F. Giunchiglia, “NUSMV: a new symbolic model checker”, International Journal on Software Tools for Technology Transfer, 2000, Volume 2, Issue 4, pp 410-425.

[9] http://www.uppaal.org/

[10] Gerd Behrmann, Alexandre David, and Kim G. Larsen, “A tutorial on Uppaal”, In proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT'04). LNCS 3185.

[11] M. N. Wan, “Reseach and implementation of a high speed bus technology in rugged surrounding” (in Chinese),Master Thesis, Capital Normal University, Beijing, 2007.

[12] R. Alur and D. L. Dill. “A theory of timed Automata”. Theoretical Computer Science 126(2):183-235, 1994

[13] Johan Bengtsson and Wang Yi, “Timed Automata: semantics, algorithms and tools”, In Lecture Notes on Concurrency and Petri Nets. W. Reisig and G. Rozenberg (eds.), LNCS 3098, Springer-Verlag, 2004.

[14] C. McClements, S.M. Parkes, and A. Leon, “The SpaceWire CODEC”, International SpaceWire Seminar, ESTEC Noordwijk, The Netherlands, November 2003

[15] Christel Baier and Joost-Pieter Katoen, “Principles of Model Checking”, London, England: The MIT Press .2008

Onboard Equipment & Software (Long)

High Processing Power Digital Signal Processor with SpaceWire and SpaceFibre Interfaces

SpaceWire Missions and Applications, Short Paper

Bruce Yu[1], Steve Parkes[2], John Franklin[3], Chris McClements[2], Pete Scott[1] and David Dillon[1]

[1]*STAR-Dundee, STAR House, 166 Nethergate, Dundee, DD1 4EE, Scotland, UK*
[2]*Space Technology Centre, 166 Nethergate, University of Dundee, Dundee, DD1 4EE, Scotland, UK*
[3]*Astrium UK, Stevenage, UK*

Email: bruce.yu@star-dundee.com

***Abstract*—There is a need for high-performance digital signal processing systems for spacecraft applications. The problem is that commercial DSP processors are not radiation tolerant and even when they can tolerate a reasonable total radiation dose, they are subject to single event upsets (SEUs). STAR-Dundee is working with Astrium to provide a solution to this problem. A commercial DSP processor is controlled by a radiation tolerant FPGA to detect and recover from SEUs. Operating in a dual redundant configuration the High Processing Power Digital Signal Processor (HPPDSP) unit is able to meet demanding signal processing applications in support of space missions. To configure and control the HPPDSP and to get data in and out of the DSP signal processor a combination of SpaceWire [1] and SpaceFibre [2] interfaces are provided.**

This paper introduces the HPPDSP, outline its overall architecture, and describe the SpaceWire and SpaceFibre interfaces provided.

Index Terms*—*SpaceWire, SpaceFibre, digital signal processing, spacecraft onboard processing

I. Introduction

Commercial DSP processors generally have high processing power which is needed by future Space missions that require very large data processing throughput.

The HPPDSP unit contains a commercial DSP processor connected to memory and an FPGA that provides fault detection and memory management services, and all the input/output functions for the unit. Three SpaceFibre links are provided along with two SpaceWire interfaces. General purpose input/output signals, status display, high-speed ADC, and an interface to a boot FLASH PROM are also provided.

The SpaceFibre interfaces provide the high-speed data input/output to the DSP processor. A DMA controller is provided to read and write data directly to processor memory. Each SpaceFibre interface has multiple virtual channels to support various independent data streams. One virtual channel on each SpaceFibre port is connected to an RMAP target to support configuration and control of the HPPDSP and sharing of critical information about detected faults.

The two external SpaceWire interfaces are connected to an internal SpaceWire router which has two internal ports dedicated to data transfer and one acting as an RMAP target providing similar capability to the RMAP targets attached to SpaceFibre.

Configured in dual redundancy, two HPPDSP units are paired to work together, one as prime and the other one as redundant. The pair consists of two identical copies of the hardware. When the prime unit fails, the redundant unit can take over.

II. Block Diagram

The block diagram of a HPPDSP unit is shown in Fig. 1.

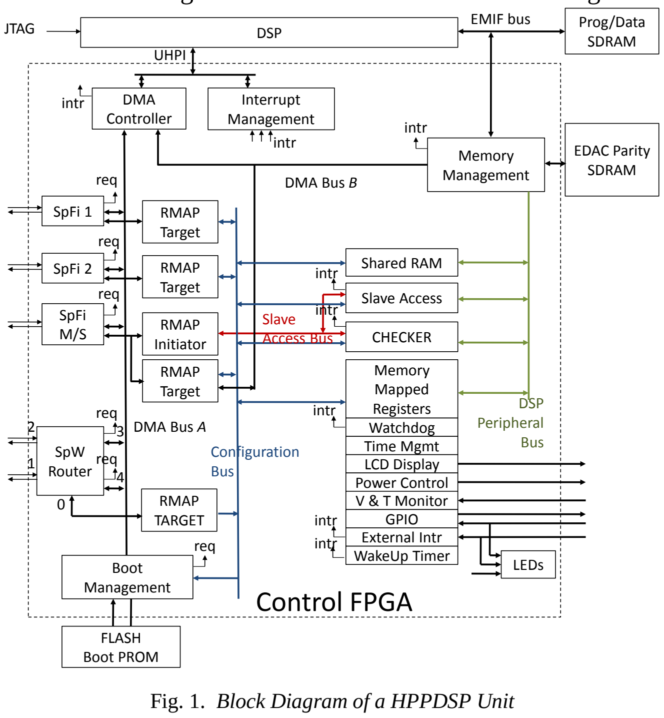


Fig. 1. *Block Diagram of a HPPDSP Unit*

The commercial DSP processor is TI TMS320C6727B [3], which is Texas Instruments' high-performance 32-/64-bit floating-point digital signal processors. It has on-chip RAM and ROM as unified program/data memory, and for external memory it has External Memory Interface (EMIF) which supports a single bank of SDRAM and a single bank of asynchronous memory. The Universal Host-Port Interface (UHPI) is a parallel interface through which an external host, i.e. Control FPGA, can access memories on the DSP. The Control FPGA is a Virtex-4 device.

The DSP can boot either directly from a FLASH-based boot PROM, or over a SpaceWire/SpaceFibre interface accessing other resources on a network. The PROM stores the boot and DSP program data, which can be uploaded from a SpaceWire/SpaceFibre network. A simple Error Detection and Correction (EDAC) technique is utilised to protect data in the PROM. These functionalities are covered by the Boot Management module.

For fast access to program and data, a 32-bit wide large SDRAM memory block is attached to the EMIF interface. An EDAC function is also included, inside Memory Management module, to protect data integrity in the SDRAM memory, which is susceptible to SEU events. The Memory Management also controls which SDRAM regions are allowed for a task to access. The Memory Management module has control over the DMA Bus *B*, from which it can access DSP memory via a DMA controller. It also can access the DSP peripheral Bus, which allows the DSP processor to access various memory mapped registers, along with Slave Access and Checker modules. The Slave Access and Checker Modules are used to exchange information and share memory data between the primary HPPDSP unit and the redundant HPPDSP unit when necessary. Both the Slave Access and Checker modules have access to an RMAP Initiator attached to SpaceFibre Master/Slave interface, so can start a RMAP transaction to the other unit of the Master/Slave pair.

SpaceFibre interface 1 and SpaceFibre interface 2, each have four Virtual Channels (VCs). VC0, connected to a RMAP Target accessing the Configuration Bus, is used to configure/control all modules attached to this Bus, which includes configuring the SpFi and SpW operating parameters. The rest of VCs, from VC1 to VC3, are connected to DMA Bus *A* for DMA data in-to/out-of DSP memory via the DMA controller. These two SpaceFibre interfaces can be configured to work as a prime/redundant pair to achieve dual redundancy.

The SpaceFibre Master/Slave interface has eight VCs. VC0 is used for configuration/control purposes. The rest of the VCs, from VC1 to VC7, are connected to DMA Bus *A* for sending a copy of any incoming IO data stream to the slave HPPDSP unit.

All these SpaceFibre interfaces use STAR-Dundee SpaceFibre Codec IP, which has direct interface to connect with an external serialiser/de-serialiser (SerDes) device, i.e. TI TLK2711[4] in this design.

There is a five port SpaceWire Router on the Control FPGA, with two external SpaceWire ports and three internal ports. Two of the internal ports are connected to DMA Bus *A* for DMA data in-to/out-of DSP memory, and the other internal port is connected to an RMAP Target accessing the Configuration Bus so that it can configure or control modules attached to this Bus.

There are many occasions where the Control FPGA needs to interrupt the DSP processor, for instance when a data error is detected by the EDAC circuit for SDRAM data and the error is not a one-bit error i.e. not self-correctable. All interrupts are gathered from their sources and then an interrupt signal is connected to a pin of UHPI interface which can be configured as an interrupt input pin to the DSP processor.

III. *SPACEWIRE INTERFACE*

A five port SpaceWire router is provided on the HPPDSP unit. It has two SpaceWire ports (ports 1 and 2), two ports connected to the DMA Bus *A* inside the Control FPGA (ports 3 and 4) and a configuration port (port 0) connected to the Configuration bus inside the Control FPGA. If nominal and redundant ports are required the two SpaceWire ports may each be given a nominal and redundant external LVDS driver/receiver. The SpaceWire Router is illustrated in Fig. 2.

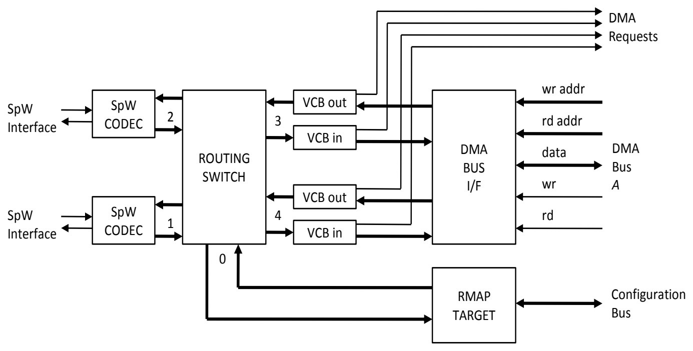


Fig. 2. *SpaceWire Router Block Diagram*

The two SpaceWire interfaces are connected to a routing switch as ports 1 and 2. Ports 3 and 4 are attached to pairs of VCBs which are connected to the DMA Bus *A*. Port 0 is attached to an RMAP Target attached to the Configuration Bus. Configuration of the SpaceWire interfaces (e.g. link speed) and router (e.g. routing tables) is performed over the Configuration Bus. They can therefore be configured by any of the SpaceFibre or SpaceWire interfaces.

IV. *SPACEFIBRE INTERFACE*

There are three SpaceFibre interfaces on the HPPDSP. Two of them, SpFi 1 and SpFi 2, are for connecting to instruments or other HPPDSP units operating in parallel. Each of these SpaceFibre interfaces has three VCs that can be used for data transfer to/from DSP memory. These VCs are connected to the DMA Bus *A*. A fourth VC is used for configuration/control purposes and is connected to an RMAP Target that is attached to the configuration bus. The VC attached to the RMAP Target provide a means of configuring the HPPDSP system remotely over SpaceFibre.

The SpaceFibre interfaces use external SerDes devices (TI TLK2711) which are available in space qualified version.

A block diagram of the SpaceFibre interfaces is given in Fig. 3.

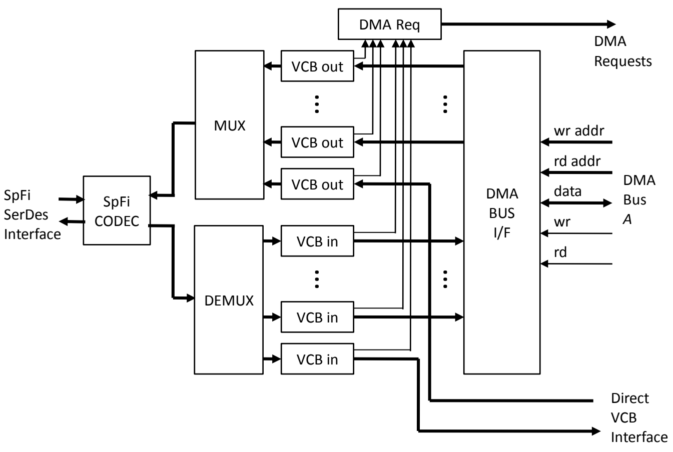


Fig. 3. *SpaceFibre Interface Block Diagram*

The DMA Bus interface connects the DMA Bus *A* to the input and output VCBs in the SpaceFibre interface. When writing to a SpaceFibre interface the output VCBs are addressed. When reading the input VCBs are addressed. The output VCBs are multiplexed by the MUX into a single stream of SpaceFibre data frames into the SpaceFibre CODEC. The SpaceFibre CODEC encodes the data frames, adding any link control characters that are required and passes the resulting symbol stream to the external SerDes for 8B/10B encoding and transmission. Symbols received from the SerDes device are passed to the SpaceFibre CODEC and the data frames are extracted and passed to the DEMUX for writing into the appropriate input VCB. The data in the input VCBs are taken out when the DMA Controller reads the VCB.

There is an input and output pair of VCBs that are not attached to the DMA Bus *A*. These are connected to an RMAP Target and used for configuring and controlling the HPPDSP unit.

SpFi 1 and SpFi 2 each have four pairs of VCBs (three attached to the DMA Bus *A* and one pair to an RMAP Target) and SpFi M/S has eight pairs (seven attached to the DMA Bus *A* and one pair to an RMAP Target).

V. *DMA CONTROLLER DESIGN*

The DMA Controller takes DMA requests from DMA Bus *B*, for a small amount of data access at any memory location.

The DMA Controller also manages transfer of data from the SpaceFibre, SpaceWire, to and from DSP memory. It does this under control of the DSP i.e. the DSP processor determines where in DSP memory the data is to be placed and how much data is to be read in a burst.

In a Master HPPDSP unit, the DMA Controller copies the data being read to the SpaceFibre master/slave interface. This is done at the same time as the data is being read out of one of the interface by the DMA controller by providing a concurrent write strobe and IO write address that specifies where the data is to be copied to. In this way the data is read from one of the
o DSP memory and concurrently written to
ster/slave interface for transferring to the

it, the DMA Controller accesses the
slave interface in place of the SpaceFibre,
erfaces. It DMAs data from VCBs in the
slave interface as if it were coming from
ibre, SpaceWire interface. For slave unit, if
requests to write data to a SpaceFibre or
via the DMA Controller it simply discards

ntroller contains several channels each
grammed by the DSP processor to perform
nsfer.

VI. *APPLICATIONS*

One of the possible target applications can be processing image data, for instance image compression, where image data arriving over a SpaceFibre link is streamed into the DSP memory and then processed by the DSP processor. Once processed the image processing results are transferred out using another SpaceFibre or SpaceWire interface, depending on the data rate required.

VII. *CONCLUSIONS*

The HPPDSP is an experimental DSP processing system for spaceflight applications with both SpaceFibre and SpaceWire interfaces. Currently the prototype board is developed and tested, and the design of the Control FPGA is nearly finished. Once the hardware design is complete software will be developed by Astrium and the entire system tested.

ACKNOWLEDGEMENTS

The authors would like acknowledge the support of ESA with Astrium as the main contractor for the HPPDSP activity.

REFERENCES

[1] ECSS Standard ECSS-E-50-12C, "SpaceWire, Links, Nodes, Routers and Networks", Issue 1, European Cooperation for Space Data Standardization, January 2008.
[2] S.M. Parkes, A.Ferrer, A. Gonzalez, and C. McClements, "SpaceFibre Specification", Draft E1, University of Dundee, September 2012.
[3] Texas Instruments, "TMS320C6727B Floating-Point Digital Signal Processors – Data Sheet," SPRS370E, September 2006.
[4] Texas Instruments, "TLK2711A 1.6 TO 2.7 GBPS TRANSCEIVER", SLLS908A, September 2009.

Development of Software Platform Supporting a Protocol for Guaranteeing the Real-Time Property of SpaceWire

SpaceWire onboard equipment and software, Long Paper

Mitsutaka Takada, Hiroaki Takada, Yang Chen
Center for Embedded Computing Systems,
Graduate School of Information Science,
Nagoya University, Furo-cho, Chikusa-ku, Nagoya, Japan.
mtakada@nces.is.nagoya-u.ac.jp,
hiro@ertl.jp, chenyang@ertl.jp

Masaharu Nomachi
Research Center for Nuclear Physics, Osaka University,
10-1 Mihogaoka, Ibaraki, Osaka, Japan.
normachi@rcnp.osaka-u.ac.jp

Takayuki Yuasa, Tadayuki Takahashi
Institute of Space and Astronautical Science, JAXA
3-1-1 Yoshinodai, Sagamihara, Kanagawa, Japan.
yuasa@astro.isas.jaxa.jp, takahashi.tadayuki@jaxa.jp

Abstract— **We have investigated a method to guarantee the real-time properties of SpaceWire based on SpaceWire-D, and proposed an extended protocol. The proposed protocol has wider applicability and a higher transfer rate than SpaceWire-D. In addition we have developed a software platform incorporating the proposed protocol to enable application software that uses the protocol to be developed with greater ease.**

First, this paper describes the proposed protocol for guaranteeing the real-time properties of SpaceWire and then presents the software platform that implements the protocol.

The proposed protocol is an extension of SpaceWire-D and incorporates the following features/concepts: subnet concept, flexible time slotting, network operation mode concept, split RMAP transaction, accommodation of any upper layer protocol, and optional FDIR mechanisms.

The developed software platform consists of an embedded real-time operating system (RTOS), called the TOPPERS/HRP2 kernel, and middleware that implements the proposed protocol.

The TOPPERS/HRP2 kernel is an open-source RTOS based on the ITRON specification, an RTOS API specification that is widely used in Japan. Key features of the TOPPERS/HRP2 kernel include its small footprint, real-time properties, high-level reliability and ability to support the protection mechanisms for memory access and object access.

The middleware running on the RTOS supports RMAP initiator functions, RMAP target functions, and other basic functions supporting SpaceWire. The RMAP initiator functions are responsible for sending RMAP commands in according with the predefined schedule table. Therefore, application software running on the middleware can initiate an RMAP transaction at arbitrary times.

The developed software platform helps application developers carry out software development without having to become involved in the complex time management process of SpaceWire packets, RMAP transactions.

In addition to this middleware, we have also studied worst-case latency (WCL) analysis and real-time scheduling of RMAP transactions for SpaceWire-D. These will be reported separately by Yang Chen et al. (this conference).

Index Terms— SpaceWire-D, time slot, middleware, real-time kernel, schedule table.

I. Introduction

SpaceWire as a network standard for spacecraft has begun to be adopted for use on scientific satellites. In order to apply SpaceWire to spacecraft other than satellites and to areas other than space, there is considered to be a need for techniques that have the ability to guarantee worst-case latency (WCL) for packet delivery over SpaceWire.

SpaceWire-D has been proposed as a method of real-time properties guarantee of SpaceWire.

In order to ensure real-time properties using existing SpaceWire nodes and routers, the network is time-divided into units called time slots in SpaceWire-D, and the basic approach is to have one RMAP packet transfer in each time slot. This approach ensures that it is easy to guarantee real-time properties, however, there is a problem in that the effective transmission rate becomes lower. It was reported in [5] that the effective transmission rate when transferring data by the RMAP packet using a command consisting of 256 bytes and 4 bytes on SpaceWire-D will be approximately 15% of the SpaceWire link rate.

Furthermore, this paper assumes (that the node receiving the RMAP command starts transmittings the RMAP reply within 5μs) that the RMAP target is implemented in hardware. If the RMAP target is implemented in software, the effective transfer rate is significantly lower to the degree that it is no longer practical.

In order to widen the application scope of SpaceWire, we have examined methods to guarantee its real-time properties. Regarding the study into guaranteeing real-time properties, we started from the requirements for the SpaceWire network. Even in the satellite network, each system – bus control, mission control and the attitude control system – has quite different network requirements. In addition, considering application to other spacecraft and non-space fields, network requirements become more diverse.

In cooperation with JAXA, Japan's Nagoya University has set up a study group to guarantee the real-time properties of SpaceWire, and called for participation from companies that develop spacecraft. In this study group, we collected requirements for the SpaceWire network and have examined real-time assurances based on it. In line with the basic approach of SpaceWire-D – and to study improvements aimed at extending the scope – we have produced guidelines for methods that guarantee the real-time properties of SpaceWire [4].

In this paper, we describe middleware (hereinafter referred to as SpaceWire middleware) that runs SpaceWire control software that was developed based on SpaceWire real-time guarantee method guidelines created by the study group using real-time OS(RTOS), and RTOS that controls SpaceWire middleware. We also describes a software platform that consists of tools used to assign time slots that are determined in advance utilizing static SpaceWire packets.

The paper is organized as follows: In Chapter 2, we describe proposal protocols aimed at extending SpaceWire-D, and SpaceWire real-time properties guarantee methods guidelines. In Chapter 3, we describe relevant components and a software platform developed by applying the guidelines of the real-time properties guarantee method. In Chapter 4, we describe the operation of the software test platform. Finally, Chapter 5 summarizes the paper.

II. Proposed protocol extends SpaceWire-D

SpaceWire middleware has been used in the implementation process based on the proposed protocol that extends the functionality of the part from SpaceWire-D. This chapter describes the part that extends from SpaceWire-D in the proposed protocol. The schedule information decision tool will be stated in each component of the software platform in Chapter 3.

A. Network operation mode

When using SpaceWire in spacecraft, we shall support a change of operation mode in a spacecraft and a switch of traffic route during a failure depending on the importance of the mission. The concept of "Network operation mode" is introduced to support a change of operation mode and a switch of route in the proposed protocol. However, another method is assumed to be provided so that the current network operation mode can be transmitted to all routers and nodes.

B. Subnet

SpaceWire-D did not describe SpaceWire networks that consist of a number of nodes. As the proposed protocol defines the SpaceWire network, it was decided to provide a constraint that is divided into multiple closed networks that do not share a network link between each network operation mode. Multiple closed networks that do not share a network link between each network operation modes are called subnets.

The following example is considered one way to use a subnet. SpaceWire network is divided into multiple subnets by the functional unit node shown in Fig. 1.

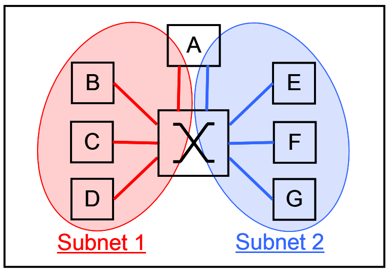


Fig. 1. Example of subnets (function)

Otherwise it has been divided into subnets as redundant paths between the nodes in Fig. 2. When it cannot communicate between nodes in subnet1, it will be able to communicate to switch to the subnet 2.

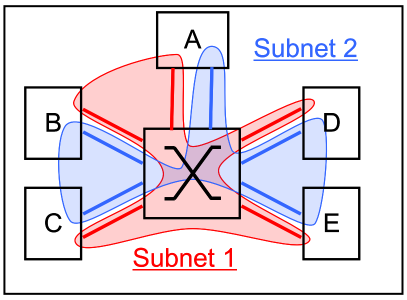


Fig. 2. Example of subnets (redundancy)

Different subnets can share the routers and nodes, but they cannot share links. However, it is assumed that no interfaces may occur during packet transfer of another link. If a packet transfer of another link causes an interface, this interface needs to be considered during calculation of packet transfer latency as well as the potential of this interface to divide time slots. When using the proposed protocol, packet transfer must finish within each closed subnet; then it will be able to calculate the allocation of time slots for the packet transfer per subnet.

C. Flexible time slot

Time slots are specified to be divided in the SpaceWire time code in SpaceWire-D. However, it is difficult to implement SpaceWire-D protocol stacks in software in the case where SpaceWire networks are configured within a short time span by time slot. Therefore, even if time code used by the entire network is short, the time slots definition needs to be more flexible so that software can accommodate the time slot used in SpaceWire-D. In this proposed protocol, it is suggested that a way to address the above problems is to extend the time slot as follows.

If the time value contained in the time code is a multiple of 2^n (where, $n = 0$ to 6, and value n depends on each subnet), this time code is referred to as the “Time code that separates time slot.” An interval from “Time code that separates time slot” to the next “Time code that separates time slot” is called the “Time slot.” When the number of time slot is 2^m, the time slot number is caluculated according to as the following equation.

$$TimeSlots = (TimeCode / 2^n) \ mod 2^m \qquad (1)$$

The length and number of time slots are determined for each subnet. Basically, the time slots are separated by all time codes, and 64 time slots are used. In addition, n=0 and m=6 are used. Any time code other than “time code that separates time slots” is ignored by the proposal protocol. When the time code cycle is shortened, usage of link bandwidth is reduced even if the latency is increased. In this case, an effective strategy is to increase the usage of link bandwidth by increasing the length of time slot used in the proposal protocol.

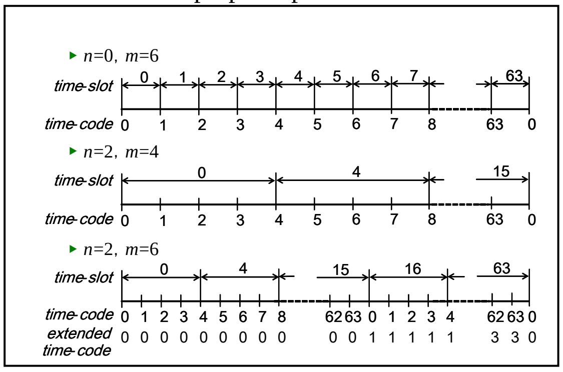


Fig. 3. Example of flexible time slot

D. Transfer packet type

SpaceWire-D can be transfered only if RMAP packet, the proposal protocol, cannot be limited to the RMAP packet. If we know the destination nodes and the size of any packet, it can be applied to a packet other than RMAP.

E. Split RMAP Transaction

In SpaceWire-D schedule, the RMAP reply packet is to be sent in the same time slot as the RMAP command packet.[3] However, if the target node is implemented in software, a long period of time is required before a RMAP reply packet is sent, and network usage as a whole is reduced because the latency is reduced. As part of the proposed protocol, it was suggested that the node could transmit RMAP reply packet by using a time slot that is different to the time slot received from the RMAP command packet. This is called “Split RMAP Transaction.” If using the Split RMAP Transaction, it is necessary to determine in advance - in the initiator and target nodes - not only the RMAP command packet but also the time slot number of the RMAP reply packet.

TABLE I. EXAMPLE OF SCHEDULE INFORMATION

Time slot number	**Packet type**	**Target node list**	**Slot number of RMAP command**	**Total packet size**	**Total RMAP reply size**
0	RMAP command ("write" command without verify)	11, 12, 13, 14	0	1024	20
	Others	15		2000	
6	Others	10	6	–	
8	RMAP command ("read" command)	11, 12, 13, 14	9	40	1024
10	Others	17		2000	
12	RAMP Reply	10	11	–	
…	…	…	…	…	…

III. Components of the Software Platform

In order to ensure the real-time properties guarantee of SpaceWire, it is necessary to not only provide the proposed protocol, but to also provide a solution that make it easy to use the proposal protocol. We consider it necessary to include the following as a component of the software platform where the SpaceWire real-time properties guarantee can be ensured.

1) SpaceWire schedule information decision tools

Used to determine the scheduling table for the appropriate time slot assigned to SpaceWire network consisting of multiple nodes.

2) SpaceWire middleware

Implementes the proposed protocols based on SpaceWire-D, and performs communication control based on the schedule information assigned by SpaceWire schedule information decision tools.

3) Embedded system Real-Time OS

A real-time kernel with high response performance that can run SpaceWire middleware and communications applications, the processing mission of software. We have developed a real-time OS - the TOPPERS/HRP2 kernel - in this study.

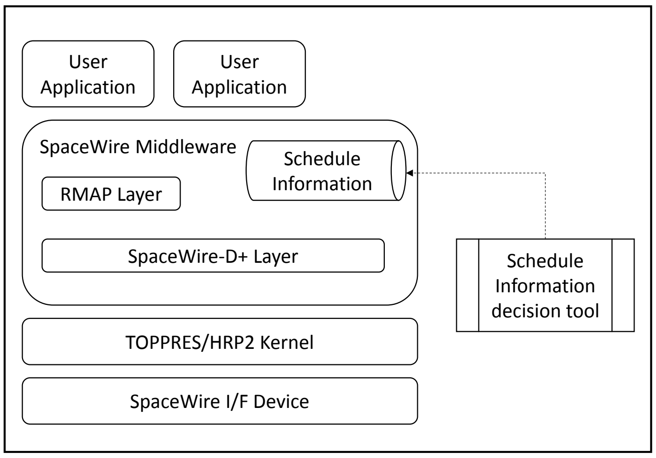


Fig. 4. System diagram of SpaceWire software platform

B. TOPPERS/HRP2 kernel

Detection of the time code is becoming a very important step in managing the time slot because of the neccessity to send SpaceWire packets according to the timing of predetermined time slots. This is because the time at which to send a packet that the user assigned to the time slot after the time code was detected is directly linked to the performance of real-time OS.

We have subsequently developed the TOPPERS/HRP2 kernel (HRP2 kernel: High Reliable system Profile kernel version 2), a real-time OS for embedded systems. Based on the μITRON specification that has become the de facto standard, the HRP2 kernel is a real-time OS the user adds a functions to in order to apply to areas where high-level reliability is also required to develop embedded software; for use in areas such as the space and medical fields. Features of the HRP2 kernel are as follows:

1) Source code is easy to read and change

The HRP2 kernel was produced in part to enable greater readability of source code or modifications. However, we have not adopted inefficient algorithms in order to pursue ease of readability. It enables efficient implementation of the algorithm even for complex structures, such as when using a heap.

2) Easy porting to another target CPU

As many parts of the HRP2 kernel have been written in C language and there is a clear separation of the target-dependent and non-dependent parts, this structure makes it easy to facilitate the porting to other targets. While interrupt processing is a very important in terms of improving run-time performance, the difference between proccessors is large and easily masks the differences between processors.

We define the "TOPPERS standard interrupt processing model", and have adopted an interrupt process to absorb the difference interrupt with other processors while at the same time; maintaining balance between the run-time performances in the HRP2 kernel. The real-time kernel, which is the base of the HRP2 kernel has been porting for the main CPU used by embedded system.

3) High performance and small footprint

The kernel, the majority of which was written in C language, exhibits high execution performance and low memory use. As a matter of policy and to reduce the amount of memory used, in the HRP2 kernel, the kernel maintains the resources statically in advance, a process called static configuration. Embedded systems, which perform predetermined processing, differ between general computers and personal computers, and response performance is required for processing. Therefore while the amount of resources used, in applications (tasks, semaphores, etc.) can be determined for each application, it is possible to maintain the resources of the kernel in advance during the application development stage.

Advantages of static configuration are as follows:

a) High response time when using kernel resources

Dynamic configuration requires preparation in order to use in run-time, in addition to resource preparation. It eliminates the need for a preparatory process through adoption of a static configuration.

b) Store configuration information into ROM region

Often, many application areas in embedded systems have greater capacity in the built-in ROM than in the RAM. Configuration information can be stored in the ROM by performing a static configuration, and it is expected to have an effect on reducing RAM consumption.

c) Memory protection function to perform static memory allocation

In recent years, embedded systems have become increasingly complex, and as such, high-level reliability is required. General computers typically have a memory protection function that prevents unauthorized access to the memory in order to ensure the reliability of complicated systems. Similarly, it may be necessary for embedded systems have memory protection even in areas where high-level reliability is required. The HRP2 kernel has a memory protection feature that prevents access other than that permitted to enable access to the memory area where a task is carried out. The HRP2 kernel can statically allocate code and data section areas that need to set the same access authority to a continuous address, and can detect illegal memory access that is occurring during execution of the embedded system. By using the HRP2 kernel, the application developer does not need to know the specific memory address and size to set in MMU/MPU. Application developers can determine attribute settings of memory protection in unites of section and object file.

4) Additional features functions of the high-level reliability

The HRP2 kernel has the following additional function in addition to the memory protection function in the kernel of μITRON specification:

- Object Access feature
- Mutex function
- Overrun handler function

5) Easy to deploy open source license for embedded systems

The HRP2 kernel has been released as open source, however, applying licenses such as GNU to embedded systems after modification will be a higher hurdle. The HRP2 kernel developers are applying for an open source license called “TOPPERS license” to make it easier to introduce when using in an embedded system. In particular, a license should notify the secretariat running a project that they are using the source code. This is called "Report wear." [7]

In this way, the HRP2 kernel was developed in mind to be utilized in highly reliable embedded systems. The previous version of the HRP2 kernel has a proven track record, and produce such as the TOPPERS/HRP kernel have been loaded on board the H-IIB rocket, which is used for launching satellites in Japan.

C. SpaceWire middleware

1) The role of SpaceWire middleware regarding real-time property guarantee

SpaceWire middleware has two primary roles. First, to forward the packet according to a predefined schedule. Application programs should no longer have to process packet transfer schedules. SpaceWire middleware also provides support so as not to affect the entire network, even if there is a problem in the application program. Second, when multiple applications are running on SpaceWire middleware, it does not use the time slots of other applications. Therefore, we decided to introduce a communication path - called a channel - in order to implement a SpaceWire middleware.

2) Channel

Channel is a logical communication from an application on a source/initiator node to a destination/target node. One end of the channel is called the communication end point. A channel is referred to as a communication path connecting a communication endpoint in source/initiator node and a communication end point on multiple destination/target nodes.

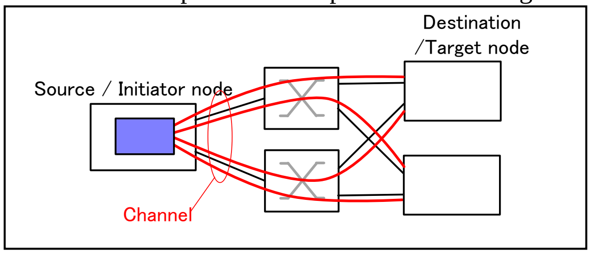


Fig. 5. Example of channel in SpaceWire network

One application can open multiple channels because a channel can be established for each communication purposes. It can also include more than one route to the same destination/target node.

The purpose of introducing the channel is to distinguish the time slot each application is using SpaceWire middleware when multiple applications are running in the node. Decision tools, which will be described later in the schedule information, are also used as a description unit of the communication request.

The following three items are determined by the channel:

- Protocol type
- Source/Initiator node
- List of destination/target nodes

The reason why there is a list of destination nodes, is because the logical topology of a star shape is considered from multiple nodes, and the prospect of the application improves since a channel is open for each application, and packet can be collected from multiple nodes in one channel. Although the method of time slot allocation becomes difficult due to the existence of multiple destination nodes, it does not affect run-time processing since the time slot allocation is completed at the application development.

The contents added with the channel number information to each entry of the schedule information are as follows.

TABLE II. EXAMPLE OF SCHEDULE INFORMATION (WITH CHANNEL ID)

Time slot number	Ch. ID	Packet type	Target node list	Slot number of RMAP command	Total packet size	Total RMAP reply size
0	1	RMAP command ("write" command without verify)	11, 12, 13, 14	0	1024	20
	4	Others	15		2000	
6	2	Others	10	6	—	
8	1	RMAP command ("read" command)	11, 12, 13, 14	9	40	1024
10	5	Others	17		2000	
12	3	RAMP Reply	10	11	—	
…	…	…	…	…	…	…

3) Main function

a) Summary of common function

The number of network operation modes and time slot for each SpaceWire node port, which is a common feature that sets the basic parameters related to the proposed protocol, is set. It also has the ability to change the network operation mode. Because the protocol agreement network operation mode between nodes are not currently standardized, the network operation mode for the middleware is obtained from the application.

b) Summary of RMAP initiator function

RMAP initiator is a function that is primarily able to generate (during system design) static end points. Communication end points have a one-to-one correspondence with the channel, and any command of RMAP Write/Read/Read-Modify-Write can be issued for one communication end point. In addition it has the ability to issue the RMAP transaction for communication end points. This function can send a RMAP Write/Read/Read-Modify-Write command and receive the reply packet. This function describes the following information for each entry in the schedule information:

- SpaceWire port ID
- Network operation mode number
- Time slot number
- Channel ID
- Command type
- Total size of the packet
- Time slots number receiving RMAP reply

- Total size of RMAP reply packet

The issuance of RMAP transaction supports both the blocking and non-blocking types. It also has a function that initializes the communication endpoint and refers to the state of the communication end points and transactions.

c) Summary of RMAP target function

This function can register how to handle each range of RMAP addresses that are accessible from other nodes. Users are usually able to choose whether to access the memory or register the callback function of the application, and leave the process to the application. The RMAP target processing function according to the command type, and in the case of memory access, returns a RMAP reply packet. Regarding RMAP memory access, the RMAP target function carries out processing according to the command type and returns a reply packet. The RMAP reply packet returns the response in the time slot determined according to the schedule information. This information is described for each schedule information entry RMAP target as it is in the RMAP initiator function.

D. Schedule Information decision tools

When SpaceWire middleware is used experimentally or the network topology is simple, schedule information can be created and used manually.

However, since the actual SpaceWire network has redundant paths and the communication requirements of the application become complex, it is difficult to execute verification even when it is created manually. Therefore, a tool that detects redundant paths from the network topology and allocates time slots automatically from the communication requirement of the application will be needed.

We have studied and developed algorithms to determine the schedule information of allocating time slots of the communication path and communication packets based on the network topology and communication requirements. This paper describes the network topology that inputs information into a tool as well as communication requirements. The algorithm to determine the schedule information has been omitted [6].

1) Network Topology

a) Subnet

A subnet is a unit used to divide networks that do not share a SpaceWire link for each network operation mode, and is used to configure the network topology according to the proposed protocol. It is necessary to describe the communication request for each subnet in the case when describing the different communication requests on the same node, however we will not cover in this paper. A subnet consists of each component of nodes, routers and links, and parameters of the subnet. The parameter requires the following information:

TABLE III. SUBNET PARAMETER

Parameter name	**Description**
Network operation mode	Since subnets are required for network operation mode, they defines network operation mode. The schedule information decision tool performs the calculation for each network operation mode.
FDIR period	FDIR is not set as a latency margin of time slots when determining the allocation of a time slot. FDIR is not necessary in the case of successful communication. However, it is necessary for recovering the entire subnet when a packet that does not match SpaceWire-D comes into the subnet. It is presumed that the value of an FDIR period is given; it is not calculated by the schedule information decision tool. Described as a period of time until error recovery FDIR from error detection.
Time slot	Time slot duration and the number of time slots in one cycle, and the number of time codes separating the time slot is set as a parameter of the time slot.

b) Node

Node is a SpaceWire device with an endpoint port of the channel and can be described in communication requirements for the source/initiator node and destination/target node. Node parameters are as follows:

TABLE IV. NODE PARAMETER

Parameter name	**Description**
Maximum sending delay time	Maximum delay time from when a packet node recognizes the time slots to the time when the packet assigned in the time slot begins transmission. This parameter becomes effective for a source/initiator node. Although it is currently set as a parameter fixed in the node, it is considered that it will provide mode appropriate values when dividing the parameter for each port.
Maximum receiving delay time	Maximum delay time from when the node starts receiving a packet until when it starts sending a reply packet. This parameter sets a conservative value because it is fixed in the node. It is presumed in future that a more appropriate value will be expected by setting the constitution of a port and receiver, a command packet type and packet length, and the memory access processing time of the node.
Port	A port number, logical address and a key that is unique to each node can be set in the port. Because the channel ties between logical addresses of the node, the logical address is always given to the node.

c) Router

In the proposed protocol, the SpaceWire router is a device used to wormhole route packet. Since a packet flowing in the time slot with the subnet does not prevent wormhole routing, propagation delay time can be set as a parameter in the router.

d) Link

A link is a line segment to link between nodes and routes, and ports of the node and router. Link ID as a parameter of the link, link speed, link propagation delay time and endpoint information of the node and router can be set as a parameter.

The network topology can be described in XML from the information.

```
<?xml version="1.0" encoding="UTF-8"?>
<SpaceWireNetworkTopology>
   <Subnet>
     <FDIR IntervalTime="5" />
     <Timeslot Number="64" SlotTime="15625" TimecodeInTimeSlot="1" />
    <Router Id="SpaceCardA-SOISOC" NetworkPropagationDelay="2">
       <Port Number="1" />
       <Port Number="2" />
    </ Router>
    <Node Id="SpaceCardA-memory" MaxSendDelayTime="5"
            MaxReceiveDelayTime="15" NetworkPropagationDelay="2">
       <Port Number="0" LogicalAddress="0x50" Key="0x50" />
    </Node>
    <Link Id="SpaceCardA-SOISOC-1-to-SpW-GbW-1" Speed="10"
                                      NetworkPropagationDelay="0">
            <Endpoint1 Id="SpaceCardA-SOISOC" Port="1" />
```

Fig. 6. Example of network topology description

2) Communication requirements definition

The purpose of the communication requirements definition is to describe the requirements for a communication packet from the application developer's perspective. Communication requirements can be seen as a list of the information in relation to the channel in network operation mode in a subnet. The elements consisting of the channel and its parameter are described in this paper.

a) Source node

This is source/initiator node and only once source node exists in the channel.

b) Destination nodes

These are the destination/target nodes, and one or more destination nodes can be described in the channel.

c) Packet information

This describes the information packet used in the channel. The following information is required when determining the schedule information.

TABLE V. PACKET INFORMATION PARAMETER

Parameter name	Description
Type	Packet type
CargoSize	Cargo maximum size used by the channel
Period	Number allocated in 1 cycle time slot by this channel. Cf. If the cycle in the network topology is 64 and the period is set as 64, it means this channel is required to be used for each time slots.
Reply	When there is a reply to RMAP command, this information is described because the calculation for the processing time of the reply packet is required.
Verify	This information is described because the calculation for the delay of the maximum receiving delay time is required when the target node is used to verify processing.
Reply Interval	This can be specified when responding to a reply packet with a time slot other than the time slot receiving the reply packet. It is effective whten the target node is implemented with software or a node that has slow processing time. It is also a parameter used to implement a Split RMAP transaction of the proposal protocol.
Jitter	Jitter can be used when time slot allocation with optional communication can be implemented by shifting the allocated time slots only at the time slot allocating a channel according to the communication cycle interval. It is only valid in the case of search algorithms able to can handle jitter.

The communication requirements definition can describe as following:

```
<SpaceWireChannelsInfo>
   <Channel Id="TEST1">
     <Source Id="SpaceCardA-memory" />
     <Destination Id="SpaceCardB-memory" Redundancy="1">
       <Path Id="SpaceCardA-memory-to-SpaceCardB-memory">
          <Link Id="SpaceCardA-memory-0-to-SpaceCardA-SOISOC-5" />
          <Link Id="SpaceCardA-SOISOC-3-to-SpaceCardA-FPGA-1" />
          <Link Id="SpaceCardB-SOISOC-5-to-SpaceCardB-memory-0" />
       </Path>
     </Destination>
      <Packet Id="TEST1_RMAP-R" Type="RMAP-R" Reply="True"
           Cargosize="1000" Period="2" ReplyInterval="0" Jitter="0"
          Requirement="Constraint" />
     <Packet Id="TEST1_RMAP-W" Type="RMAP-W" Reply="True"
     Cargosize="1000"             Period="2" ReplyInterval="0" Jitter="0"
     Requirement="Constraint" />
   </Channel>
</SpaceWireChannelsInfo>
```

Fig. 7. Example of communication requirement description

IV. OPERATION TEST

In this study, implementation of a software platform was executed. An operation test using actual equipment was executed to determine whether applications created on top of the software platform work correctly in the examined proposed protocol. The following results were reported:

A. Test contents

SpaceWire network topology and communication requirements are described in XML format, the schedule information decision tool analyzes paths, and the allocation of time slot is implemented.

- Prepare the test scenarios for SpaceWire Traffic Generator to generate pseudo traffic by the allocated time slot.
- Embed the results of the time slot allocation of source/initiator node into the configured application.
- Implement communication between initiator/target nodes, connect the equipment, and send packets and time code from Traffic Generator.
- Analyze the RMAP packet information between initiator/target nodes using SpaceWire Link Analyzer.

B. Test results

It was confirmed that initiator/target nodes send and receive RMAP command reply packets with using the assigned time slot.

We confirmed that the packets cannot be transmitted in the time slot assigned by the initiator/target node when packets in the schedule information not assumed by the Traffic Generator are transferred, SpaceWire middleware detects associated errors and alerts the application software.

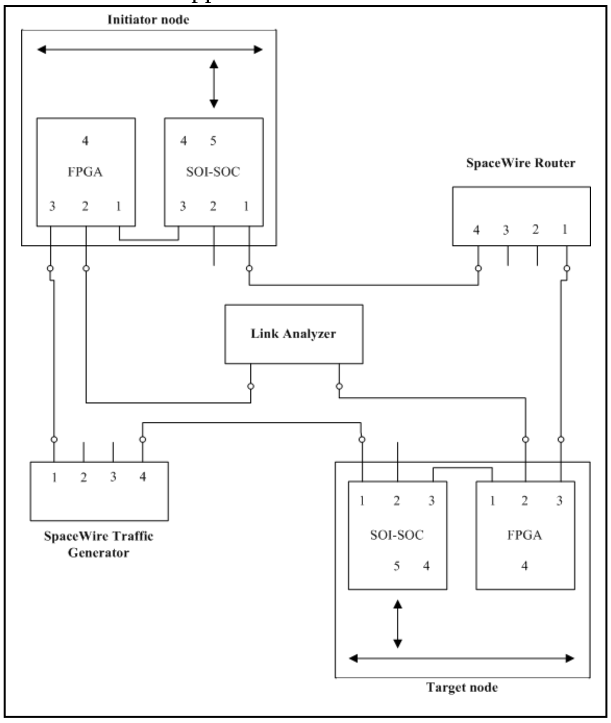


Fig. 8. Test environment of software platform

V. CONCLUSION

SpaceWire software platform, which is the proposal protocol, has been put together as a guideline based on SpaceWire-D. Furthermore, a software platform that satisfies the proposal protocol has been developed, operation tests have been executed, and communication that guarantees the real-time properties of SpaceWire has been confirmed.

In future, we expect to examine the scheduling algorithm used in the schedule information decision tool, and porting SpaceWire middleware to other target boards and OS. Because input information about the schedule information decision tool is in text base format, it is expected to be effective in confirming the network topology that becomes complex when it is possible to present descriptions and displays in a graphical environment.

VI. REFERNCES

[1] ECSS, “SpaceWire – Links, nodes, routers and networks” ECSS-E-ST-50-12C, 31 July 2008, available from http://www.ecss.nl.

[2] ECSS, “SpaceWire – Remote memory access protocol”, ECSS-E-ST-50-52C, 5 Februrary 2010, availabe from http://www.ecss.nl.

[3] SpaceWire WG, “SpaceWire-D Deterministic Control and Data Delivery Over SpaceWire Networks”, Draft B, April 2010, availabe from http://spacewire.esa.int/WG/SpaceWire/

[4] NCES, “SpaceWire Reai-Time Property Guarantee Methods Guidline”, NCES-SPWRT-1-100, 2 July 2012

[5] S. Parkes, A Ferrer, S. Mills, A. Mason, “SpaceWire-D:Deterministic Data Delivery with SpaceWire”, International SpaceWire Conference, St Pertersburg, Russia, June 2010.

[6] Y. Chen, M. Takada, R. Kurachi, H. Takada, “A Scheduling Method of RMAP Transaction for SpaceWire-D”, International SpaceWire Conferenece, Gothenburg, Sweden, June 2013

[7] TOPPRES Project, http://www.toppers.jp/en/index.html

Advanced SpaceWire core with external clock recovery PHY and programmable protocol processing

Onboard Equipment & Software, Short Paper

Björn Osterloh, Andre Schäfer
Digital Signal processing & Information technology GmbH
DSI
Bremen, Germany
osterloh@dsi-it.de

Harald Michalik
Institute of Computer and Network Engineering
IDA, Technical University Braunschweig
Braunschweig, Germany

***Abstract*—Challenges for SpaceWire implementation within Microsemi RTAX FPGAs are especially the clock recovery of a SpaceWire link. With the UT200spWPHY from Aeroflex a space suitable device is provided for external clock recovery. For the Payload Data Handling Unit (PDHU) on the ESA ExoMars mission we have developed a SpaceWire core with interface to the SpaceWire PHY. The core shows significant improvements in terms of resource utilization, complexity, implementation effort and performance. Additionally, we have encountered the need for SpaceWire cores with enhanced protocol processing. SpaceWire supports the implementation of variety protocols, which provides a high flexibility. Dependent on the mission, protocols change and therefore adaption to mission specific requirements is needed. This has led us to develop an advanced SpaceWire core with integrated programmable protocol processing. As most programmable machines, they have a high risk to be non-deterministic because usually they represent processors with infinite states. In these systems it is hard to achieve the complete verification and validation coverage. Therefore, we have chosen a programmable stack machine approach, which is deterministic and provides easy validation.**

***Index Terms*—SpaceWire core, Aeroflex UT200spPHY, programmable protocol processing**

I. SpaceWire Core with external clock recovery

Currently, we are developing for the ESA ExoMars mission the Payload Data Handling Unit (PDHU). The main objective of the PDHU is payload data management with PUS13 support. Data storage is performed in a 1TBit Flash based mass memory. The PDHU comprises 4 SpaceWire links to the instruments, operating at 167 Mbps raw data rate with a net data rate to the PDHU of 100 Mbps. The target device architecture for the SpaceWire cores is a Microsemi RTAX FPGA. Within these devices the most challenging part of a SpaceWire implementation is the clock recovery.

SpaceWire uses Data-Strobe (DS) encoding and the clock can be recovered by simply XORing the Data and Strobe signal. An example for an RTAX clock recovery implementation circuit is depicted in Figure I-1.

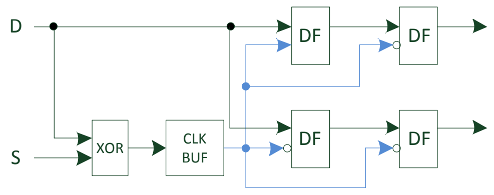


Figure I-1: RTAX DS Clock Recovery

The clock recovery requires a clock path with a delay larger than the data path to prevent Flip Flop setup time violation. On the other hand, the delay difference must not get near to the bit period because otherwise a hold time violation would occur. Implementing the clock recovery in an FPGA is difficult because the tools usually do not perform well to handle a clock path with an XOR in it because both D and S have to be treated as clocks. Careful manual timing analysis is required and for high data rates even manual placing of critical FPGA primitives. Furthermore, these timing requirements have to be guaranteed over the full temperature and voltage range, and even have to consider device degradation and radiation effects e.g. total dose.

The UT200SpWPHY implements this critical circuit and provides a simple single data rate, two bit with clock interface to the FPGA, as depicted in Figure I-2.

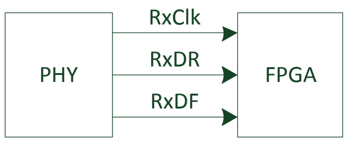


Figure I-2: PHY RX interface to FPGA

The *RxDR* signal represents the on the rising edge (even) received bit. And the *RxDF* signal represents the on the falling edge (odd) received bit. This scheme has several advantages for a SpW RTAX implementation with:

I. Synchronous Clock to Data relation results in a simpler SpW Receiver implementation
II. No time consuming manual timing checks which results in a robust FPGA design.
III. Supports data rates beyond an FPGA only approach.

We have developed a SpW Core with interface to the UT200SpWPHY. The layout results are depicted in Table I-1 and Table I-2:

Resource	*Used*	*Total (%)*
Sequential	297	2,76
Combinational	485	2,26
RAMs	1	1,5

Table I-1: SpW RTAX 2000S resource utilization

Clock Source	*Frequency (MHz)*
RxClk (from PHY)	145
Clk (Core Clk)	102

Table I-2: SpW RTAX performance

The RxClk of 145 MHz provides a raw link data rate from the transmitting end of 290 Mbps, due to the two bit transfer. However, the limiting factor is the transmitter Clk which is equal to the maximum SpW Core frequency of 100 MHz. Consequently, a maximum transmitter raw data rate of 100 Mbps can be achieved. This provides a maximum full-duplex data rate (data and FCTs only) of 70 Mbps for the transmitter and 228 Mbps for the receiver (net). It has to be considered that the maximum data rate of the SpW PHY is limited to 200Mbps (raw). For ExoMars only a unidirectional data transfer from instrument to PDHU has to be supported. In this case the PDHU SpW transmitter provides only the flow-control characters to instrument. Receiver and transmitter are allowed to operate at different signalling rates. The transmitter can run by a factor up to 20 slower than receiver in a unidirectional setup without wasting bandwidth. This is because only one FCT (Flow Control Token) from transmitter has to be sent every 8 Data Character. For ExoMars the Core Clk frequency is 40 Mhz and an overall unidirectional data rate of up to 320 Mbps can be supported. Considering a full-duplex operation, the ExoMars setup achieves data rates of: 26Mbps transmitter and 127 Mbps receiver (net).

The Aeroflex UT200SpWPHY provides significant improvements for a SpW RTAX implementation in terms of performance, resource utilization and implementation effort.

II. SpW core with Programmable Protocol Processing

SpaceWire supports a variety of protocols. Instruments with SpaceWire and CCSDS compliant protocols are capable to transfer within the packet structure Housekeeping (HK) data and e.g. image data. These packets must be checked for accuracy and furthermore the data has to be demultiplexed into: HK data for e.g. status of the instrument and e.g. image data. Furthermore, the SpW Remote Memory Access Protocol (RMAP) provides means to access memory or registers directly of a SpW node [3]. Since the protocol and data content changes from mission to mission, a dedicated decoder or protocol handler has to be implemented for each mission specific requirement. An improvement is to implement programmable protocol processing for this task. The disadvantage is that programmable processing is mostly related to CPUs. CPUs are highly flexible but have also a high risk to be unpredictable. A CPU with heap, stack, interrupts and cache maybe represented as a state machine with infinitive states. Furthermore, the translation of source code into CPU instructions (compiler) with optimization stages implies also high risk for non-determinism. This approach requires high validation effort for both: CPU (hardware) and software. CPUs have furthermore high resource utilization requirements. Therefore, we have evaluated different architectures to find an appropriate candidate which provides:

I. Low resource utilization, low complexity.
II. High data throughput.
III. Determinism and robustness.
IV. Easy to program
V. Linear deterministic program flow.
VI. Small resource requirements for programs.

Within this context we were looking for a simple stack machine with Forth support. The Forth language has several advantages like its simple compiler and is especially very compact in terms of code size. We found the J1 Forth CPU developed by James Bowman [4]. The J1 has been developed to process video streams in Xilinx Spartan-3E FPGAs and is implemented with less than 200 lines of Verilog. The internal states of the CPU consist of: (i) a 33 deep data stack of 16 bit width (ii) a 32 deep return stack of 16 bit width and (iii) 13 bit program counter. No other states exist in the J1; neither condition flags, modes or extra registers. This low complexity leads to a low state set and is therefore ideal for a robust deterministic design. The J1 is subdivided into five categories of instructions with (i) literal, (ii) jump, (iii) call and (iv) ALU which are implemented in an unencoded hardwired layout. The ALU supports overall 16 operation codes which comprise e.g. add, and, or, shift and stack operations. Instruction fetch and decode is performed in parallel due to the non-dependency of ALU codes and instruction codes. Programming the J1 is very simple. A set of Basewords is available which are written in Assembler. These Basewords are direct ALU operations with e.g. add and stack pointer manipulation and represent the supported Forth words. All additional user defined functions are implemented in a Dictionary which comprises e.g. loop or *if* comparison statements which are build-up from the available Baseword set. The compiler has therefore a low complexity and only maps the hardwired ALU operations from the Baseword set into any user defined program sequence and calculates the offsets for jump conditions. This simplicity of the compiler also carries out a linear and deterministic program flow.

The J1 has been developed for Xilinx and dual-port RAM. Xilinx RAM provides the capability to initialize its content within the FPGA configuration stream. This is used in the J1 for loading the program code. Forth stores variables usually within the program code, there are no separate memory section for program and data. RAM is a costly resource in an RTAX device due to the low availability. Additional, a RTAX device does not provide the initialization of RAM with a predefined configuration.

We have therefore modified the J1 to a RTAX suitable design:

A minor task was the translation from Verilog to VHDL which is more commonly used in Europe. The memory areas for program code and variables have been separated. As replacement for local variables the return stack can be used or, if necessary, an additional RAM can be added. The compiler has been modified to store the variables consecutive in the RAM area. The program code is represented by a ROM which is implemented as combinational logic. This can be become costly in terms of resource utilization and depends highly on the program code size. But this has also the advantage for secure sensitive applications, because the program code itself cannot be read-out and is hardwired in the RTAX anti-fuse device. To support an external ROM we have modified the J1 to support hand-shaking mechanism. This provides the ability to connect to common bus interfaces e.g. AMBA or Wishbone, and external memory devices e.g. ROM or EEPROM.

We have implemented the modified J1 into a Microsemi ProASIC3E in a basic hardware setup with a Wishbone bus and a UART interface to test its functionality. First results showed the correct behaviour of the core and also 100 % code coverage during simulation could be easily achieved. In the ProASIC3E the system runs at 40 MHz. The core has been synthesized for an RTAX2000S and the results are depicted in Table II-1.

Resource	*Used*	*Total (%)*
Sequential	290	1
Combinational	116	1
RAMs (optinonal)	1	1,5

Table II-1. J1 RTAX implementation (without ROM)

The depicted resource utilizations are without any combinational ROM implementation. The core itself achieves then an operating frequency of 100 MHz. As mentioned before, the operating frequency depends highly on the program code size if it is implemented as combinational ROM. Therefore, we are working on a cascaded ROM implementation to improve the overall performance.

The next step will be to add a SpaceWire core to the J1 and implement a protocol handler e.g. RMAP for a common space application. RMAP is especially suitable for an implementation because of its relatively low-level complexity. The J1 could be connected to the SpaceWire input/output ports and perform all necessary processing with packet data integrity checks for the verified option, acknowledge codes and error codes processing and the actual data transfer to destination. The J1 architecture provides means to extend the available ALU operation code set. This could be used to implement CRC calculation directly within the CPU. Overall we think the J1 RTAX implementation provides an efficient and robust solution to implement SpaceWire protocol and data handling within an space suitable device.

III. Conclusion

The Aeroflex UT200SpWPHY provides significant improvements for a SpW RTAX implementation in terms of performance, resource utilization and especially implementation effort. SpW cores with integrated protocol processing based on stack machines provide high flexibility, fast adaption to mission specific protocols and non-complex validation. Within this scope we have presented a Forth programmable stack machine based on a modified J1 which can be implemented into a Microsemi RTAX device. The stack machine has low complexity, low resource utilization requirements and is easy to program. Especially the low complexity provides deterministic design which has several advantages for the verification and validation. The stack machine provides an efficient and robust solution to implement SpaceWire protocol and data handling within a space suitable design. The verification of the stack machine has been completed and we plan now to connect it to a SpW core and implement a protocol handler e.g. RMAP.

IV. References

[1] ECSS, *Space Engineering: SpaceWire–Links, nodes, routers, and networks*, ESA-ESTEC, Noordwijk Netherlands, January 2003, ECSS-E-50-12A

[2] Aeroflex, *UT200SpWPHY01 SpaceWire Physical Layer Transceiver*, Aeroflex Datasheet, February 2008.

[3] ECSS, *Space Engineering: SpaceWire–Remote memory access protocol*, ESA-ESTEC, Noordwijk Netherlands, February 2005, ECSS-E-50-52C

[4] James Bowman, *J1: a small Forth CPU Core for FPGAs*, EuroForth 2010, p.43-46, Hamburg September 2010

Wednesday 12 June

Components 1 (Long & Short)

18x SpaceWire Router based on the DARE 180nm Library

SpaceWire Components
Long Paper

Sandi Habinc, Jonas Ekergarn, Martin Simlastik, Fredrik Ringhage
Aeroflex Gaisler AB
Gothenburg, Sweden
info@gaisler.com

Steven Redant, Kurt Stinkens, Geert Thys, Jagadeesa Das Arul Mahesh
Imec
Leuven, Belgium

Martin Suess
European Space Agency
Noordwijk, The Netherlands

***Abstract*— The 18x SpaceWire router is a new 18 port stand-alone router component currently being specified by Aeroflex Gaisler. Today there is no component available on the world market exhibiting more than eight SpaceWire ports. The goal with this new development is to provide this missing key component to the ever increasing number of customers requiring manifold ports.**

The 18x router is based on the GRSPWROUTER configurable SpaceWire IP core developed by Aeroflex Gaisler. The IP core has been configured to provide 16 SpaceWire ports with on-chip LVDS transceivers and two SpaceWire ports with LVTTL signals supporting off-chip LVDS devices.

The device includes support for the incoming SpaceWire standard revision 1 (ECSS-E-ST-50-12C Rev. 1), the SpaceWire-D protocol and the SpaceWire Plug-and-Play protocol currently being developed for ECSS.

***Index Terms*—SpaceWire, Networking, Spacecraft Electronics**

I. Introduction

Currently there is no SpaceWire router component on the market with more than 8 SpaceWire ports. Both ESA and several companies in the space industry have indicated 16 as the most viable number of SpaceWire ports for routers in the near future. Aeroflex Gaisler intends to provide this key component with a new 18 port SpaceWire router ASIC. The design is be based on the GRSPWROUTER configurable SpaceWire router IP core [1]. This core supports three different port types: SpaceWire ports, AMBA ports and FIFO ports. These will be further explained later in the IP core section.

During the development phase, two configurations of the IP core were identified as potential candidates for the final ASIC: one with 16 SpaceWire ports with on-chip LVDS transceivers and two additional SpaceWire ports or two FIFO ports; and the other with 16 SpaceWire ports and two internal AMBA ports connected to a PCI interface. Both were evaluated in detail to determine which one would eventually be used for manufacturing. The final choice was driven by the number of available pins in the package that was selected, a 256 pin ceramic quad flat package.

Other considerations that were taken into account were such as whether to include support for the incoming revision 1 of the SpaceWire standard (ECSS-E-ST-50-12C Rev. 1), the new SpaceWire-D and Plug-and-Play protocols. The problem has been the lack of a firm schedule for finalization of these standards. In fact, none of the standards have been completed at the time of tape out.

However, Aeroflex Gaisler is actively involved in the revision 1 work and has also been reviewing and discussing the two other protocols with the developers. In this way the risk implementing something that will later on changes in the protocols have been mitigated.

II. Router IP Core Properties

The GRSPWROUTER IP core is the central component in both of the suggested configurations. It supports from 2 to 31 ports of three different types: SpaceWire, AMBA and FIFO. The SpaceWire ports are normal SpaceWire links and will support at least 200 Mbit/s. FIFO ports provide 9-bit parallel interfaces with control signals in each direction (read/write) which can be used to interface external units or to cascade two or more 18x routers without any glue logic. The AMBA ports interface to an AMBA AHB bus using DMA on the bus. All three port types connect to the core router switch matrix using identical FIFO based interfaces. There is no way to distinguish the three ports on the SpaceWire packet level and upwards.

The configurability provided by the IP core makes it usable in many different applications. It has already been used in several standard rad-hard components on Actel RTAX2000SL and RT ProASIC3 FPGAs and is also used in the Next Generation MicroProcessor (NGMP) system-on-chip activity funded by the European Space Agency.

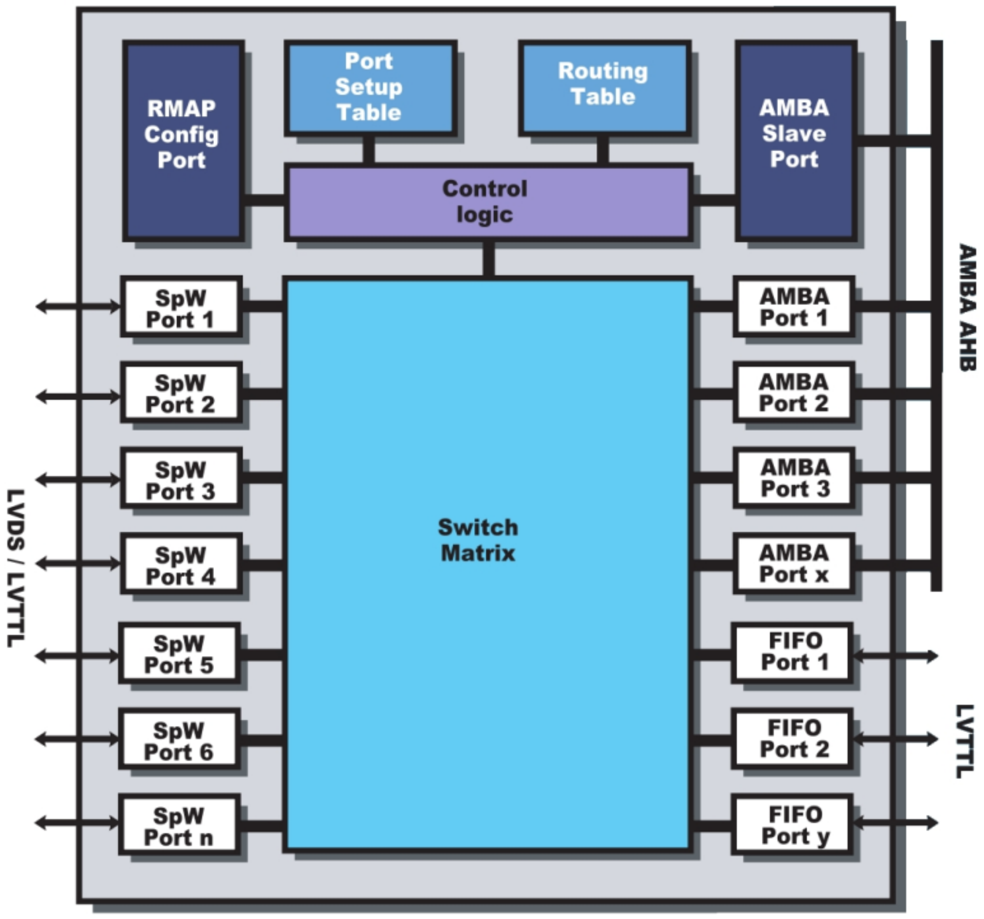


Fig. 1. GRSPWROUTER IP core overview

All mandatory features currently in the ECSS SpaceWire standard are supported by the core as well as some additional key functions not being available in other implementations e.g. packet distribution.

III. Overall Functionality

This section lists the key features that were common to the two potential configurations of the router presented earlier. The list consists of features available in the router IP core as well as external auxiliary interfaces.

The base consists of the 16 SpaceWire ports with on-chip LVDS transceivers. Each router port, regardless of type, is equipped with a timer which can be enabled/disabled. It is used to prevent deadlocks resulting from stalling source or destination nodes which could lock a port indefinitely. This feature might be introduced in the upcoming revision 1 of the SpaceWire standard but is already available in this design.

All addressing modes mentioned in the standard are fully supported. Physical and logical addresses can be individually enabled to use group adaptive routing or packet distribution to any number of physical ports available in the router. The addressing is setup using a routing and port setup table.

The addressing tables and port FIFOs in the router consist of a considerable amount of memory which can experience SEUs and the contents can thus be corrupted. All memory is protected by hardened flip-flops, simplifying the design.

All configuration and status access are handled through configuration port 0 which is accessed using the RMAP protocol from any of the other ports. The allowed ports for configuration accesses can be restricted if needed using several configuration options.

For diagnostic and test purposes UART and JTAG interfaces are provided. These low pin count interfaces are suitable in the small package that will be used (see below) but at the same time have sufficient bandwidth for the amount of status and configuration in the router internals. As this method is available most of the router configuration options have been set to known good values after the reset which can then be changed using these interfaces.

IV. Final Configuration

The final configuration that has been selected for the ASIC consist of the base mentioned in the previous section with 16 SpaceWire ports with on-chip LVDS transceivers and in addition two SpaceWire ports with support for external LVDS transceivers.. The only difference between the two different SpaceWire port types is the I/O type of the pads.

The major design choice for this configuration was whether to include two FIFO ports or two SpaceWire ports. The selection of the two additional SpaceWire ports was motivated by the pin count of the selected package, as well as the fact more and more processor devices have built-in SpaceWire ports (of the with LVTTL signaling) and therefore parallel FIFO ports would not be readily used without the need for an FPGA device between the router and the processor. It is also not that difficult to include SpaceWire link in FPGAs, considering the large variety of SpaceWire IP cores available (see discussion further down).

The target package for the router is a simple to handle low-pin ceramic quad flat package which is quite limiting and does require reducing the amount of configuration pins even more than previously mentioned to fit two FIFO ports. Choosing two additional SpaceWire ports instead saves up to 36 pins without reducing flexibility of the ASIC.

One of the applications of the FIFO ports is to cascade one or more routers without any glue logic. For this purpose the SpaceWire ports will work equally well and would in fact simplify matters. In most cases cascading would be done on a printed circuit board and it is well understood how to route SpaceWire signals on such a board. The FIFO interfaces are most useful when connecting directly to external processors and memories. To use a SpaceWire link instead would require the insertion of glue-logic providing a complete SpaceWire codec which would typically be done using a FPGA which increases design complexity considerably. It is however anticipated that the need to interface to external processors using parallel interfaces will be less required in the future since most processors will be equipped with SpaceWire interfaces.

V. SpaceWIre Standard Revision 1 Support

An upcoming revision 1 of the SpaceWire standard is planned for the near future which contains some changes affecting the router ASIC development. Some additions result in old devices potentially not being forward compatible. It has to be carefully considered if and how these new features are implemented. The final details of the updates have not been decided yet and there is no date set for when this will be ready so there is a considerable risk in implementing these new features before the standard is finalized.

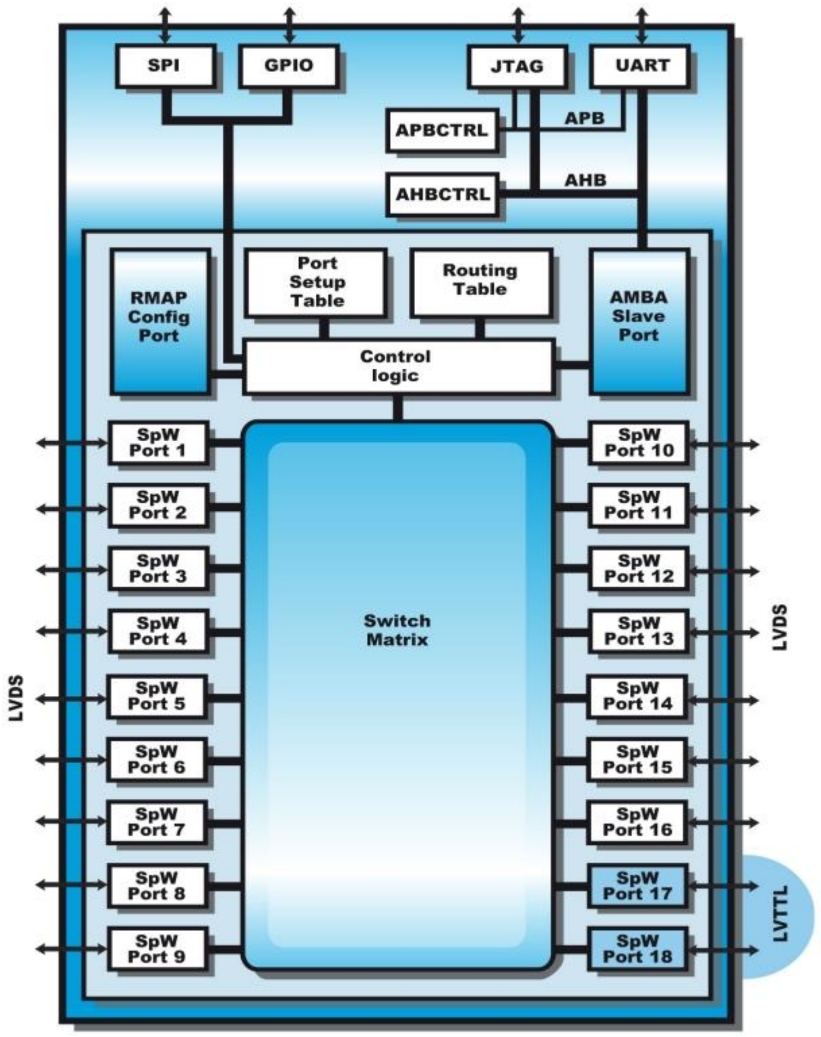


Fig. 2. 18x SpaceWire router ASIC overview

Three changes have been identified as having technical impact. The first one is the addition of timers in routers. This will probably be optional in the standard and not restricting the implementation details to any larger extent. The GRSPWROUTER IP core already contains a timer feature as previously mentioned which makes it probable that no changes will be needed to the core.

The second change is a modification of the link interface FSM. Two requirements have been identified that potentially can cause the codec to make unwanted transitions. These are unlikely corner cases and very few if any problems have been seen in practice. This will probably not affect backward compatibility with old codecs and so the risk is estimated to be very low to include these fixes in the router. Tests will be made during validation on FPGA that no disturbances occur with older devices.

The final and most complicated change is the addition of an interrupt code. It uses one of the reserved control bit combinations of time-codes and it must therefore be made sure that it cannot interfere with the normal time-code facilities. Existing devices might not be forward compatible with revision 1 compliant devices due to the interrupt code. Some issues with these new codes are still under discussion, but the basic specification has been included in the standard. This is indentified as the part of revision 1 causing the highest implementation risk if included in the router ASIC. The desired way to go is that the router is flexible enough to allow ports' handling of the new code to be configured individually. In this way the router can be used as a device enabling old and new equipment to be used in the same SpaceWire network.

VI. SpaceWire-D Support

There is a new protocol emerging called SpaceWire-D where D stands for deterministic. This is anticipated to be widely used in the future to provide deterministic and low-latency transfer of control and command information while still preserving the high bandwidth of SpaceWire. It basically consists of a time-slotting table replicated in each unit (node or router) in the SpaceWire network. Therefore a router needs to have support for SpaceWire-D if it is used in a network utilizing that protocol.

The SpaceWire router ASIC implements the following SpaceWire-D support..

Monitoring of received packet length has been implemented, with the maximum packet length and enable being programmable per port. In the case the length of a received packet exceeds the aforementioned maximum length, the packet will be truncated and an EEP will be inserted to the destination port. The source port spills the incoming packet up to and including the next EOP/EEP. The maximum length is possible to configure up to the maximum length of an RMAP packet thus 2^{25} bytes.

Monitoring of packet reception while receiving a Time-Code has been implemented, enable being programmable per port. In the case a packet is being received while a filtered Time-Code is received as per above, the packet will be truncated and an EEP shall be inserted (in the same way as for packet length truncation). Note that also Distributed Interrupts can be used for truncating packets, being programmable in the router.

VII. SpaceWire Plug-and-Play support

The SpaceWire router ASIC implements basic support for SpaceWire Plug-and-Play, which covers device identification and support for network discovery. The function can be disabled by means of a configuration pin.

VIII. SpaceWire In-System Test

A built-in self-test is provided for the verification of the SpaceWire router and codec functionality. The SpaceWire In-System Test (SIST) protocol provides a means for verifying larger part of the designs' functionality without the need to generate high speed test patterns and observe results at high frequencies.

The internal SIST module is connected to the router via a dedicated FIFO port. The FIFO port is one of the standard ports of the GRSPWROUTER IP. The other side of the SIST module is connected to the AMBA APB bus, which is only accessible through the JTAG and UART (debug-) interfaces. Thus is it is not possible to configure the SIST module via a SpaceWire link.

The SIST module can generate and send SpaceWire packets via the FIFO port. It can also receive SpaceWire packets via the FIFO port and check there contents. The packets are generated deterministically and can therefore also be easily checked on reception.

The packet format is similar to the commands defined for the RMAP protocol (ECSS-E-ST-50-52C):

- SpW Address (0 to 31 bytes)
- Logical Address (1 byte)
- Protocol ID (1 byte)
- Transaction Identifier (2 bytes) (i.e. seed)
- Data Length (3 bytes)
- Header CRC (1 byte as per ECSS-E-ST-50-52C, covering header from Logical Address, inclusive)
- Data (0 to 16 MiB-1) (data is a pseudo-random generated bit string based on the seed)
- Data CRC (1 byte as per ECSS-E-ST-50-52C, covering all Data bytes)
- End-Of-Packet

Packets of up to 2^{24} bytes can be generated and checked. Sequences of up to 2^{16} packets can be generated, or auto repeat can be enabled. The data is generated by means of a 16-bit wide LFSR, with a programmable polynomial. The stated of the LFSR (a.k.a. seed) at the beginning of the data in the packet is transmitted as part of the packet header, allowing each packet to be checked independently. The seed can also be used to detect dropped packets. The length of the packet data field is sent in the packet header. The only managed parameter is the polynomial; everything else can be derived from the packet header.

Packets are automatically generated in an initiator, the contents of a packet is deterministic. Packets are automatically checked in a target when received, providing statistics. The initiator and target are normally the same end-point in a SpaceWire network, but may be different.

It is possible to combine the SIST functionality with the internal loop-back function, or with external cables looping back the SpaceWire signals per port or between pair of ports.

The SIST module also allows direct data read and write to the FIFO port, as well as sending and receiving signaling codes (time-codes and distributed interrupts).

The packet follows the "SpaceWire protocol identification - ECSS‐E‐ST‐50‐51C" format. The SpW Address bytes can be used for path addressing or regional local addressing in a SpaceWire network.

The SIST functionality is protected by means of a protected general on/off register (protection done by expected fixed pattern in data). It is not accessible through SpaceWire RMAP or SpaceWire PnP accesses to configuration port 0. The SIST module can also be clock-gated to save power (default at reset) via JTAG and UART interfaces.

IX. Power-saving

The SpaceWire router ASIC incorporate the following power saving functions:

- Disabling of unused on-chip LVDS receivers/transmitter
- Disabling of unused off-chip LVDS receivers/transmitter or repeater devices

The existing power-down functionality provided for the LVDS I/O cells in the DARE+ library is being utilized.

Signals for disabling the off-chip LVDS devices are shared with the external pins provided by a General Purpose I/O Port. It is possible to control up to 18 external LVDS devices, with one external pin per devices. The control of the external pins is made directly from a ports enable bit in the SpaceWire router configuration registers.

SpaceWire ports that are not in use (i.e. disabled) in the router are also placed in low-power mode by gating the incoming clocks.

X. Technology

The 18x SpaceWire router ASIC will be manufactured in the 180nm UMC CMOS technology, based on the DARE+ (Design against Radiation Effects) library from IMEC (BE). The technology is radiation hard, with at least 300 krad(Si) TID tolerance, high SEL tolerance and SEU hardened flip-flops.

XI. Prototyping

Prototypes for evaluation of the router are already available and are based on a Xilinx Virtex 4 FPGA with an accompanying evaluation board compatible with RASTA. The board provides the possibility to interface both through FIFO ports and the PCI interface depending on the configuration (although a final selection how has been made). All features planned for the ASIC are included and run at full-speed.

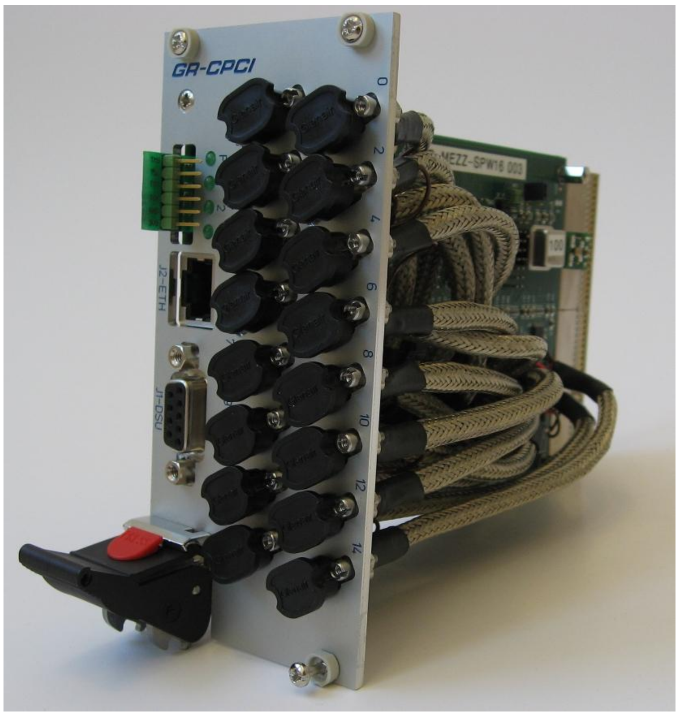


Fig. 3. Prototyping board

XII. Status and Conclusion

The new SpaceWire router ASIC design has at the time of writing been submitted to ASIC layout.

The first ASIC prototypes are expected to go into production in June 2013, with validated parts being available for potential customers in early 2014.

References

[1] European Cooperation for Space Standardization, “Space Engineering; SpaceWire Links, nodes, routers and neworks,” ECSS-E-ST-50-12C, July 2008.

[2] Redant, S.; Marec, R.; Baguena, L.; Liegeon, E.; Van Thielen, B.; Beeckman, G.; Ribeiro, P.; Fernandez-Leon, A. and Glass, B. Radiation test results on first silicon in the DARE library, IEEE transactions on Nuclear Science, VOL. 52, NO. 5, October 2005

Atmel’s New Rad-Hard Sparc V8 Processor Embedding State-of-the-Art SpaceWire

GANRY Nicolas
Aerospace Marketing ASSP and Processors
Atmel Aerospace
Nantes, France
nicolas.ganry@atmel.com

Abstract— The AT6981 is a new generation of processing component designed for critical spaceflight applications, which combines a high-performance SPARC® V8 radiation hard processor, with enough on-chip memory for many aerospace applications and state-of-the-art SpaceWire networking technology from STAR-Dundee. The AT6981 is implemented in Atmel 90nm rad-hard technology, enabling at least 200 MHz operating speed for the processor with power consumption levels around 1W*. The device is ITAR-free being manufactured in a commercial foundry. This paper describes this new processor prototypes of which will be available in late 2013.

I. INTRODUCTION

Building upon the company’s thirty years of innovation in the aerospace market, Atmel will introduce the AT6981 in 2013 a new SPARC® V8 rad-hard processor integrating advanced SpaceWire technology [1] and a SpaceWire router with 8 external SpaceWire ports each supporting link speeds up to 200 Mbit/s. The AT6981 has been developed in collaboration with STAR-Dundee based on their SpaceWire engine IP. The AT6981 runs at 200 MHz with a target for low power consumption around 1W*. Atmel will present this new standard space processor during the 2013 International SpaceWire Conference at the same time as the presentation of the STAR-Dundee SpaceWire engine [2].

II. ATMEL’S UNRIVALLED FLIGHT HERITAGE

Over the last 16 years, Atmel has steadily built a space microprocessor strategy based on the SPARC architecture. With worldwide sales of over 3000 flight models featuring the Atmel TSC695F and already over 600 flight models with the Atmel AT697F, Atmel’s SPARC processor roadmap has an unrivalled flight heritage. The upcoming AT6981 rad-hard SPARC V8 processor benefits from this solid experience.

III. AT6981 SHORT DESCRIPTION

The AT6981 is based on the rad-hard LEON2FT processor, it integrates all commonly-used space peripherals including 1553, CAN, SPI, UART, DSU and Ethernet. The device embeds a fully-compliant IEEE754 FPU without truncation as well as an MMU native to the SPARC processor. The SoC integration is done in 90nm rad-hard Atmel technology, enabling at least 200 MHz operating speed for the processor with power consumption level around 1W*. Atmel has leveraged its significant rad-hard experience to develop dedicated rad-hard libraries for fabrication in a 90nm commercial foundry, thus securing a multi-source supply chain and insuring an ITAR-free product design. Atmel continues to offer best-in-class power-to-performance ratios that offer more possibilities for space applications by reducing costs, sizes and embedded power supply.

The AT6981 embeds three SpaceWire engines allowing the concurrent transmission of three SpaceWire packets and at the same time concurrent reception of three SpaceWire packets. These state-of-the-art SpaceWire engines offload the communication tasks from the processor. They each support the SpaceWire Remote Memory Access Protocol (RMAP) [3] as both an Initiator and Target device and support other protocols with a selective DMA controller. SpaceWire Plug-and-Play [4] and SpaceWire-D [5] protocols are supported and full time-code support is included. The embedded SpaceWire router has 12 ports: eight external SpaceWire ports, three ports to the SpaceWire engines and a configuration port. LVDS drivers are included on chip for the SpaceWire interfaces. The AT6981 benefits from the close collaboration between STAR-Dundee and Atmel on the design to achieve an embedded system with high processing power and excellent interfacing capabilities.

The AT6981 will be available in 256 MQFP and in 349 LGA ceramic packages.

All embedded IPs belong to Atmel’s proprietary portfolio dedicated to aerospace applications that includes IP such as SpaceWire and 1553. Just as the SpaceWire IP was developed in partnership with STAR-Dundee, the 1553 IP was developed with Maya Technologies and has been proven in-flight in space applications.

IV. AT6981 Key Features

In addition to a powerful SPARC V8 processor core with a high level of integration and performance, the AT6981 embeds a 1-Mbyte hardened SRAM memory with EDAC for PCB area savings and fast access at full CPU speed. It also features SRAM and DDR1 interfaces as external memory. The overall power consumption of the device with embedded memory is targeted around 2W worst case.

In order to facilitate analog-to-digital operations and provide an even higher level of integration, the AT6981 embeds a dedicated waveform generation (PWM) unit for analog control/command, as well as several ADC/DAC interfaces for analog acquisition/conversion. Those functions are really useful for engine control management and for measurements control. PWM unit is programmable and ADC/DAC digital interface is done in the same way as the AT7913 device. Having this digital part integrated in the SoC reduces the need to use external an FPGA in order to connect analog ADC or DAC.

The AT6981 is the newest device in the Atmel SPARC V8 portfolio. Compared to the AT697F and the AT7913 RTC, the AT6981 offers more performance with an operating speed of 200MHz and a higher level of system-on-chip integration with embedded memory, SpaceWire router, 1553 and Ethernet.

The AT6981 is a rad-hard by design processor that will be space-qualified and will support:

Total dose of 300Krads (Si) according to the MIL-STD883 method 1019

- SEU error rate better than 1 E-5 error/device/day
- No Single Event Latch up below a LET threshold of 70 MeV.cm²/mg

V. AT6981 Architecture

The architecture of the AT6981 device is illustrated in Figure 1.

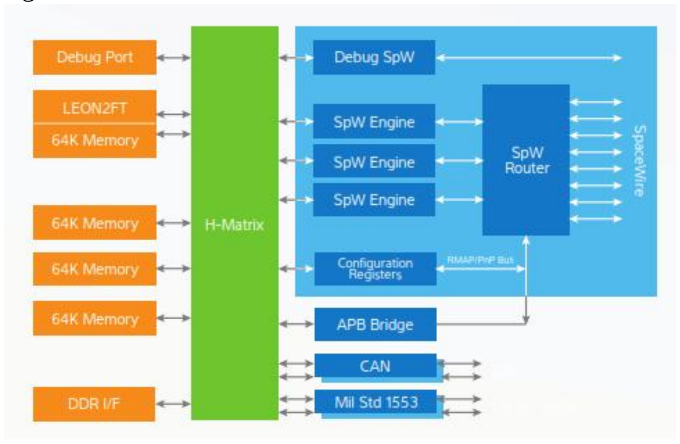


Figure 1 AT6981 Architecture

The AT6981 comprises a SPARC® V8 processor, several banks of memory, comprehensive SpaceWire network capability and various other interfaces. More details on the main AT6981 features are provided in the following subsections:

A. Processor

AT6981 CPU core is a Sparc V8 running at 200 MHz, it uses the LEON2FT core from ESA. This core is already embedded in several space missions with the AT697F from ATMEL. Native MMU of the SparcV8 architecture is activated and a powerful FPU is added which gives to AT6981 the best processing performances on the market today. This one CPU core device allows an easy and safe migration of the software from AT697F without compromise performances. AT6981 benefits from all development tools available for LEON core as it offers a standard DSU interface for trace and debug.

B. Hmatrix

The AT6981 bus architecture is unique on space market. This device takes benefit from Atmel strong IP portfolio and powerful architecture coming for the commercial microcontroller business where Atmel is one of the leaders today.

The AT6981 System on Chip is built around a HMatrix bus which is multi AHB compliant and brings some AHB arbitration mechanisms to support multi threading. By this well proven Atmel technology, conflicts management for concurrent access is becoming much easier, even completely transparent for the CPU core running software.

For example, you can manage in parallel all those activities without loading the main CPU core:

- Run three Space Wire 200Mbit/s transfer
- Run two 1553 communication flow
- Run two high speed CAN transfer
- Run a MAC Ethernet 100Mbit/s connection
- Run a SPI or TWI session as well

Each peripheral is connected to its own protected memory area and can take benefit from the 200MHz x 32bits AHB bus bandwidth without disturbing CPU internal operations. During full speed transfer session, processor is never interrupted and has a fully deterministic behavior to manage control of all operations.

This architecture, which provides up to 6.4 Gbit/s bandwidth, is ready for targeted future evolution like SpaceFibre, Gbit Ethernet and multi-core. It will enable a smooth transition for coming product derivatives of this high speed SPARC® V8 architecture.

C. SpaceWire

The AT6981 includes three SpaceWire engines each of which has dedicated RMAP target and initiator hardware which offloads RMAP packet generation and checking from the processor. The RMAP target can be configured to allow a remote unit to read and write memory locations inside the processor memory space without interrupting the host software. The RMAP initiator facilitates access to remote memory spaces through RMAP protocol commands and offloads the generation of multiple transactions and the reply packet checking from the processor.

From Hmatrix, a multi-channel DMA packet transmission and reception controller is available to the processor to send and receive data through a SpaceWire router. The DMA channels are optimized to support high throughput of SpaceWire packets with minimal interruption of the processor. Generation and checking of CRC-8 and CRC-16 checksums are supported by the DMA channels.

Packets are routed to the SpaceWire network through a SpaceWire router with eight external SpaceWire ports. This allows the AT6891 to connect to many peripherals and also act as a routing device. Protocol support is provided for the SpaceWire-D deterministic data delivery protocol [5], the SpaceWire plug and play protocol [4], multiple time-code counters and distributed interrupt time-codes [6].

D. Low power consumption

AT6981 is a low power consumption device with dedicated mechanism in order to adapt the power consumption to the level of application complexity. Those mechanisms are:

- GEN clock programmable block delivering clock for each IPs and peripherals. Clock speed can be changed and gated
- Dedicated reset per IPs in order to reinitialize them locally after the clock coming ON.

E. Rad Hard by design

All internal memories have a dedicated scrubber with internal EDAC in order to manage auto correction.

This scrubber is fully programmable on period of the scrubbing cycle and the protected RAM array. It is an additional value to the external EDAC capability provided with the 1Mbytes of on chip available high speed SRAM to allow customer own correction management.

All Memory blocks are designed in a way to never have any adjacent bits for a same word. This technique simplifies strongly the error management activities which allow using only a simple EDAC for data single event protection. By this way it's not needed to implement an heavy TMR mechanisms to protect register files which trigger some potential performances limitation.

TMR mechanisms are implemented on all logic of the design with also an SET filtering method.

Rad hard libraries on this proposed 90nm technology are developed by Atmel in France based on all well proven libraries from Atmel commercial products. AT6981 benefits from the strong 30 years' experience of Atmel France in rad hardening techniques.

VI. Detailed Block Diagram

A more detailed block diagram of the AT6981 is provided in Figure 2.

The AHB H-Matrix is at the heart of the AT6981 device connecting the processor, memory banks, SpaceWire engines and other IO functions. Several internal RAM blocks are provided to support concurrent memory accesses by the processor and IO facilities.

The three SpaceWire engines, Ethernet, CAN and MILSTD 1553 interfaces are all connected as master devices to the H-Matrix allowing them to read and write to the memory using distributed DMA capability.

The lower speed peripheral devices including SPI, TWI, UART, timers, watchdog timers, PMW, ADC interface, DAC interface, parallel input/output and interrupts, are connected via an APB bus and peripheral bridge to the H-Matrix.

Various forms of external memory (PROM, SRAM, SDRAM and DDR) can be attached directly to the AT6981 devices, providing ready of expansion of the internal memory when required.

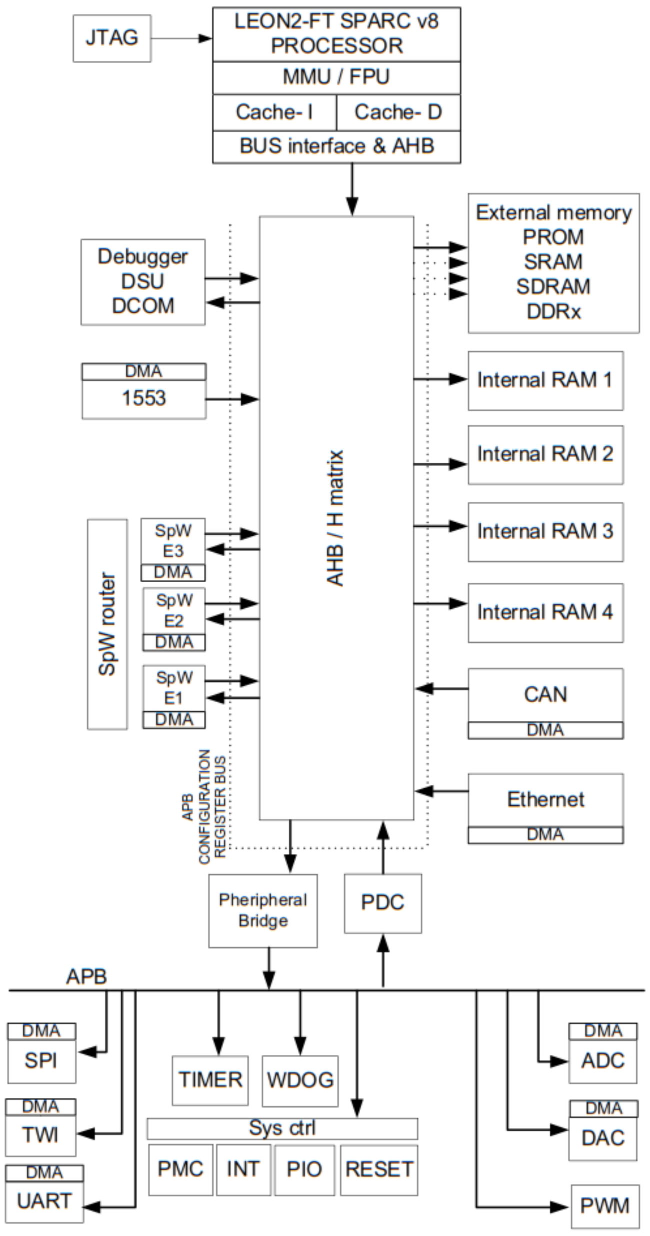


Figure 2 Detailed Block Diagram of AT6981

VII. AT6981 Software, Tools and Services

With the AT6981, Atmel will offer an ecosystem of software and tools that will be used by Atmel for the full chipset validation and qualification. This guarantees the best starting point for development. A full package that includes a hardware reference board with associated software drivers in addition to a Software Development Environment (SDE) for debug and trace will be proposed to customers. The SDE is provided by STAR-Dundee and supports the well-known DSU interface.

The set of embedded software drivers is the same that the one which is provided with each Atmel component. It's a highly modular package which includes a hardware abstraction layer to simplify hardware changes, limiting the impact on software and reducing efforts for later upgrade.

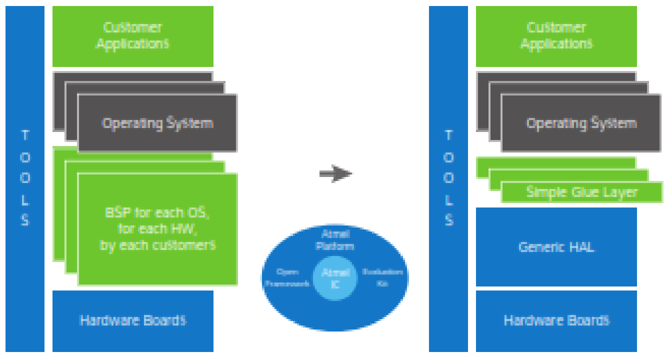


Figure 3 AT6981 Delivery Package

By taking advantage of the hardware and software building blocks available with the AT6981, our customer is able to manage his own system design, improve targeted application time-to-market and be compatible with many other services that will be proposed by Atmel and partners.

VIII. AT6981 schedule

The AT6981 is in its final stages of development and first samples will be available in Q4 2013. Flight models are targeted to be fully space-qualified in 2014. QMLQ & QMLV qualtity grades will be proposed for flight models. Early development starting in Q3 2013 can be based on the simulation model or on the FPGA set-up provided by Atmel.

IX. Conclusions

The AT6981 device is the first product of the partnership between STAR-Dundee and Atmel, enabling integration of state-of-the-art SpaceWire technology into Atmel products.

Providing integration of more peripherals and memory blocks around the SPARC V8 processor core enables size, weight and cost improvements for today's space applications: on-board computing, telemetry/telecommand, remote terminal units, sensors, instruments and payloads. In addition its high level of system integration, the AT6981 offers more powerful processing with 200MHz and embedded fast memory to complement the higher bandwidth capabilities of peripherals with SpaceWire 200Mbit/s.

The AT6981 architecture based on Rad Hard 90nm is the starting point for further evolution; evolution through higher performance by replacing SpaceWire by a SpaceFibre IP, by Gbit Ethernet and by adding an additional CPU and/or DSP core, evolution through more flexibility by adding a programmable area inside the SoC in order to allow better customization for the targeted space applications; and evolution through dedicated design by considering this architecture and IP cores as a starting point for your own ASIC design.

References

[1] ECSS Standard ECSS-E-ST-50-12C, "SpaceWire, Links, Nodes, Routers and Networks", Issue 1, European Cooperation for Space Data Standardization, July 2008.

[2] Chris McClements, Steve Parkes, Albert Ferrer and Alberto Gonzalez Villafranca, "High performance SpaceWire RMAP/DMA engine for the CASTOR microprocessor", Paper 75, International SpaceWire Conference, Gothenburg, June 2013.

[3] ECSS, "SpaceWire - Remote memory access protocol", ECSS-E-ST-50-52C, Feb 2010.

[4] P. Mendham, S. Parkes, "SpaceWire Plug-and-play: a Roadmap", International SpaceWire conference, Nara, Japan, 2008.

[5] S. Parkes, Albert. Ferrer, S Mills, A Mason, "SpaceWire-D: Deterministic data delivery with SpaceWire", International SpaceWire conference, Russia, 2010.

[6] Yuriy Sheynin, Sergey Gorbatchev, Liudmila Onishchenko, "Real-Time Signalling in SpaceWire Networks", International SpaceWire conference, Nara, Japan, 2008.

* Under the same conditions of measurement as current AT697F devices

Reliability Study of Over/Under Voltage for LVDS Physical Layer of SpaceWire

Long Paper

Jennifer Larsen
Aeroflex Colorado Springs
4350 Centennial Blvd
Colorado Springs, CO 80907
Larsen@aeroflex.com

Rob Ciccariello
Aeroflex Colorado Springs
4350 Centennial Blvd
Colorado Springs, CO 80907
Ciccariello@aeroflex.com

Abstract— The SpaceWire Standard ECSS-E-ST-50-12C calls for a Low Voltage Differential Signaling (LVDS) physical layer as defined in ANSI/TIA/EIA-644, Electrical Characteristics of Low Voltage Differential Signaling Interface Circuits. Extensive reliability studies by Aeroflex Colorado Springs catalog the effects of operation outside the recommended operating conditions on Aeroflex cold-spare LVDS drivers, receivers, and SpaceWire devices built on the Aeroflex 0.25µm technology node. These reliability studies focus on a hypothetical failure where the LVDS I/O voltage and switching frequency exceed the ABSOLUTE MAXIMUM RATINGS defined in the Aeroflex Datasheet and corresponding Defense Logic Agency (DLA) Standard Microcircuit Drawing.

This paper describes the operation and reliability effects on the 3.3V LVDS drivers, receivers, and SpaceWire devices with cold spare buffers, focusing on current and voltage excursions during overshoot/undershoot events on the LVDS I/O. Hot Carrier Injection, Electromigration, and oxide wear out are also examined.

I. LVDS Operation

LVDS is a method used to transmit and receive hundreds of megabits per second over differential media using a low voltage signal swing (~350mV). LVDS communications are performed by a driver and a receiver. The driver accepts a standard Complementary Metal Oxide Semiconductor (CMOS) signal and outputs a constant current, differential, signal. The LVDS receiver senses the differential voltage across a 100Ω termination resistor and outputs a standard CMOS signal equivalent to the supply voltage. The differential aspect of LVDS allows systems to run at high data rates, with low switching power, high noise immunity, and relatively wide common mode range.

The LVDS driver works by using NMOS Field Effect Transistors (FETs) to control the direction of the constant 3.5mA current source through the termination resistor. The driver current, flowing through the 100Ω termination resistor placed across the differential inputs of the receiver, generates a +/-350mV I-R drop which is sensed as a logic high/low by the receiver. The LVDS receiver has very high DC input impedance, virtually all of the driver's current flows, in a loop, from the source terminal through the 100Ω termination resistor and back into the sinking terminal. As the current flow direction through the termination resistor changes, a logic 1 or logic 0 state is created at the receiver output.

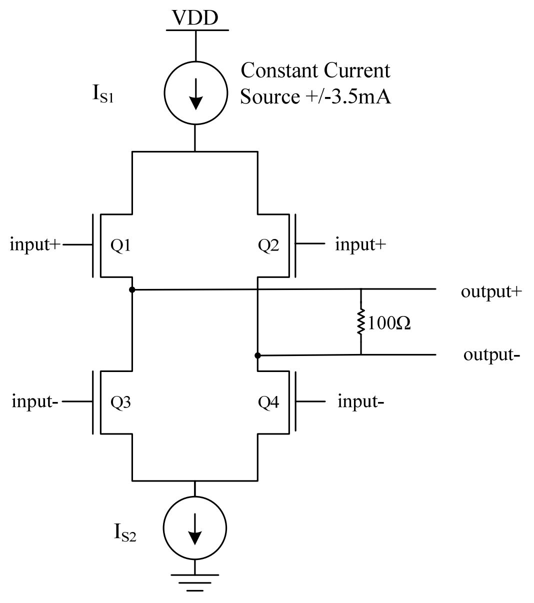


Figure 1. Simplified LVDS Output Driver Schematic

Faults in SpaceWire systems can be caused by many factors including, but not limited to, system requirements, poor termination, edge rate of the signal, environmental effects, and human error. The three fault scenarios examined in this paper are: undervoltage on LVDS I/O, overvoltage on LVDS I/O, and switching frequency, above 200MHz (400Mbps) specification of 0.25µm LVDS/SpW devices.

II. Aeroflex 0.25µm LVDS and SpaceWire Technology Node

The LVDS drivers and receivers used in all the devices listed in Table 1 are very similar in structure and layout.

Table 1. Aeroflex Colorado Springs devices built on 0.25µm technology node

Aeroflex Part Number	SMD Number
UT54LVDS031LV/E	5962-98651
UT54LVDS032LV/E	5962-98652
UT54LVDS032LVT	5962-04201
UT200SpWPHY01	5962-06232
UT200SpW4RTR	5962-08244

Aeroflex Colorado Springs defines “operating life” as an average failure rate over the 15 year operating life as less than 10 FITs (i.e. 1×10^{-9} hr^{-1}) total for all wear out mechanisms at the worst case operating conditions for each mechanism. The reliability models involve understanding the physics of failure of the mechanisms that concern the 0.25µm CMOS technology. These include reliability with respect to gate dielectrics and reliability related to metallization. Other reliability concerns, including environment, ESD, latch-up, and radiation are considered the principally package or design related and are discussed in particular product qualification plans.

Gate dielectric reliability is the dominant concern for CMOS technology. This includes breakdown and charge trapping mechanisms such as hot carrier. Metallization concerns center on electromigration. To calculate the combined effect of all these concerns, a series model is used, with the fail rate for each mechanism taken at worst case conditions.

III. Undervoltage/Undershoot on LVDS Input (Electromigration)

A negative voltage on an LVDS receiver input can result in high current due to the turn on of the input to ground ESD protection diode. This could be caused by poor cable connections between the driver and receiver, see figure 3. Although Aeroflex Colorado Springs LVDS inputs are qualified to HBM ESD class 1 (1000V), an extended time at a voltage above the ESD diode turn on could result in damage to the interconnect metallization between the input pad and ground. The mechanism that causes interconnect damage is known as Electromigration (EM). EM is caused by momentum transfer from electrons to the metal atoms in a conductor during current flow. Over time at stress, enough momentum is transferred to cause metal atoms to migrate, which can ultimately result in metal voiding (high resistance interconnects) or buildup of metal that could result in line to line shorting. EM is accelerated by both increased current density and increased temperature, and follows Black’s equation as shown in the following equation:

$$ttf_{use} = ttf_{stress}\left(e^{\left[\frac{-Ea}{k}\left(\frac{1}{T_{stress}}-\frac{1}{T_{use}}\right)\right]}\right)\left(\frac{J_{stress}}{J_{use}}\right)^{n}$$

The subscript use refers to use conditions and the subscript stress refers to stress conditions, T refers to temperatures, J refers to current density, k is Boltzmann’s constant, E_a is the activation energy, and *n* is the current density exponent.

Once the ESD protection diode is turned on, (~-0.5V) the resistance of the receiver essentially drops to 0, and the current through the ESD protection is completely driven by the output resistance of the supply driving the voltage. For this example, consider the case where the under voltage is supplied by the LVDS driver, as would be the case for ringing caused by poor cabling connections.

We assume the UT54LVDS031LV/E 3.3V quad LVDS driver is connected to a UT54LVDS032LV/E receiver, and that undershoots of varying voltage and duration occur. The quad driver has an output resistance of ~300Ω in short circuit conditions. Based on layout information on the UT54LVDS031LV/E and electromigration test structure data collected through technology qualification, we can use the equation above to calculate part lifetime for a range of undershoot voltages. Figure 2 shows estimated mean time to failure for the receiver interconnect under a range of undershoot voltages at 125°C.

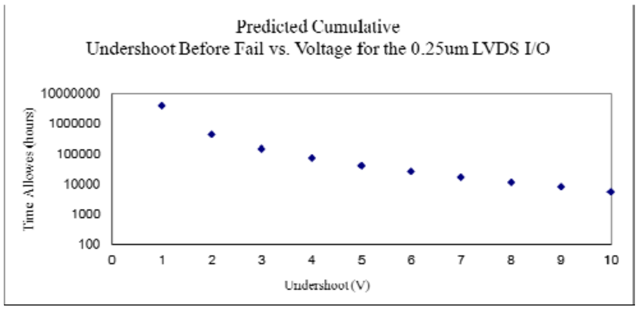


Figure 2: Predicted Cumulative Undershoot before Fail vs. Voltage for the UT54LVDS031LV/E LVDS inputs

A caveat to this graph is if -10V was held continuously on an LVDS input, joule heating would cause the temperature to rise, resulting in significant reduction of life time for the device. Overall, however, this data shows wide margin for undershoot in use conditions. For example assume that at maximum frequency (200MHz) a 3V undershoot were to occur for 1ns every cycle, the chart above would predict that the input could survive for greater than 80 years at 125C.

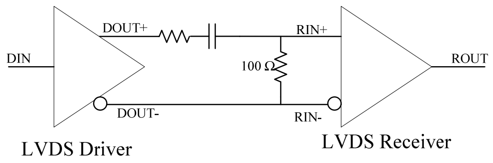


Figure 3. Impedance mismatched undershoot case.

Assuming a case where the LVDS Driver and LVDS Receiver are poorly terminated and the cable media is impedance mismatched, see figure 3. Based on layout and simulation information on the UT54LVDS031LV/E a 1.0V undershoot results in an electromigration acceleration factor of 1.57. This acceleration factor coincides with a 1FIT max undershoot time of 2,052,504.6 hours or a mean time to failure, MTTF, max undershoot of 3,862,096.1 hours.

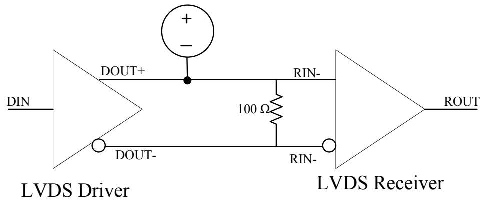


Figure 4. One time undershoot event.

Another case of an undershoot/undervoltage on the LVDS I/O would be if there was a one-time event where a negative voltage was driven onto the LVDS lines, see figure 4.

IV. Overvoltage/overshoot on LVDS I/O (Oxide Wear out)

The cold spare feature of Aeroflex LVDS inputs allows the user to apply active input signals to the LVDS I/O with VDD grounded. Because of this, standard ESD protection diodes between input pad and VDD are not viable. The Aeroflex proprietary cold spare ESD protection is designed to shunt the current associated with a HBM event, and has passed qualification up to 1000V. In the case of a longer duration overshoot, the voltage at the pad is applied directly across the input gate oxide. Breakdown of this oxide would likely result in shorting of the input to ground and catastrophic failure.

Oxide breakdown can be split into two types, instantaneous and long term wear out. Aeroflex does not recommend operation above the absolute maximum for its parts, but data shows that the gate oxide can withstand > 10V without instantaneous breakdown. Long term wear out occurs on all oxides placed under electric field stress. This is also known as Time Dependent Dielectric Breakdown (TDDB). TDDB is caused by the buildup of trapped charge in a dielectric due to electric field stress. Over time at stress, enough trapped charge may build up such that somewhere in the dielectric, the local electric field exceeds the critical field for breakdown. TDDB is accelerated by both increased electric field and increased temperature, and follows a generalized Erying model as shown in the equation that follows:

$$\frac{ttf_{use}}{ttf_{stress}} = \left[\left(e^{\left(\frac{-E_a}{k_B}\left(\frac{1}{T_{stress}}-\frac{1}{T_{use}}\right)\right)}\right)\left(e^{\left(-r\left(\frac{V_{stress}}{t_{ox}}-\frac{V_{use}}{t_{ox}}\right)\right)}\right)\right]$$

Where ttf_{use} and ttf_{stress} are the times to failure under use and stress conditions respectively, T_{use} and T_{stress} are the absolute temperatures (in °K) of the dielectric under use and stress conditions respectively, V_{use} and V_{stress} are the use and stress voltages respectively that appear across the dielectric thickness of t_{ox}, E_a is the (Arrhenius) thermal activation energy for dielectric breakdown in the particular dielectric materials of interest, r is a model parameter for the electric field acceleration of dielectric breakdown for the particular process and dielectric material of interest, and k_B is Boltzmann's constant (8.62 x 10^{-5} eV/°K).

Again, since these are cold spared LVDS I/O parts, meaning that the I/O structure is non-typical [8], the effect of an overshoot is on the input oxide. With the supply voltage of any of the devices listed in Table 1 set to 3.6V (maximum recommended supply voltage) an overshoot of +1.0V is allowed for approximately 346,106.6 hours.

V. Increased Switching Frequency (Hot Carrier Injection)

As part of this reliability study Aeroflex completed an assessment on a device manufactured on the 0.25µm Aeroflex technology node when there is continuous operation at 50MHz above its 200MHz specified max switching limit.

The main reliability concern for operation above the specified maximum operation frequency is the risk of hot carrier ionization (HCI) resulting in increased threshold of the NMOS transistors in the high frequency path. HCI occurs during switching, when the transistors conduct peak drain currents (shoot through current) and results in worst case degradation when the gate-source voltage (V_{GS}) is ~1/2 the drain-source voltage (V_{DS}). HCI is strongly dependent on transistor length, with smaller lengths resulting in greater degradation. HCI results in the ionization of atoms in the channel [9] which can result in charged particles becoming trapped in the gate oxide, which degrades transistor performance. The area of concern is the LVDS output driver; a simple schematic is shown in Figure 1.

The Aeroflex Colorado Springs implementation of LVDS Drivers design uses NMOS transistors [8] as in the schematic shown in figure 1. Review of the datasheet specifications for the LVDS devices built on the 0.25µm technology node, [4][5][6][7] using the specified 100Ω output load, show that this circuit is designed to have a maximum differential output voltage of ~400mV (V_{OD}) and a maximum offset voltage (V_{OS}) of 1.450V. Assuming zero voltage drop across the current source, 2.0V can be taken to be equivalent to the maximum V_{DS} voltage across any transistor. The specification also defines the maximum output rise and fall time (t_{LHT} and t_{HLT}) as ~600ps. Based on a part continuously running at 250MHz required frequency (equivalent to 500Mbs), the driver will be switching ~30% of the time. Because of the differential nature of the output, each transistor sees switching conditions ~15% of the time. To calculate the effect of hot carrier, Aeroflex Colorado Springs makes the conservative assumption that worst case bias conditions (V_{DS} at 2.0V, and V_{GS} at 0.5(V_{DS})) are held throughout the switching time.

The LVDS Drivers built on the TSMC 0.25µm process technology use 3.3V transistors. This process has been evaluated by Aeroflex Colorado Springs and has been qualified to QML-V levels. [10] TSMC characterizes HCI on discrete transistors by holding the transistors in a saturated on state at highly accelerated drain voltages at 25ºC. Transistors are measured at regular intervals to determine the effect on threshold voltage and saturated current and time to fail is defined to be the point at which saturated drain current shifts by 10%. HCI is also worst case at cold temperatures, so degradation at -55ºC was considered. Review of the model predicts a degradation of ~0.1% transistor saturated current at 15 years. This is significantly less than the 10% limit, and thus, has a negligible effect on operation. Operating Aeroflex LVDS Drivers built on 0.25µm can maintain 250MHz (500Mbps) continuous use without significant degradation at 15 years.

VI. Conclusion

Aeroflex LVDS I/O built on the 0.25µm technology node are capable of handling a +/-1.0V over/undershoot without compromising a 15 year mission life. The LVDS I/O can also sustain operation at 250MHz. The results discussed are not guaranteed by Aeroflex. Any operation outside of the ABSOLUTE MAXIMUM RATINGS, as stated in the datasheet and/or SMD may affect device reliability and performance.

VI. References

[1] Telecommunications industry Association, "Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits ANSI/TIA/EIA-644", January 30, 2001.

[2] IEEE P1355, "Standard for Heterogeneous InterConnect (HIC) IEEE 1355-1995", Conference Title, Location, June 12, 1996.

[3] ESA Publications Division, "SpaceWire Standard Document ECSS-E-ST-50-12C", The Netherlands, July 31, 2008.

[4] Aeroflex Colorado Springs, "UT54LVDS031LV/E 3.3-VOLT QUAD DRIVER Datasheet", Colorado Springs, Colorado.

[5] Aeroflex Colorado Springs, "UT54LVDS032LV/E 3.3-VOLT QUAD RECEIVER Datasheet", Colorado Springs, Colorado.

[6] Aeroflex Colorado Springs, "UT200SpWPHY01 SpaceWire Physical Layer Transceiver Datasheet", Colorado Springs, Colorado.

[7] Aeroflex Colorado Springs, "UT200SpW4RTR SpaceWire 4-Port Router Datasheet", Colorado Springs, Colorado.

[8] Aeroflex Colorado Springs, "Cold Spare Functionality of the LVDS Family", Application Note

[9] EDN, "Survival guide to high-speed A/D converter digital outputs", Jonathan Harris, 2012

[10] Aeroflex Colorado Springs, "Reliability Assessment: TSMC .25u CMOS"

[11] Aeroflex Colorado Springs, "Theory of Operation and VDD Fault Scenario Application Note", Colorado Springs, Colorado.

European LVDS Transceiver Development

Session: SpaceWire components, Long Paper

Fredrik Sturesson, Sandi Habinc
Aeroflex Gaisler AB
Kungsgatan 12, SE-411 19 Göteborg, Sweden
fredrik@gaisler.com, sandi@gaisler.com

Jan Wouters, Steven Redant
IMEC
Kapeldreef 75, 3001 Leuven, Belgium
jan.wouters@imec.be, steven.redant@imec.be

Jørgen Ilstad
European Space Agency
Keplerlaan 1, 2220AG Noordwijk ZH, The Netherlands
jorgen.ilstad@esa.int

Abstract— **Two LVDS components are currently being developed by Aeroflex Gaisler (SE) and imec (B) under ESA contract 4000105762. The targeted technology is UMC 180 nm using the DARE library from imec (B) which has been extended and enhanced specifically for this development.**

Index Terms—**Component, Radiation, LVDS, SpaceWire.**

I. Introduction

One of the key elements in SpaceWire communication is the low voltage differential signaling (LVDS) [1] defined to be the physical signal level by the SpaceWire standard [2]. LVDS provides the ability for communication with high signal integrity and high speed from board-to-board and equipment-to-equipment in a spacecraft system.

Many ASICs and FPGAs implementing the SpaceWire protocol do not provide LVDS interfaces. Hence, there is a need for external LVDS components translating between single-ended signals and LVDS signals both for the transmitter and for the receiver side. Even in cases where the LVDS interface are implemented in the ASIC/FPGAs, external stand-alone LVDS components may be attractive to achieve higher system robustness; a stand-alone LVDS component with higher voltage tolerance can prevent failure propagation from outside of the equipment via the LVDS interface into the ASIC/FPGA which often implement the most critical functionality in a system design.

In an ongoing development, funded by the European Space Agency (ESA), two LVDS components are developed addressing both these needs: one is a dual transceiver and the other one is a 4x4 cross-point switch. Both components, the latter when configured as a quad LVDS repeater, provide all signals needed to host one SpaceWire channel in one single component and package. The Dual transceiver translates single-ended signals to LVDS signals and vice versa while the 4x4 cross-point switch provides LVDS to LVDS signals.

These functions are today available in commercial LVDS devices, but the high reliability requirements and harsh radiation environment in space applications have motivated us to this new development. Critical characteristics have been addressed, such as Single Event Transient (SET), Single Event Upset (SEU) and Latch-up (SEL) immunity, Total Ionizing Dose (TID) hardness, Extended Common Mode range (ECM) and Failsafe protection of receivers, cold sparing for redundancy purpose, high voltage and ESD tolerance and excellent channel-to-channel timing.

This paper is organized in the following manner: In the Product Specification section the preliminary specification of both products is reported. The following section discusses the main characteristics of the LVDS devices. Finally, the radiation performance is discussed including results from simulation.

All reported results come from simulations performed on layout level including package parasitic when applicable. Thus, it is expected to reflect the performance of the final products. This will be confirmed under the upcoming prototype evaluation stage of this development.

II. Product Specification

Both products are supplied with one single power supply of 3.3V±10%. The operational temperature ranges from -55°C to +125°C.

The LVDS input signals and LVDS output signals are fully compliant to the LVDS standard [1]. The LVDS inputs are implemented without internal termination resistors. This allows the user to match his termination to the actual characteristics of his transmission line. The LVDS inputs are implemented with active fail-safe functionality and they support an extended common mode range of -4.0V to 5.0V.

All single-ended signals are fully compliant with the LVTTL and LVCMOS standard [3]. In addition, the inputs support 5V TTL input signals.

The single-ended inputs and outputs and the LVDS inputs and outputs support cold sparing. Thus, cold redundant devices may share the same signals as active devices.

A. Dual Transceiver

The dual transceiver is functionally equivalent to Texas Instrument's DS90LV049 [4] but it comes with a different package and pin-out configuration. The functional diagram and connection diagram of the dual transceiver is provided in Fig. 1. : R_{IN1+}, R_{IN1-}, R_{IN2+}, and R_{IN2-} are LVDS input signals; D_{OUT1+}, D_{OUT1-}, D_{OUT2+}, and D_{OUT2-} are LVDS output signals; D_{IN1}, D_{IN2} EN, EN\ are single-ended input signals; and R_{OUT1}, $R_{OUT2,}$ are single-ended output signals. The AND-function of the two enable pins (EN and EN\) allows for one single signal of any polarity to enable or disable all LVDS and LVTTL outputs. Thus, an inverting function of an already available signal from e.g. an ASIC will never be needed with this product.

The package of the dual transceiver is a hermetically sealed 16-pin dual-lead flat package with 1.27mm pitch. The package complies with the military standard case outline drawing CDFP3-F16 [5]. This is the most common case outline of small pin-count devices for space applications. Thus, well-established processes with proven high reliability can be used for assembly to printed circuit boards (PCB).

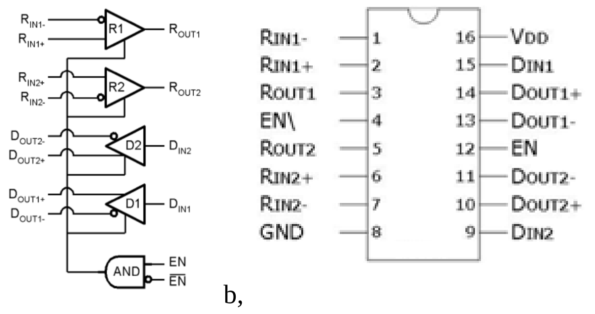


Fig. 1. Functional diagram (a) and connection diagram (b) of the dual transceiver

B. *4x4 Cross-point Switch*

The 4x4 Cross-point Switch will be functionally equivalent to Texas Instrument's SN65LVDS125A [6] however it will come with a different package and pin-out configuration (the design of pin-out and functionality is still preliminary). The functional diagram and connection diagram of the dual transceiver is provided in Fig. 2. The 4x4 mux is controlled by eight select signals (S10-S41): two select signals per LVDS output channel (iY/iZ) select any of the four LVDS input channels (jA/jB) as its input. It is one enable control signal (iDE) per each LVDS output channel.

The package is under development. It will be a hermetically sealed 40-pin dual-lead flat package with 0.635mm pitch. By halving the pitch dimension, this package outline will have similar size as the dual transceiver. The preliminary outline dimensions are 6.0mm x 14.0 mm x 2.4mm.

This package has two more pins than the SN65LVDS125A device. The additional two pins might be used as mode select pins mimicking the function of other LVDS devices on the market like e.g. a quad single-ended–to-LVDS driver (SN55LVDS31) and/or it can be used for power-down control of unused LVDS channels.

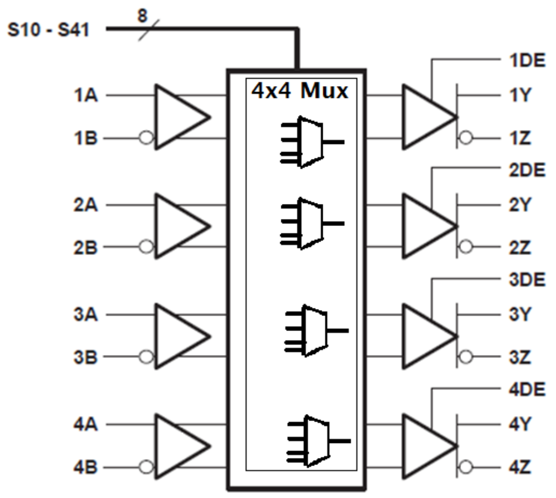


Fig. 2. Functional diagram of the 4x4 cross-point Switch

III. Key Characteristics

A. *Pin Configuration*

The target application of the Dual transceiver is to provide single-ended to LVDS conversion for all signals of one SpaceWire port within one single package.

The pin-out configuration has been defined in order to best match the SpaceWire connector standard [2]; The RX signals are provided on one side of the package and the TX on the other side; this is the same configuration as the SpaceWire connector (illustrated in Fig. 3.). Another benefit with this pin-out configuration is that a cold redundant component in an area efficient manner can be placed on opposite side of the printed circuit board (PCB). With the top of one of the device placed towards the bottom of the other device placed on the opposite side of the PCB (pin no.1 meets pin no.8 etc.). all common signals will be shifted just one pin distance (1.27mm) away from each others. With this configuration the common signals can be connected with a through board via-hole and an additional trace of at most 2mm. Thus, very short stubs will be needed which will guarantee best possible signal quality.

The pin-configuration of the Dual transceiver is the same as Texas Instruments' SN65LVDS050/051devices [7]. However, the functionality of the enable signals is different.

The pin configuration of the 4x4 Cross point switch has not been defined yet.

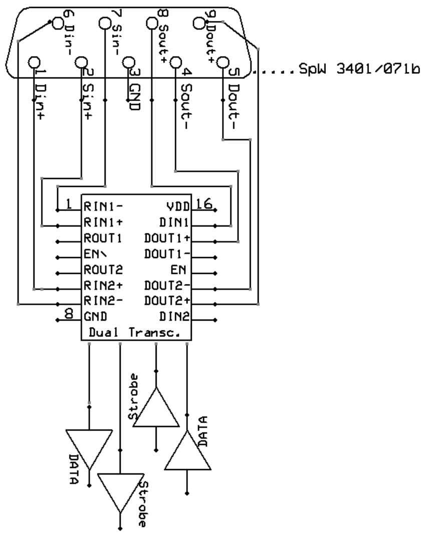


Fig. 3. Illustration of the signal matching of the Dual Transceiver (in middle) and the SpaceWire connector (on top). The SpaceWire RX signals, Data and strobe (Din/Sin), are on the left side and the TX signals, Data and Strobe (Dout/Sout), are on the right side. The converted single-ended SpaceWire signals can be traced on the PCB under the package in the direction away from the connector. Note that this illustration does not show the termination resistors tp the LVDS receivers.

B. Switching rates

Both devices support 400MBps switching rates. This puts high demands on low skew and jitter in all stages of the signal chain.

In Fig. 4. simulation results of the LVDS input stage is provided demonstrating a well-defined eye diagram over the full operational range.

The skew and jitter contribution from the LVDS output stage are comparable benign but the single-ended LVTTL outputs provides more challenges. With single-ended signals, any difference in rising and falling characteristics will consume on the available skew and jitter budget. In Fig. 5 simulation results of the single-ended output stage are provided demonstrating rising and falling signals crossing each other at 1.6V while the mid-point of input switch levels for LVTTL compatible inputs are 1.4V (V_{IH} = 2.0V – V_{IL}=0.8V) [3]. The actual switch point of any LVTTL input is not specified by the standard [3]. It may vary: between devices types, with temperature, voltage supply and input slope rate and between rising and falling edges. All these factors will define the overall skew of the interface and hence the achievable data rate. We will provide IBIS models in order to support PCB designers in optimizing their interface to our products.

In interfaces using more than one signal, like e.g. SpaceWire using a data and a strobe signal [2], the skew between the signals (channel-to-channel skew) will affect the maximum achievable data rate. The best channel-to-channel skew is achieved by putting all channels within the same device; in one device both the temperature and the voltage parameters of the eye diagram in Fig. 4. are identical and the contribution from process variations are minimized. We have simulated the channel-to-channel skew with Monte Carlo simulations within one device between two LVDS-to-single-ended channels, two single-ended-to-LVDS channels and two LVDS-to-LVDS channels: all pairs having a channel-to-channel skew below 250ps adding to the skew and jitter of the single channels alone.

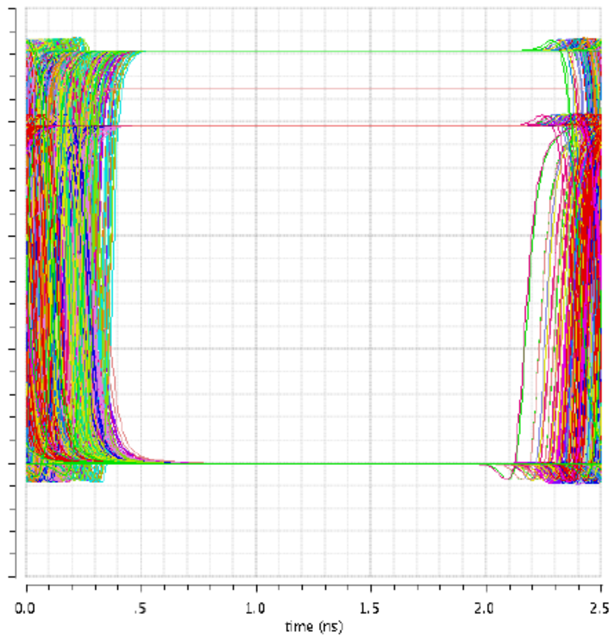


Fig. 4. Simulation result of LVDS input stage over all process, voltage and temperature corners. Input signal is a 100mV differential arbitrary 400 Mbps data signal.

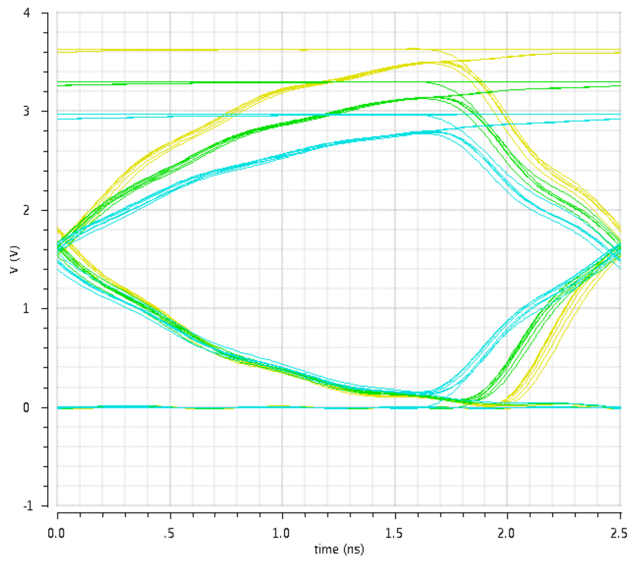


Fig. 5. Simulation result of the single-ended output with a 15pF capacitive load at slow process and temperature corner with 3.0V, 3.3V and 3.6V supply voltage. An arbitrary 400 Mbps data signal was applied to an internal node before the output stage.

C. ESD performance

ESD is one of the major threats to the overall reliability of an electronic system. Before assembly, ESD damages may induce latent defects to an electronic device, a damage that later after being mounted into an electronic equipment may escalate causing a catastrophic failure on system level. After assembly, most terminals of an electronic device are well protected to ESD damage but terminals directly connected the external interface of the equipment are still at danger. This is the reason that we have put extra design efforts to protect all LVDS signals against ESD. Both LVDS inputs and outputs are designed to withstand 8kV human body model (HBM) ESD. All other terminals are protected to withstand 3kV HBM ESD.

D. Over voltage protection

A major concern when designing high reliability systems are failure propagation. This could be a typical scenario: a DC/DC converter fails on an equipment (A) leading to an overvoltage condition in this equipment: the overvoltage propagates through the supply to the input or output signals of an interface device in this equipment; one of its input or output signal is connected to a interface device in another equipment (B); its input or output signal cannot withstand the overvoltage and propagates the overvoltage originating from equipment A further into equipment B.

One effective protection against this failure propagation scenario is to provide high voltage tolerance of all terminals on interfacing devices: a high voltage tolerance on the supply terminal of the interface device in equipment A can block the overvoltage propagating to its input and output signals. If not blocked instead a high voltage tolerance of the input and output terminals on the interface device in equipment B can block the overvoltage propagating to its supply. The absolute maximum voltage ratings of the products are provided in TABLE I. The voltage tolerance of our products exceeds the tolerance of most other LVDS devices thus by using our products the reliability with respect to failure propagation in a system will improve.

TABLE I. ABSOLUTE MAXIMUM RATINGS

Symbol	*Parameter*	*Min*	*Max*	*Unit*
V_{DD}	Voltage supply Range	-0.3	4.6	V
V_{IN}	Single-ended input voltage range	-0.3	6.0	V
V_{IN_LVDS}	LVDS input voltage range	-5.0	6.0	V
V_{OUT}	Single-ended output voltage range	-0.3	4.6	V
V_{OUT_LVDS}	LVDS output voltage range	-0.3	4.6	V

IV. RADIATION PERFORMANCE

Our products are manufactured in UMC's 180nm commercial CMOS technology. No changes to the process have been performed in order to increase the TID hardness, SEL immunity or SEU performance. Instead, the products are developed with imec's DARE (Design against Radiation Effects) library. Functionalities not already available in the library have been added using the DARE design methodology, using ELT and guarding. The DARE solution has previous heritage for both digital and analogue designs with proven good radiation performance [8,9]; pure digital circuits have been shown to be good to at least 1Mrad(Si) and analogue designs exceed 100kRad(Si). SEL has never been recorded in any design based on the DARE concept

Since the products contain no memory elements, SEU hardening is not applicable, but SET hardening is. SET hardening has been part of the design flow from smallest design-block level and upwards in the design hierarchy. SET pulses have been injected on each node in the design simulating a heavy ion hit while the effect on the output of the circuit has been assessed. Where needed hardening by means of SET filtering has been implemented.

Other LVDS devices have been radiation tested for Single event effects (SEE) showing numerous different effects on the differential output signal [10]: timing error of a transition, extended zero differential output voltage, bit state inversion (0-to-1 or 1-to-0), and transients on the common mode voltage level. These results have motivated us doing extensive SET simulations addressing these reported effects. In all simulations, SET pulses have been injected corresponding to a heavy ion injection with a LET of 60 MeV-cm^2/mg.

We have simulated SETs in the LVDS receiver stage. Both with a static input signal and dynamic switching input signals, we recorded events where the state of the receiver output signal toggles (bit state inversion). The critical nodes have been located, but SET hardening of the receiver stage is a trade-off between SET robustness and its speed performance. For these nodes we have favoured the speed performance. The worst case recorded bit state inversion was 3.2ns long induced with a SET injection corresponding to a heavy ion hit with a LET of 60 MeV-cm^2/mg. For lower LETs the duration time of the inversion decreases. At a LET below 7 MeV-cm^2/mg no events were recorded. We have measured the total area of the sensitive nodes in order to assess the total SET cross section and then estimated the expected SET rate in a geostationary orbit using the CREME96 tool [11]. The expected SET rate for bit inversions between 0.7ns and 3.2ns is below 2x10^{-5} events/day. One SpaceWire link requires four LVDS receivers, thus the SET rate of the link would be 8x10^{-5} events/day. In table II, the expected SET rate is transferred to bit error rate per different data rates assuming each SET will cause a bit error. This SET induced bit error rate is compared to the maximum overall bit error rate of 1x10^{-12} required for the SpaceWire standard [2] demonstrating an adequate margin for all data rates.

TABLE II. SET INDUCED BIT ERROR RATE (BER) PER SPACEWIRE INTERFACE

Data Rate		*SET induced BER*	
Per second	***Per day***	***errors per bit***	***Ratio versus a BER of 1x10^{-12}***
10 Mbps	8.64x10^{+11} bits/day	9.3x10^{-17}	9x10^{-5}
100 Mbps	8.64x10^{+12} bits/day	9.3 x10^{-18}	9x10^{-6}
200 Mbps	1.73x10^{+13} bits/day	4.6 x10^{-18}	5x10^{-6}
400 Mbps	3.46x10^{+13} bits/day	2.3 x10^{-18}	2x10^{-6}

In simulation of the LVDS driver stage, no event with bit inversion or zero differential voltage output has never been recorded. However, we have recorded disturbances on the differential voltage and the common mode output voltage. In Fig. 6. the worst recorded common mode SET is reported. From top: the 1st plot shows the common mode voltage, the 2nd plot shows the differential voltage and the 3rd and the 4th show the voltage on the two differential lines separately. The SET amplitude of the common mode voltage exceeds the LVDS standard [1] (1.125V to 1.375V) while the differential disturbances are well above the LVDS standard (>±100mV). Whether or not a LVDS receiver can reject a common mode disturbance like this depends on its characteristics. In theory, only common mode disturbances from the LVDS driver exceeding the input common mode range can be recorded erroneously. 0V to 2.4V is the standard [1] common mode range of a LVDS receiver. Thus, in theory the disturbance reported in Fig. 6. will be rejected. Our receiver with -4V to 5V common mode range provides even more margin. In Fig. 7 all SET events in the LVDS driver are simulated together with our LVDS receiver (no SETs injected in the receiver). From the top: the 1st plot shows the cumulated eye diagram of all SETs in the driver stage and the 2nd and 3rd plots show the rising and falling transition, respectively, of the receiver stage output signal. The results in Fig. 7 includes among other events the common mode event reported in Fig. 6. In conclusion, the final outcome of all SETs in the LVDS driver stage are disturbances that by the receiver stage are recorded as tiny timing errors on the transitions with at most 0.250ns. The simulation results can be compared with the SEE results of timing error of a transition reported by R.Koga [10] for heavy ions in the range of 0.9 MeV-cm2/mg to 90 MeV-cm^2/mg. For all tested LVDS devices, timing errors of 2ns and more was reported. The LET threshold was around 10 MeV-cm^2/mg with a saturation cross section around 1x10^{-4} cm^2/device. We have recorded timing errors one order of magnitude smaller at a significant higher LET. Thus, we are confident that the LVDS driver stage will perform outstanding compared to the LVDS devices tested by R.Koga [10].

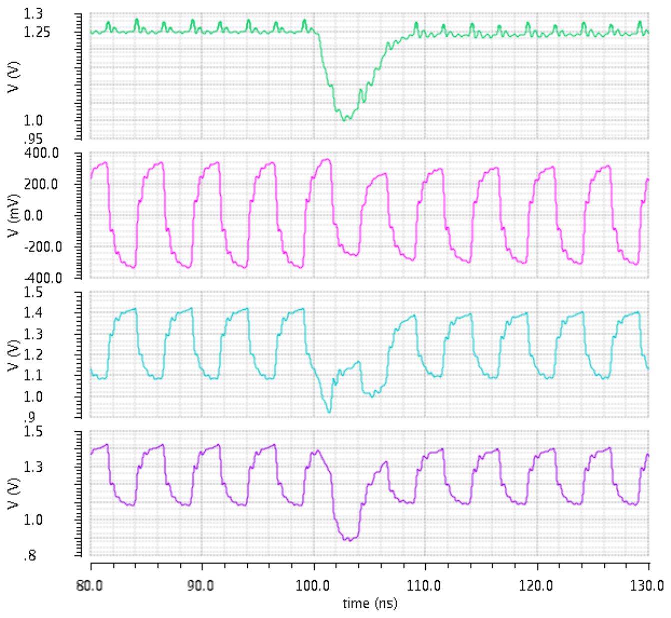


Fig. 6. Worst case recorded common mode disturbance in SET simulation on LVDS driver stage.

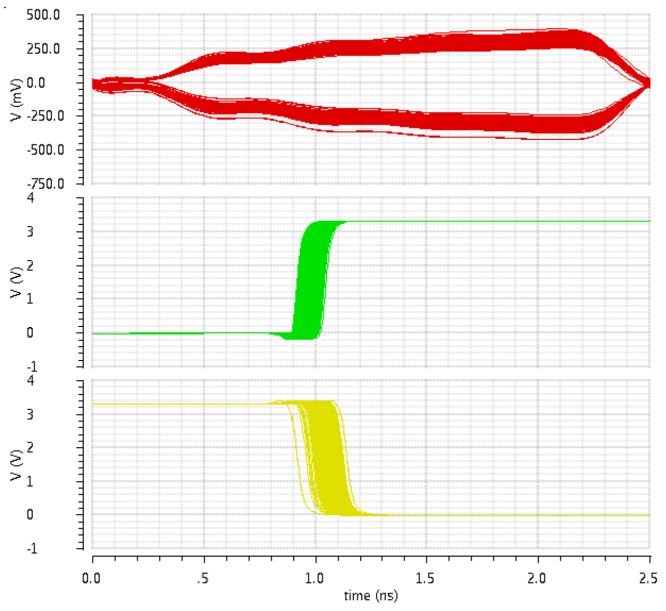


Fig. 7. Cumulated SET simulation result of LVDS driver stage with LVDS receiver stage connected to its output.

We also simulated the LVDS driver stage together with its internal supporting circuits like e.g. the current generator and the voltage band-gap circuit. Here we identified a node in the current generator circuit causing a timing error of almost one nanosecond (see Fig. 8.). We measured the area of this node. It is smaller than the area of the sensitive nodes in the receiver. We repeated the simulation with SET injections corresponding to a LET of 7 MeV-cm^2/mg. No events were recorded. Thus, the expected SEE rate is less than the one expected for one receiver (2x10^{-5} events/day geostationary orbit).

We have also simulated SETs in the single-ended input and output stages and all the internal signal paths with no critical SETs recorded.

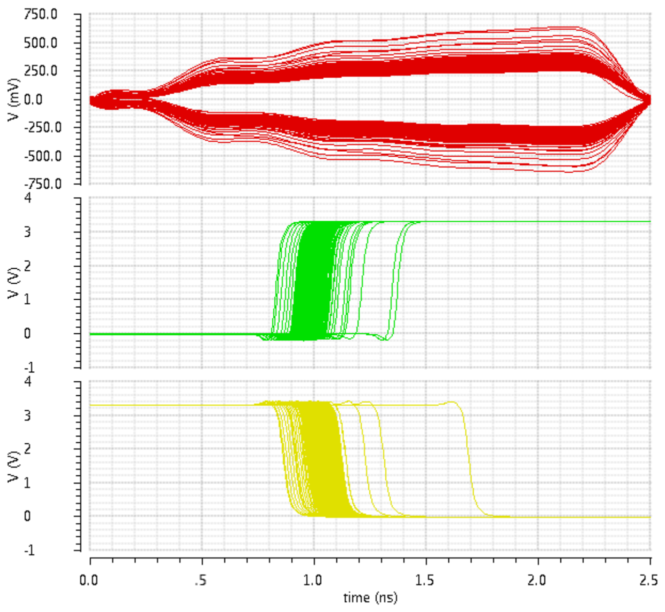


Fig. 8. Cumulated SET simulation result of LVDS driver stage and its supporting circuits with LVDS receiver stage connected to its output.

V. Conclusion

In this paper we presented two LVDS products being developed with the target to serve the need of SpaceWire communication in space applications. We have presented its key features and strengths.

This development has now completed the first design stage. We sent the first prototype to wafer manufacturing in March 2013. We expect them ready end of May. This prototype implements the functionality of the Dual transceiver as presented herein. Radiation tests for single event effects and total ionizing dose, ESD tests and electrical characterization will be performed on this prototype. Moreover, new potential features like power-down capabilities will be evaluated. In the next stage, the final designs of the Dual transceiver and the 4x4 cross point will be finalized and manufactured. The final products will be qualified to the ESCC standard for hermitically sealed monolithic circuits [12] with a targeted product release in 2014. This is a fully European development, thus no U.S export restrictions rules will apply to these products.

References

[1] Telecommunications Industry Association, "TIA/EIA Standard; Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits," ANSI/TIA/EIA-644-A-2001, January 2001.

[2] European Cooperation for Space Standardization, "Space Engineering; SpaceWire Links, nodes, routers and neworks," ECSS-E-ST-50-12C, July 2008.

[3] JEDEC Standard, *Interface Standard for Nominal 3 V/3.3 V Supply Digital Integrated Circuits*, JESD8-C. June 2006.

[4] DS90LV049 Datasheet, Texas Instruments

[5] Military standard, *Electronic Component Case Outlines*, MIL-STD-1835D, 1 June 2004.

[6] SN65LVDS125A Datasheet, Texas Instruments

[7] SN65LVDS050/051 Datasheet, Texas Instruments

[8] *R*edant, S.; Van Thielen, B.; Dupont, S.; Baguena, L.; Liegeon, E.; Marec, R.; Fernandez-Leon, A. and Glass, B. *HIT based flip-flops in the DARE library*. In: 14th Biennial Single Event Effects Symposium - SEE. 2004. (27-29 April 2004; Manhattan Beach, CA, USA.) (CD-ROM proceedings)

[9] Redant, S.; Marec, R.; Baguena, L.; Liegeon, E.; Van Thielen, B.; Beeckman, G.; Ribeiro, P.; Fernandez-Leon, A. and Glass, B. Radiation test results on first silicon in the DARE library, IEEE transactions on Nuclear Science, VOL. 52, NO. 5, October 2005

[10] R. KOGA, J. George, S. Bielat, and P Yu, "*Single Event Sensitivity of High-Speed Differential Signaling Devices to Heavy Ions and Protons," R*adiation Effects Datashop (REDW), 2011 IEEE.

[11] A. J Tylka et al, Chen, B. Mulgrew, and P. M. Grant, "*CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code,*" IEEE Trans. Nucl. Sci., vol. 44, pp. 2150–2160, Dec. 1997.

[12] European Space Component Coordination, "Integrated Circuits, Monolithic, Hermetically Sealed," ESCC Generic Specification No. 9000, Issue 6, September 2009.

.

Radiation-Tested Extended Common Mode LVDS Components

SpaceWire Components, Short Paper

Volodymyr Burkhay, André Rocke
TELEFUNKEN Semiconductors GmbH & Co. KG
Vahrenwalder Str. 247, D-30179 Hannover, Germany
volodymyr.burkhay@telefunkensemi.com,
andre.rocke@telefunkensemi.com

César Boatella Polo, Gianluca Furano, Farid Guettache,
Jørgen Ilstad, Giorgio Magistrati
European Space Agency
Keplerlaan 1, NL-2201 AZ Noordwijk ZH, The Netherlands
cesar.boatella.polo@esa.int, gianluca.furano@esa.int,
farid.guettache@esa.int, jorgen.ilstad@esa.int,
giorgio.magistrati@esa.int

***Abstract*—Extended Common Mode LVDS components from TELEFUNKEN Semiconductors are being tested for their radiation hardness. The collected test results are introduced and discussed; the results which are not yet available for the time being will be presented during the conference.**

***Index Terms*—LVDS, extended common mode, SpaceWire component, SOI, radiation test, TID, SEE.**

I. Introduction

LVDS translator IC components are widely used for SpaceWire (SpW) applications and are absolutely essential for aerospace equipment manufacturers. However none of the European IC manufacturers introduced such radiation hard components to the market until now. On the other hand extensive demand on radiation hard LVDS components suitable for extended common mode applications at high communication speed arose [1]. Those would help solving some currently existing robustness issues.

TELEFUNKEN Semiconductors is the first European IC supplier who developed components for extended common mode LVDS applications (see Fig. 1.), which are currently being tested for their radiation hardness by ESA.

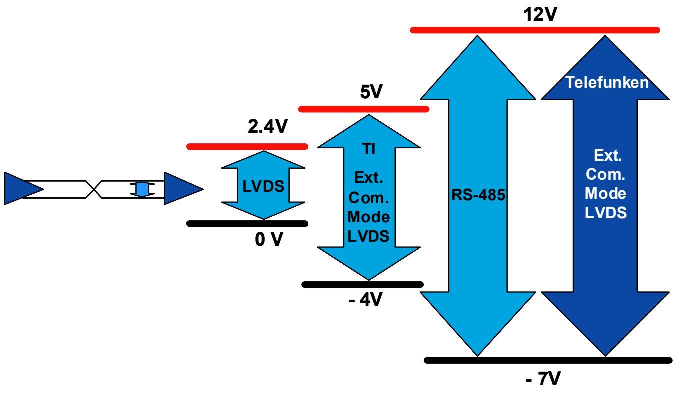


Fig. 1. Extended Common Mode LVDS

The radiation testing started with a high dose-rate Total Ionizing Dose (TID) test on unbiased components followed by Single Event Effects (SEE) tests and low dose-rate TID test on biased components.

II. Tested Components

The extended common mode capable LVDS components developed by TELEFUNKEN Semiconductors comprise LVDS receivers, drivers and splitters manufactured using Silicon-On-Insulator (SOI) technology TFSMART2.

Generally SOI technologies are known to mitigate SEE due to much smaller volume of charge collecting silicon compared to bulk devices [2]. If the SOI devices are fully isolated, as this is the case in TFSMART2, they are immune to latch-up thus no single event latch-up can occur. Additionally TFSMART2 features body ties for each device type, which due to charge diversion phenomena in SOI technology enhances the SEE immunity [2]. Combining bipolar and 3.3V CMOS logic devices having 0.35µm minimum feature size with high voltage DMOS devices up to 100V on the same die, this BCD IC manufacturing technology offers a high potential for aerospace applications [3]. Besides latch-up it is also inherently resistant to such parasitic effects as substrate leakage and others thanks to SOI, which improves the performance and makes it suitable for high temperature range.

The extended common mode capable LVDS components have been designed for the combination of the RS-485 receiver input voltage range and high-speed performance and efficiency of LVDS, providing robust but also fast communication channels. Those ICs translate the LVDS signals to 3.3V CMOS/TTL and vice versa with max provided data rate of 400Mbps and higher. The max data rate of such translators is limited by the CMOS I/O circuits, thus the best test vehicle for the data rate performance demonstration of LVDS circuits is a fully differential LVDS repeater. Such repeater comprising the same input as LVDS receiver and the same outputs as LVDS driver ICs shows significantly higher max data rate, which exceeds 800Mbps. The full DC common mode rejection range of the 1 to 4 repeater TF90LVDS104 at 1Gbps data rate is shown on the Fig. 2. as eye diagrams measured using PRBS23 input signal pattern with 200mV differential magnitude.

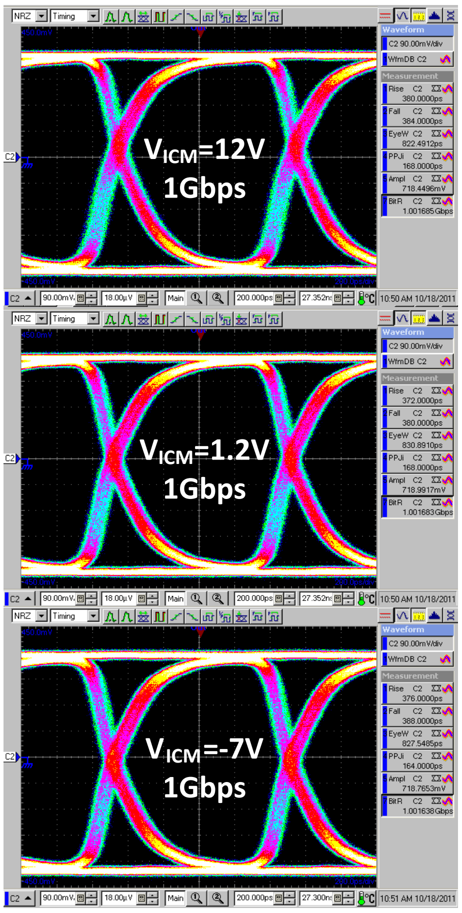


Fig. 2. Highest speed LVDS eye diagrams over common mode

From the extended common mode capable TELEFUNKEN Semiconductors LVDS components two component types have been selected for radiation testing: the LVDS receiver TF90LVD{S, T}032 [4] and the complementary LVDS driver TF90LVDS031 [5]. All described radiation tests are performed on these two component types.

III. Radiation Tests

A. Total Ionizing Dose Test

This test has been performed at the ESTEC ^{60}Co facility using a high dose-rate of 4.5krad/h [3].

The ICs of each of both component types have been divided into 6 groups: 5 irradiated groups and one control group; each irradiated group contained 5 ICs. The 5 groups of both components have been irradiated to the total dose of 5krad, 10krad, 20krad, 40krad and 100krad respectively and the parameter drifts have been measured. (There was a shipping period of 2 days between irradiation and post-radiation measurements.) Then the ICs annealed 7 days at room temperature and 5 hours at the temperature of 100°C, subsequent measurements followed.

The test results are shown in Fig. 3. The drifts of all examined parameters are shown relative to their pre-radiation values. The data points “5krad” to “100krad” are calculated from the mean values of the 5 different groups of ICs irradiated to the corresponding total dose. The data points “after room temperature anneal” and “after hot temperature anneal” belong to the group of ICs irradiated to 100krad total dose.

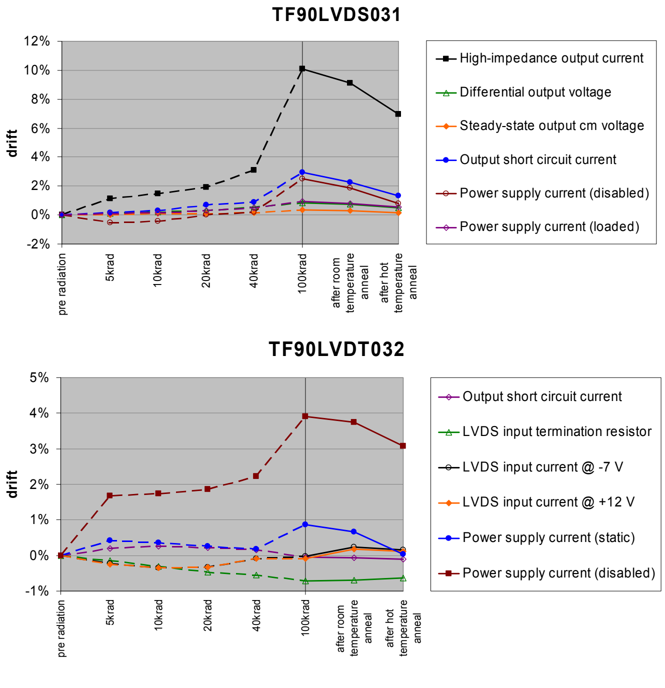


Fig. 3. High dose-rate unbiased TID test results

The shown test results are looking plausible, since the observable drift trend is constant through the total dose steps. The data points near 0% might be more influenced by measurement tolerances. The highest parameter drift is 10% whereas the majority of parameters doesn’t show measureable drifts. The TF90LVDS031 parameters “Differential output voltage” and “Steady-state output common mode voltage” show low drifts. They indicate that the voltage reference circuit was not significantly impacted by the radiation. The “Output short circuit current” of TF90LVDS031 shows that the drift of the current reference circuit might be approximately 3%. The TF90LVDS031 parameter “High-impedance output current” has wide tolerances. It shows 10% drift at 100krad total dose which might indicate some degree of degradation in gate oxide properties.

Finally, all tested parts keep their complete functionality after irradiation to the given TID radiation doses, room temperature annealing and accelerated ageing. No critical drifts or specification limit violations have been observed.

B. Single Event Effects Test

SEE heavy ions test and low dose-rate TID test on biased components are planned to be performed in week 19/2013; the results will be presented at the conference. The SEE test will be conduct in respect of ESA guideline: Single Event Effects Test Method and Guidelines ESCC Basic Specification No. 25100.

The purpose of single test for heavy ions test is to determine the sensitivity of Single Events Phenomena (SEL, SEU and SET for this application) against LET of incident ions and extract the cross section saturation and LET threshold for calculation and simulation of SEE in orbit.

The test will be performed on two or three different pairs (driver-receiver pair) of component samples with the case lid removed, the two samples in a pair will be irradiated separately and the not irradiated sample will be a part of the test equipment for the DUT. Every component will be tested for SEL/SEU/SET. The DUT will be a part of SpW communication channel and the behavior will be observed using Link Analyzer and Digital Signal Oscilloscope (see Fig. 4.). The test equipment used in this configuration is able to capture failures causing data corruption and display accurately the behavior of the SpW link during these events. The digital signal oscilloscope captures accurately SET behavior of the devices, being both common and differential mode distortions to the LVDS signal, as well as transients on the CMOS logic outputs of the LVDS receiver.

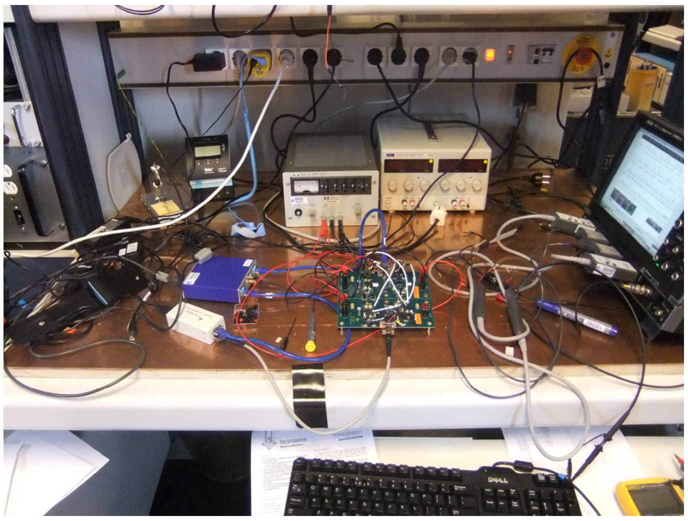


Fig. 4. DUT under functional test in ESTEC Avionics Lab

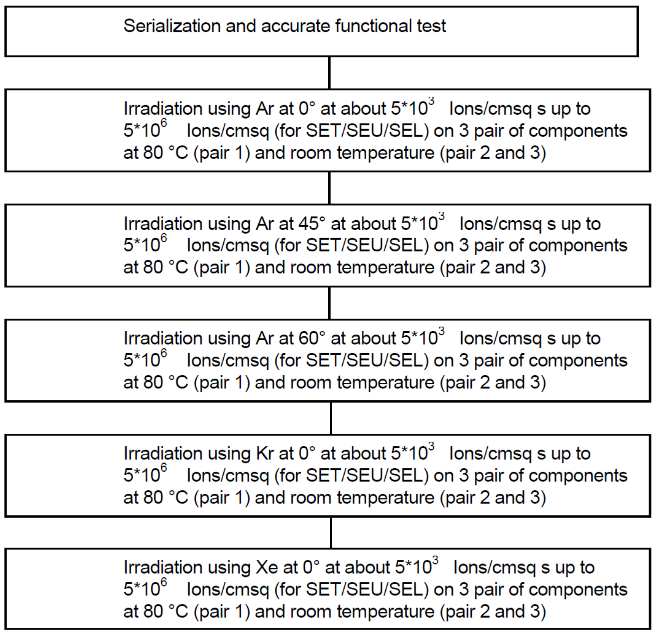
The components will tentatively be tested with the ions shown in the following table:

Ion	Kinetic Energy/nucleon (MeV/n)	Angle (deg)	Air (cm)	Range (μm)	LET (MeV/mg/cm^2)
Ar-40	20	0	20	132.2	9.19
Ar-40	20	45	20	93.5	12.99
Ar-40	20	60	20	66.1	18.37
Kr-84	20	0	12	68.0	33.14
Xe-129	20	0	15	50.83	60.68

The components will be irradiated at the flux of about 5*10^3 ions/cm^2/s up to a total fluence of 5*10^6 ions/cm^2 or 200 SEE events for each irradiation run. The test flow is shown in the Fig. 5.

The components will be tested at high temperature +70°C (first pair) and at room temperature for the second and the third pair. The SEE test campaign will be performed with the support of MAPRAD srl (Perugia, Italy) at the LNS lab of Catania (Italy).

SEE tests performed on a Point-of-Load converter IC manufactured in the same technology TFSMART2 showed no fails [6], which suggests good results also for current SEE test. During the Conference all available radiation test results will be presented.



Serialization and accurate functional test

Irradiation using Ar at 0° at about 5*10^3 Ions/cmsq s up to 5*10^6 Ions/cmsq (for SET/SEU/SEL) on 3 pair of components at 80 °C (pair 1) and room temperature (pair 2 and 3)

Irradiation using Ar at 45° at about 5*10^3 Ions/cmsq s up to 5*10^6 Ions/cmsq (for SET/SEU/SEL) on 3 pair of components at 80 °C (pair 1) and room temperature (pair 2 and 3)

Irradiation using Ar at 60° at about 5*10^3 Ions/cmsq s up to 5*10^6 Ions/cmsq (for SET/SEU/SEL) on 3 pair of components at 80 °C (pair 1) and room temperature (pair 2 and 3)

Irradiation using Kr at 0° at about 5*10^3 Ions/cmsq s up to 5*10^6 Ions/cmsq (for SET/SEU/SEL) on 3 pair of components at 80 °C (pair 1) and room temperature (pair 2 and 3)

Irradiation using Xe at 0° at about 5*10^3 Ions/cmsq s up to 5*10^6 Ions/cmsq (for SET/SEU/SEL) on 3 pair of components at 80 °C (pair 1) and room temperature (pair 2 and 3)

Fig. 5. Radiation Test Flow

IV. Conclusion

After performing a high dose-rate TID test up to 100krad on unbiased extended common mode LVDS components from TELEFUNKEN Semiconductors promising results were obtained. After the TID exposure none of the component specifications were violated and all tested parts kept their complete functionality.

SEE heavy ions test and low dose-rate TID test on biased components will follow soon. During the conference all available radiation test results will be presented.

We are looking forward to obtain good results from the described radiation tests. Afterwards the high-quality European components will be made available on the space market and further components for extended common mode LVDS application will be developed.

References

[1] J. Ilstad, “ESA’s Requirements for future LVDS devices”, LVDS Application Workshop, Noordwijk, June 2011

[2] A. Samaras, “JUPITER Mission and Strong Environment” Radiation Specification for System conception, JUICE Mission, CNES, July 2012

[3] V. Burkhay, G. Ilicali, A. Rocke, “Radiation Test of TFSMART2 Technology using Extended Common Mode LVDS and DC-DC Converter Components”, 4th International Workshop on Analogue and Mixed Signal Integrated Circuits for Space Applications, Noordwijk, August 2012

[4] TELEFUNKEN Semiconductors, "TF90LVDS032 / TF90LVDT032 Quad LVDS Line Receivers with Extended Common Mode" Datasheet, April 2012, http://www.tfproducts.com/dwh/ds/if/ecmlvds/tf90lvds032.pdf

[5] TELEFUNKEN Semiconductors, "TF90LVDS031 Quad LVDS Line Driver" Datasheet, October 2011, http://www.tfproducts.com/dwh/ds/if/ecmlvds/tf90lvds031.pdf

[6] G. Magistrati, "Laser Beam Test and SEE Test Report of Telefunken TF6002" , ESA unclassified - for official use, February 2013

Components 2 (Long)

New DSP based IP, Devices and Systems for Space Applications featuring SpW / SpFi Interfaces

SpaceWire Components – Long Paper

R. Trautner
TEC-EDP, ESA/ESTEC
Keplerlaan 1, 2200AG Noordwijk,
The Netherlands
Roland.Trautner@esa.int

Abstract—Digital Signal Processors (DSPs) are important components for many types of space systems such as instruments, payload data processors, and platform subsystems. As the only existing European DSP (TSC21020) is becoming obsolete, ESA is pursuing the development of new DSP devices, related IP, and system designs for space applications. In this paper, we provide an overview of DSP related developments supported by ESA, with relevant information on the integration of SpW and SpFi interfaces and related system aspects such as bandwidth constraints and utilization of specific features. The baseline architectures of future DSP ASICs are presented, and the ESA roadmap for Digital Signal Processing is summarized.

Index Terms— Digital Signal Processing, SpaceWire, SpaceFibre, SpW, SpFi.

I. Introduction

Digital Signal Processors are based on architectures that make them particularly suitable for applications such as payload data processing, real time control loops, and similar applications where digital data needs to be processed at high speed and low power consumption. The expected obsolescence of the only European DSP, but also its outdated design and performance, create a problematic gap in terms of power efficient processing capability for many application areas. The development of a new DSP has been delayed by the lack of available funding. As a result, equipment developers are often forced to use backup solutions involving FPGAs or dedicated Application Specific Integrated Circuits (ASICs). However, these come with associated disadvantages such as high power consumption, lower reliability, long development time, or high cost. At an ESA–industry round table held in 2007 [1], a number of DSP technology development routes have been defined. These have been addressed via a number of different R&D contracts, leading to the development of technologies and architectures that can bridge the gap until a new, high performance, radiation hardened DSP component is available. The development routes include processor boards based on Commercial Off The Shelf (COTS) DSPs, development of radiation hardened DSP IP cores for future ASIC developments including prototype chip developments, and preparatory activities supporting the future development of a European Next Generation Space DSP (NGDSP). These activities are presented in the following chapters.

II. COTS based DSP boards

The use of commercial components for space applications is an option that may allow to achieve higher performance, lower power, and smaller footprint and volume than would be possible by using only space qualified components. In general, the use of commercial components does not lead to lower cost; in most cases the additional cost of qualification more than compensates for the lower cost of component procurement. However, the current lack of a space qualified high performance DSP component has led to significant efforts in the development of processor boards based on commercial DSPs. These boards need to provide protection mechanisms for mitigation of radiation induced processing errors. Among the DSPs available on commercial markets, the TMS320C6727 available from TI® has been found to be particularly suitable due to its good performance and availability as a QML-V component. It is latchup immune and sufficiently tolerant to Total Ionising Dose (TID). However, the component is sensitive to Single Event Effects (SEE) and must therefore be protected by suitable radiation mitigation techniques in order to achieve an availability that is sufficient for space applications.

A. Hi-P COTS based Computer

This activity is part of a broader development effort that includes the development of highly reliable (Hi-R), highly available (Hi-V) and high performance (Hi-P) COTS based computers. The Hi-P development (" High Performance COTS Based Computer Step 2" , ESTEC contract nr. 4000105087) is based on the aforementioned TI® DSP, with radiation mitigation techniques implemented in a combination of hardware and software. A key element of this activity, which is performed by Astrium (F) in collaboration with CGS (I), is an architecture that combines a high reliability control element ("SmartIO") with a scalable number of DSP based processing modules (PMs), supporting the tailoring of reliability, application performance and latency according to the user and application requirements. Depending on the number of available PMs, task duplication in time (each processing task repeated at least once) or duplication / triplication in space (task executed on multiple PMs in parallel) with subsequent voting and possible re-calculation can be chosen.

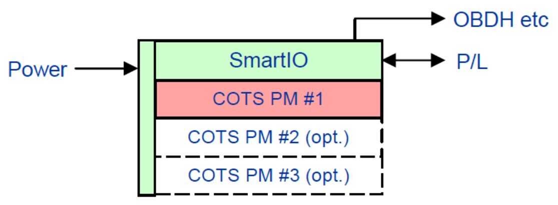


Fig. 1. Hi-P CBC modular architecture

The Hi-P system architecture is shown in Fig. 1, while a candidate architecture for the processing module is depicted in Fig. 2. For very high bandwidth, input data (for example originating from a payload such as a radar) can be routed to the processing modules via a separate switch matrix, which allows to avoid a bandwidth bottleneck in the SmartIo which is typically based on standard General Purpose Processor (GPP).

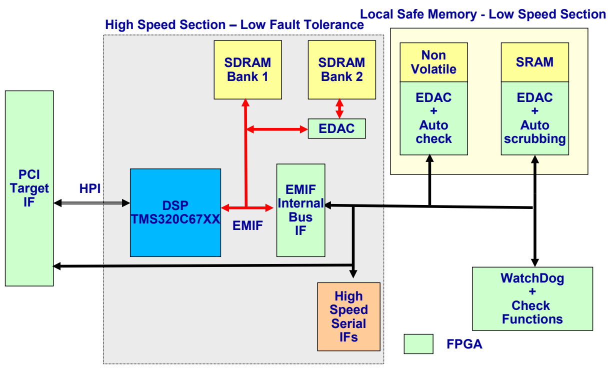


Fig. 2. Hi-P PM candidate architecture

Sensitive elements of the PM architecture, including all space standard interfaces, are implemented in rad-hard FPGA or ASIC technology, and support fast data verification via checksum calculation as well as monitoring of the COTS DSP.

The PMs are expected to provide 1 or 2 SpW interfaces (with RMAP target functionality) as well as 1 or 2 SpFi links based on TI's TLK2711 serializer / de-serializer circuits. The final number of implemented interfaces will depend on available FPGA resources. While the speed of the FPGA allows adequate performance for the SpW interfaces, the SpFi links may run at lower than typical speed which will be optimized during the PM detailed design phase.

For a flight application the FPGA may be replaced by a dedicated ASIC, allowing both higher speed and lower power consumption in addition to higher reliability. In order to assess the system performance, a number of performance benchmarks [2] will be implemented. The results will allow direct performance comparisons with other platforms. It is expected that the activity will be concluded in Q1 2014. The target Technology Readiness Level (TRL) is 5, and TRL 6 for critical technologies.

B. HPPDSP

This activity is a second development based on the same COTS DSP, but with a different technical baseline, and with a specific set of requirements derived from studies of future science missions. Low mass and very low power consumption are among the driving design requirements. The architecture is based on a dual DSP concept with FPGA based monitoring of task execution and data consistency . The development is performed by Astrium UK ("High Processing Performance Digital Signal Processor", ESA Contract 1-6182, 2009) supported by University of Dundee (UK). It includes the development of a TRL 4 processor breadboard as well as implementation of demonstration software including ESA's NGDSP benchmarks [2]. Also here, both SpW and SpFi interfaces are integrated. Additional information is available in [3]. The end of the activity is expected in the 2nd half of 2013.

III. Rad-Hard Fixed Point DSP and NoC Elements

While the development of a space qualified floating-point DSP has been hampered by funding constraints, some significant work has been performed on fixed point DSP IP cores and related Network-on-Chip (NoC) technology. This included FPGA breadboarding, design radiation hardening and prototype chip development. In addition, fine-grained massively parallel architectures are also being investigated.

A. Massively Parallel Processor Breadboard

This development activity ("Massively Parallel Processor Breadboarding Study", ESA contract nr. 21986, 2008-2012) was performed by RECORE Systems b.v. (NL). It has succeeded in the development of a NoC based system that combines two fixed point VLIW Xentium™ DSP cores with a LEON2 controller [4] . The FPGA based design includes features such as SpW including RMAP protocol support, CCSDS timers, ADC / DAC interfaces, and on-chip as well as off-chip memories. The basic architecture of the developed system is depicted in Fig. 3.

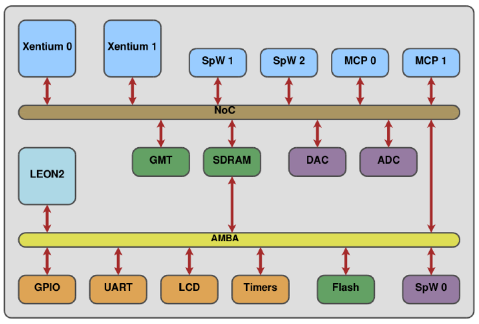


Fig. 3. MPPB architecture

All elements requiring high bandwidth connectivity are connected to the 32-bit wide NoC. Due to the chosen number of network lanes and routers, multiple high data rate transfers can be handled concurrently without congestion. The MPPB NoC architecture is shown in Fig. 4.

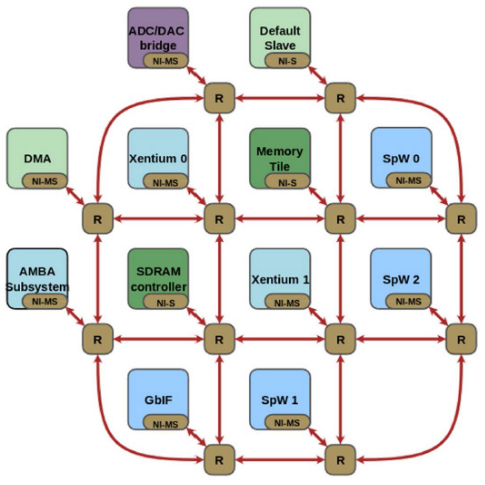


Fig. 4. MPPB NoC architecture

The Xentium™ DSP cores support up to 4 MACs per clock cycle for 16 bit data, and up to 2 MACs per clock cycle for 32 bit data words.

In the FPGA implementation which provides a system clock of 50 MHz the speed of SpW interfaces is limited to 100 Mbps. In an ASIC implementation the full SpW speed can be achieved. It should be noted that the NoC architecture, which provides 32bit bi-directional connections between routers and operates at full system clock, provides very high bandwidth for connected data sources and sinks. This architecture is therefore a good candidate for future designs aiming at high bandwidth applications. The MPPB study has been completed in 2012, and final presentation materials are available via [5].

B. DARE+ Application ASIC

The successful MPPB activity has opened the door to the development of multi-core high performance processor ASICs based on NoC and VLIW fixed point DSPs. However, a key step towards this goal is the radiation hardening of the DSP IP and NoC elements such as routers, network bridges, DMA, memory tiles, and relevant interface IP.

The DARE technology [6], a rad-hard ASIC library based on a commercial (non-space qualified) 180nm ASIC manufacturing process (UMC) has been developed by IMEC (BE) under ESA contract since 1999. Following the basic library development and its evaluation by means of test vehicles, a subsequent activity called DARE+ ("DARE plus – ASICs for Extreme Radiation Hardness and Harsh Environments", contract Nr. 4000104087) was started for fixing identified issues and for the development of additional library elements. Part of this activity is the design, manufacturing and test of an application ASIC (called XentiumDARE, or XD). It includes key parts of the MPPB IP (DSP core, NoC routers, NoC bridge, SpW RMAP–NoC interface, on-chip memory tile, and others). All architectural elements have been radiation hardened either via ASIC library elements or via architectural changes such as triplication and insertion of EDACs. Programming and debugging is possible via either SpW RMAP or UART. Due to chip size constraints the memory tile also serves as DSP instruction cache which would be kept separate in a flight ASIC implementation. The Application ASIC architecture is depicted in Fig. 5.

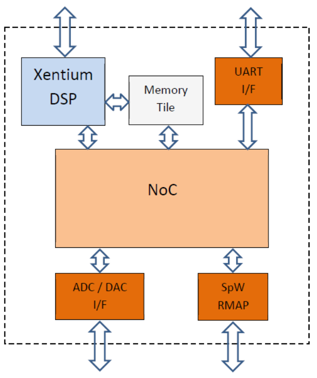


Fig. 5. DARE+ Application ASIC architecture

The SpW I/F with RMAP target functionality is directly mapped to the NoC, which provides an internal bandwidth of 3.2 Gbps (bi-directional) at a a target system clock of 100MHz. Data transfers are supported via DMA. Due to the lack of High Speed Serial Link (HSSL) IP in DARE, SpW is expected to remain the main standard high speed interface for future DARE180 based ASIC developments.

C. High Performance Data Processor

This activity ("High Performance Data Processor", ESA contract nr. 4000102909) is performed by ISD (Greece) and Astrium (Germany) [7]. It is based on a proposal from industry, and aims at the development of a processor prototype that is based on scalable reconfigurable fixed point processing array technology from PACT (Germany). Fig. 4 shows the basic architecture of the envisaged prototype chip.

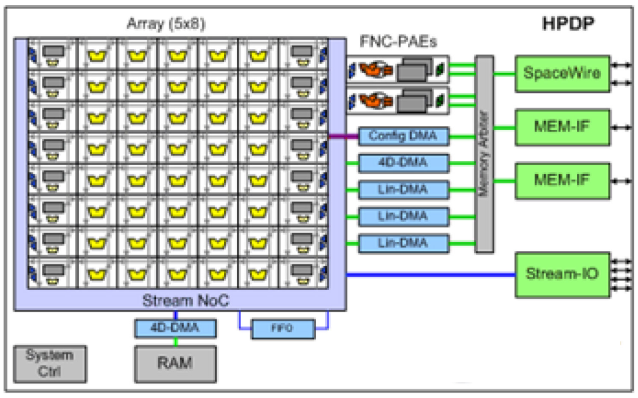


Fig. 6. HPDP prototype chip architecture

In this architecture, a large reconfigurable processing core capable of handling high bandwidth data streams is supported by a number of additional processing elements including DMAs, memory and stream I/O interfaces. The chip design is aiming mainly at telecom applications. A set of test applications (various DVB-S processing steps such as encoders/decoders, PSK modulators/demodulators, simple FFT and filter routines) will be demonstrated. SpaceWire interfaces are used for the control links to a platform's data handling system, while payload data streams are routed via separate interfaces with a total bandwidth comparable to that of a SpFi link. The prototype ASIC development, which is based on a commercial 65nm technology [8], has suffered some delays; results from this activity are now expected around 2015.

IV. Near Term DSP ASIC Projects

In addition to the activities described in the last paragraphs, the preparations for the development of new, performant DSP ASICs have continued. Based on funding from different sources such as the Core Technology Program (CTP) of ESA's Science Directorate and the European Component Initiative (ECI4) development steps for both fixed point and floating point DSP ASICs will be implemented in the near future.

A. Scalable Sensor Data Processor

This activity, which is expected to start in 2013 and deliver prototype ASICs by 2015, will integrate the results of MPPB, DARE+ application ASIC, MPPB assessment results, and other work into an ASIC development that aims at a commercial product that may be available around the 2016 timeframe. The baseline architecture is very similar to MPPB and includes a LEON2 and two Xentium™ DSPs connected via a NoC. The target for system clock is 100 MHz. The SSDP design exploits the mixed signal capabilities of the DARE technology by integrating both fast and slow ADCs as well as multiplexers (MUX) and sensor signal conditioning circuits.

Additional IPs such as Pulse Width Modulation (PWM) units and standard peripherals are integrated in order to create a versatile chip which is highly suitable for applications such as payload data processing, instrument control, and platform subsystems that include sensor data processing functions.

A functional diagram is shown in Fig. 7. In the upper half it shows the NoC subsystem with associated DSPs, fast interfaces, and bridges to external components. The lower half consists essentially of a standard LEON2 system with AHB/APB buses and typical peripherals. A particular feature is the housekeeping (HK) data acquisition ADC and MUX connected to the AHB bus which is managed by the LEON. The final design will be based on consolidated user requirements and architectural tradeoffs.

It is expected that RMAP enabled SpW links will be the key digital interfaces of this ASIC, providing typical speeds of 200 Mbps. A parallel interface to external ADC/DAC components that also supports data streaming among multiple SSDP chips is also foreseen. It will provide data rates close to ~1 Gbit/sec. Additional interfaces will include standard peripherals such as UARTs and parallel I/Os.

The SSDP development may be accompanied by additional developments for software and processor boards depending on available funding and the needs of the early user community.

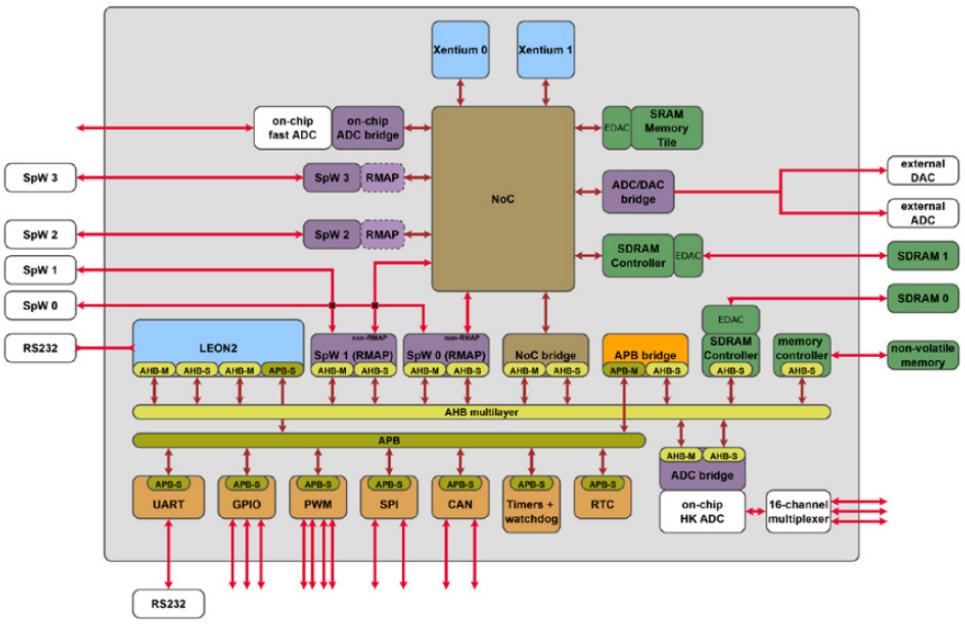


Fig. 7. SSDP DARE+ Application ASIC functional diagram

B. *Next Generation Floating Point DSP*

The development of a radiation hard high performance Floating Point DSP that can replace the outdated TSC21020 and that allows more reliable and power efficient solutions than the COTS DSPs described in II. remains the goal of the main DSP development line.

In a previous TRP activity ("European DSP Tradeoff and Definition Study", ESA contract nr. 420002645, 2008-2012) performed by Astrium (F/UK/GE) with support from ISD (GR), a number of candidate commercial DSP IPs were evaluated. The initial assessment included availability as an IP, performance of the DSP core, and capabilities and user friendliness of the Software Development Environment (SDE).

In a following step, a primary candidate and a backup solution were investigated in detail, including an assessment of the migration to available (ATMEL 180nm) and near future (STM 65nm) rad-hard ASIC technologies. The assessment also addressed required modifications of the architectures for radiation hardening, addition of space specific features, and compatibility with space qualified peripheral components.

Three commercial DSPs were evaluated:

- ATMEL DIOPSIS 940HF
- Analog Devices ADSP-21469
- Texas Instruments TMS320C6727B

For the TMS320C6727B, the manufacturer was not inclined to license the design for an ASIC development. However, as the component is already available as a latchup insensitive component (and used in several COTS based computer developments) the performance evaluation and SDE assessment was continued. For the other candidates IP licensing is possible and the detailed assessment was performed. The ADI DSP was found to be the superior device in terms of both performance and SDE quality. The ATMEL device was kept as a backup.

The analysis of achievable performances revealed that only the STM 65nm process would allow to achieve the performance goal of at least 1 GFLOP. It also provides the added benefit of HSSL IP for the implementation of SpFi. The migration to the rad-hard ASIC library as well as architectural changes required for radiation hardening (which includes adding EDACs for internal and external memories) will reduce the maximum clock frequency of the rad-hard DSP below that achieved by commercial devices. A clock frequency in the range of 200 MHz is expected based on the initial analysis.

In addition to the integration of EDACs and other means for radiation hardening, some significant design changes will be introduced to adapt the architecture to the needs of space applications. The most important additions will be SpW and SpFi interfaces and an interface to external suitably qualified memories (DDR2 or DDR3 in addition to SRAM and ROM). On the other hand, some IPs that are part of the commercial DSP design such as audio processor and some accelerators that are of no or limited use for space applications may be removed.

In an initial assumption for the NGDSP ASIC architecture shown in Fig. 5, 4 SpW links are assumed which allows connections to redundant data sources and data sinks in a typical spacecraft architecture. The supported data rate is 200 Mbps or higher. For the implementation of SpFi links the DSP's internal 32 bit peripheral bus will impose limitations on achievable bandwidth. Depending on the core clock frequency, the peripheral bus, which is running at ½ of the core clock, will provide bandwidth not exceeding 3.2 – 4 Gbps. It is therefore possible that the number of SpFi IPs will be limited to 2, with 4 external interfaces implemented in order to provide redundant links as for the SpW interfaces.

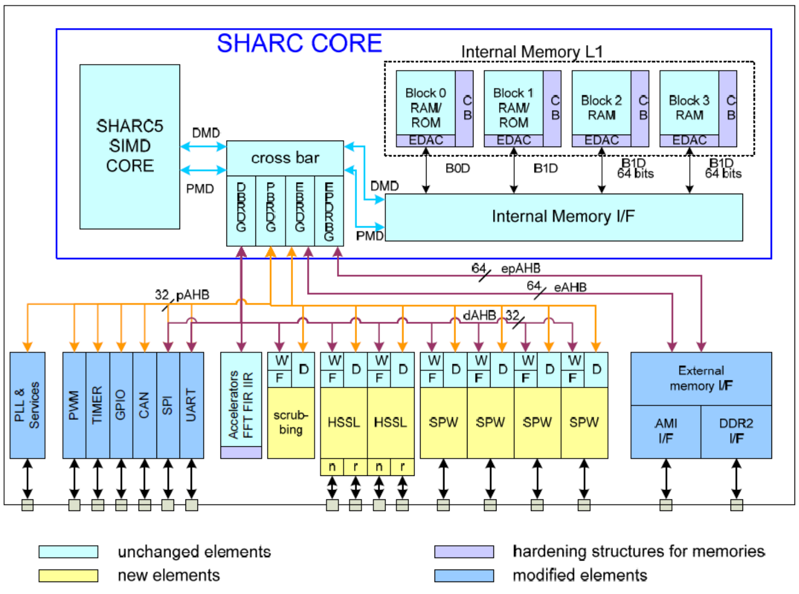


Fig. 8. Draft NGDSP ASIC architecture

All architectural changes need to take into account possible implications on the SDE, as one key development objective is to keep modifications transparent for the SDE wherever possible. The next step in the development of the NGDSP ASIC will be a feasibility study which is expected to include synthesis tests using the DSM 65nm rad hard library and corresponding toolchain. Other tasks, like partial FPGA prototyping including IP integration tests, SDE tests etc. may be included as well. This activity is expected to start not later than 2nd half of 2013.

V. ESA Roadmap for Digital Signal Processing

All the activities introduced in previous paragraphs are part of the ESA roadmap for Digital Signal Processing. This roadmap, which is based on the identified needs of European industry and other stakeholders, but also includes activities proposed by industry and enabled by direct national funding, is periodically adjusted in order to reflect the realities of technology evolution, availability of funding, identified synergies, and progress of ongoing activities. Fig. 9 shows an overview of the presented activities and their interdependences. Additional activities may soon appear in support of the presented core activities after the down selection of ESA TRP proposals for the 2014/2015 timeframe. The complete roadmap, which includes additional activities not presented here due to status, limited relevance for this paper, or paper volume restrictions, is available and frequently updated on ESA's On-board Data Processing Website [5].

VI. Summary

European industry and other stakeholders have an urgent need for technologies that support and enable reliable digital signal processing for space applications at high performance and power efficiency. As existing components are outdated and will soon become obsolete, powerful new technologies based on COTS components as well as rad-hard ASICs are being developed. Radiation hard fixed point DSP IP for ASIC developments is now commercially available; COTS based DSP boards are expected to be ready for adoption by projects in 2014, while new DSP ASICs are expected to be available as prototypes in 2015 and 2016 and as flight models in 2016 and 2017 for SSDP and NGDSP, respectively. Evaluation boards and associated software are expected to become available at the same time as ASIC prototypes.

References

[1] R. Trautner, " Next Generation Processor for On-board Payload Data Processing Application – ESA Round Table Synthesis" , TEC-EDP/2007.35/RT, 2007. The document is available via [5].

[2] R. Trautner, "Next Generation Space Digital Signal Processor Software Benchmark" , TEC-EDP/2008.18/RT, 2008. The document is available via [5].

[3] Bruce Yu, Chris McClements, Pete Scott, David Dillon, Steve Parkes, "High Processing Power Digital Signal Processor with SpaceWire and SpaceFibre Interfaces", 2013 SpaceWire Conference, Goteborg, 2013.

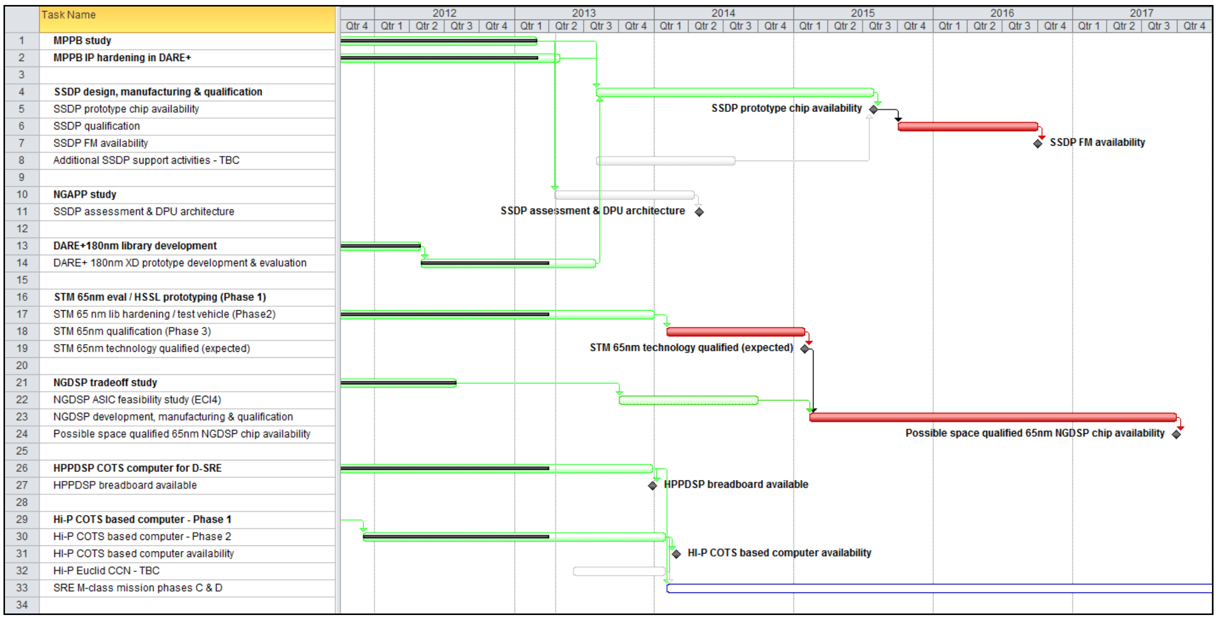
[4] K. Walters et al., Multicore SoC for On-board Payload Signal Processing, Adaptive Hardware and Systems Conference, San Diego, USA, 2011.

[5] ESA On-board Data Processing Website, http://www.esa.int/TEC/OBDP/

[6] S. Redant, et. al., "The Design Against Radiation Effects (DARE) Library", 5th RADECS Workshop. Madrid, Spain, 2004.

[7] M. Syed, High Performance Data Processor, Adaptive Hardware and Systems Conference, Noordwijk, The Netherlands, 2008.

[8] C. Papadas, Prototyping the HPDP Chip on STM 65nm Process, DASIA, Malta, 2011.



	Task Name
1	**MPPB study**
2	**MPPB IP hardening in DARE+**
3	
4	**SSDP design, manufacturing & qualification**
5	SSDP prototype chip availability
6	SSDP qualification
7	SSDP FM availability
8	Additional SSDP support activities - TBC
9	
10	**NGAPP study**
11	SSDP assessment & DPU architecture
12	
13	**DARE+180nm library development**
14	DARE+ 180nm XD prototype development & evaluation
15	
16	**STM 65nm eval / HSSL prototyping (Phase 1)**
17	STM 65 nm lib hardening / test vehicle (Phase2)
18	STM 65nm qualification (Phase 3)
19	STM 65nm technology qualified (expected)
20	
21	**NGDSP tradeoff study**
22	NGDSP ASIC feasibility study (ECI4)
23	NGDSP development, manufacturing & qualification
24	Possible space qualified 65nm NGDSP chip availability
25	
26	**HPPDSP COTS computer for D-SRE**
27	HPPDSP breadboard available
28	
29	**Hi-P COTS based computer - Phase 1**
30	Hi-P COTS based computer - Phase 2
31	Hi-P COTS based computer availability
32	Hi-P Euclid CCN - TBC
33	SRE M-class mission phases C & D
34	

2012
2013
2014
2015
2016
2017
Qtr 4
Qtr 1
Qtr 2
Qtr 3
Qtr 4
SSDP prototype chip availability
SSDP FM availability
SSDP assessment & DPU architecture
STM 65nm technology qualified (expected)
Possible space qualified 65nm NGDSP chip availability
HPPDSP breadboard available
HI-P COTS based computer availability

Fig. 9. Presented Activities: Schedule and Interdependence

A Radiation Tolerant SpaceFibre Interface Device

SpaceWire Component, Long Paper

Steve Parkes[1], Albert Ferrer[2], Alberto Gonzalez[2], Chris McClements[2], Ran Ginosar[3], Tuvia Liran[3], Dov Alon[3], Michael Goldberg[3], Gal Sokolov[4], Gennady Burdo[4], Nimrod Blatt[4], Paul Rastetter[5], Milos Krstic[6], Alberto Crescenzio[7]

[1]Space Technology Centre, 166 Nethergate, University of Dundee, Dundee, DD1 4EE, Scotland, UK
[2]STAR-Dundee, STAR House, 166 Nethergate, Dundee, DD1 4EE, Scotland, UK
[3]Ramon-Chips Ltd., 104 Galil Street, Nofit 36001, Israel
[4]ACE-IC, Caesarea, Israel
[5]EADS Astrium GmbH, Munich, Germany
[6]IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany
[7]Synergie CAD Instruments S.R.L., Italy

Email: sparkes@computing.dundee.ac.uk

***Abstract*— The Very High-Speed Serial Interface (VHiSSI) device aims to provide a versatile SpaceFibre interface device in a small package. The device can act as a parallel interface device providing several modes of operation, or it can act as a SpaceWire to SpaceFibre bridge.**

This paper describes the VHiSSI chip in detail, outlines the applications it can be used for, and summarises the status of the VHiSSi project.

Index Terms*—*SpaceWire, SpaceFibre, networks, spacecraft onboard processing

I. INTRODUCTION

Space-based Earth observation and scientific instrumentation currently under development will push the limits of on-board data-handling technology. In the past Mil-Std 1553 and proprietary data-links were used to get instrument data from the instruments to the on-board mass memory unit and to the down-link telemetry system. Over the past decade the proprietary data links have been replaced with a standard networking technology designed for use on-board spacecraft: SpaceWire. While SpaceWire is currently being used to fulfil the on-board data-handling requirements of many missions, there are some very high data-rate instruments which are beyond its capabilities.

Several future space-based instruments, for example synthetic aperture radar (SAR) and hyper-spectral imagers, will be capable of producing data at data rates of several Gbits/s. New downlink telemetry techniques (laser and Ka-band communications) will be able to provide much higher downlink capacity than previously possible. High speed memory technologies will be able to serve multiple high data rate instruments and stream data to ground on demand. To support the growing need for onboard communications network bandwidth, technologies able to support multi-Gbits/s data transfer have been developed, e.g. Channel Link and, Wizard Link. Unfortunately these are all restricted USA devices resulting in a critical European dependency.

ESA has been developing a standard multi-Gbits/s network technology called SpaceFibre [1] [2] [3] [4]. SpaceFibre provides multi-Gbits/s data rates over fibre-optic and electrical cable. It provides a coherent quality of service mechanism able to support bandwidth reserved, scheduled and priority based qualities of service. It provides extensive fault detection, isolation and recovery (FDIR) capabilities, including a link level retry function that recovers from errors and resends data transparent to the user application. SpaceFibre uses the same packet format as SpaceWire [5] making it easy to bridge existing SpaceWire devices into a SpaceFibre network.

The VHiSSI project is a European Union Framework 7 research project which will integrate a complete SpaceFibre protocol engine, together with the physical layer interfaces, in a radiation tolerant chip manufactured by a European foundry. It will provide a complete SpaceFibre solution in a single chip.

The VHiSSI research programme aims to create very high-speed data-interface technology which is a critical component technology for future spacecraft payloads, particularly telecommunications and Earth observation payloads where multi-Gbits/s data-rates are urgently needed. A complete solution to very high-speed data networking onboard spacecraft will be provided, levering research on SpaceFibre, using a European fabrication facility, and providing a non-dependent technology.

The VHiSSI research programme will:

- Provide multi-Gbit/s serial data-link technology, essential for future spacecraft onboard data-handling systems.

- Lever prior and concurrent research on the emerging SpaceFibre standard, to provide a complete multi-Gbit/s serial technology for spacecraft onboard data-links and networks, including fault detection, isolation and recovery (FDIR) and quality of service (QoS).
- Provide a versatile chip architecture, which can be adapted and configured to support multiple applications.
- Provide the critical clock-recovery mechanism on existing European chip technology.
- Use a European semiconductor fabrication facility, enhancing and developing its capabilities for radiation tolerant chip design and production with a radiation tolerant library.
- Provide a non-dependent technology (ITAR free), allowing unrestricted use on European spacecraft and substantial export opportunities - an important capability for Europe.

This paper describes the work carried out by University of Dundee and STAR-Dundee on this project and summarises the current state of the project. The team working on this project comprises engineers from:

- University of Dundee who are leading the project, and who are responsible for project management and the VHiSSI device architecture and specification.
- Astrium GmbH who are responsible for gathering and requirements and use cases for the VHiSSI device.
- STAR-Dundee Ltd who are responsible for designing the digital part of the VHiSSI chip in register-transfer level (RTL) VHDL code.
- ACE-IC who are responsible for the design of the SerDes and CML transceivers.
- Ramon Chips who are responsible for the radiation tolerant library for the IHP chip manufacturing process and low level design of the VHiSSI device.
- IHP who are responsible for manufacturing the VHiSSI device and digital/static test of the chip.
- SCI who are responsible for supporting the high performance/analogue testing of the experimental VHiSSI chip.

II. VHiSSI Chip Architecture

The overall architectural block diagram of the VHiSSI chip is illustrated in Figure 1.

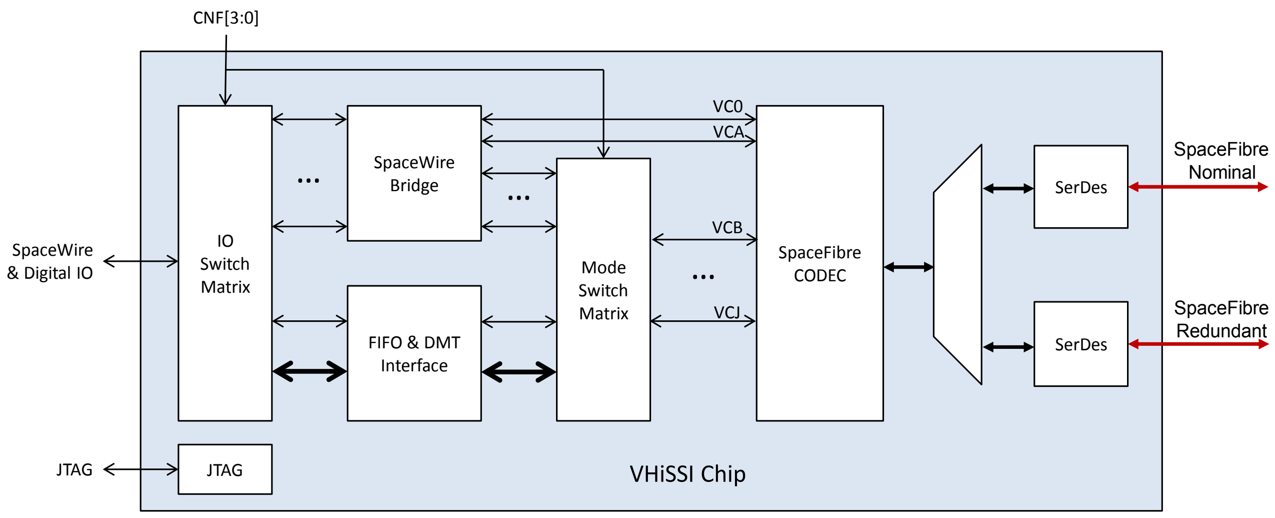


Figure 1 VHiSSI Overall Architecture

There are five main functions within the VHiSSI chip:

- SpaceWire Bridge
- FIFO, DMA, Memory and Transaction Interface
- SpaceFibre Interface
- SerDes
- IO Switch Matrix
- Mode Switch Matrix

The SpaceWire Bridge provides a bridge between SpaceWire and SpaceFibre with up to 11 SpaceWire interfaces being available. The SpaceWire Bridge includes a SpaceWire router which allows routing between SpaceWire ports and Virtual Channel (VC) buffers of the two SpaceFibre interfaces. Configuration of the VHiSSI chip can be carried out over any SpaceWire interface connected to the embedded SpaceWire router or over VC0 or VCA of the SpaceFibre interface. The

SpaceWire Bridge is connected to the IO Switch Matrix and to the Mode Switch Matrix.

The FIFO and DMA, Memory and Transaction (DMT) Interface provides various types of parallel interface into the VHiSSI chip for sending and receiving data over the SpaceFibre interfaces. The various parallel interface functions have been designed with specific application scenarios in mind and between them are able to operate with many types of local host system, including FPGAs and processors. The parallel interface is also designed to use a small number of pins, so that the VHiSSI chip can fit into a small (100 pin) package. The FIFO mode provides a direct parallel interface to two SpaceFibre virtual channels. The memory type interface provides a 32-bit bus interface for accessing VHiSSI registers or VC buffers. It is a multiplexed address/data bus, with the VHiSSI device providing an internal address latch/counter to hold the register/VC buffer address. The transaction interface is similar to the memory interface, but aims to simplify software interfacing. A single address line is used to distinguish commands and status information from data. A command is written to the VHiSSI device to specify the transaction that is about to take place. For data transfer to/from a VC buffer, a read of status information provides the status of the VC buffer identified in the command. The data transfer can then take place in a burst transfer the maximum size of which is determined by the VC buffer status information. The DMA interface puts the VHiSSI chip in control of data transfers. When there is data ready to transfer, an internal DMA controller in the VHiSSI device requests control of the external data bus. Once granted it then affects the data transfer. An external address latch/counter is required, which may be implemented in an FPGA. The FIFO and DMT interface is connected to the IO Switch Matrix and to the Mode Switch Matrix. On reset the IO pins and connections to the VC buffers from the FIFO and DMT interface and SpaceWire Bridge are determined and set by these two switch matrices.

The SpaceFibre Interface has 11 virtual channels. VC 0 is intended primarily for VHiSSI device and local system configuration and monitoring and is connected to the embedded SpaceWire router. The other VCs have programmable VC numbers and so are referred to by letters. VCA is connected to the embedded SpaceWire router. The other VCs are either connected to the SpaceWire router, directly to a SpaceWire interface, or to the parallel interface, depending on the mode of operation. Each VC supports full SpaceFibre QoS which can be configured independently for each VC. VC0 and VCA are directly connected to the embedded SpaceWire router. The other SpaceFibre VC buffers are connected to the Mode Switch Matrix which connects them to either the SpaceWire Bridge or the parallel interface. The other side of the SpaceFibre interface is connected via a multiplexer to either the nominal or redundant SerDes and CML transceiver.

The SerDes converts parallel data words from the SpaceFibre interface into a serial bit stream and vice versa. On the receive side the bit clock is recovered from the serial bit stream by the SerDes. The SerDes includes integral CML transceivers.

The IO Switch Matrix connects either the SpaceWire LVDS, SpaceWire LVTTL or parallel interface signals from the FIFO and DMT interface to the digital IO pins of the VHiSSI chip. Configuration is static and determined on exit from device reset, i.e. on the rising edge of the RSTN signal.

The Mode Switch Matrix connects either the SpaceWire Bridge or FIFO and DMT interface (parallel interface) to the VC buffers of the two SpaceFibre interfaces. Configuration is static and determined on exit from device reset, i.e. on the rising edge of the RSTN signal.

In addition to these major functions the VHiSSI chip includes a JTAG test port and some other device test modes.

III. VHiSSI Chip Applications

In this section several applications of the VHiSSI device are considered

A. High Data-Rate Instrument Interface

SpaceFibre offers substantially higher data rates than SpaceWire to support high data-rate instruments. Connection of a high data-rate instrument to a mass memory unit via SpaceFibre is illustrated in Figure 2.

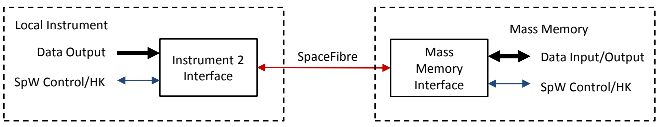


Figure 2 High Data-Rate Instrument Connected To Mass Memory

To provide data at high-speed from a local instrument to the SpaceFibre interface a parallel interface is required. To operate with current space qualified FPGAs this interface has to be 32 bits wide, which requires a 62.5 MHz interface clock (32-bits x 62.5 MHz = 2 Gbits/s, which after 8B/10B encoding is 2.5 Gbits/s signalling rate).

The simplest type of interface is a FIFO type interface, which is straightforward to connect to an FPGA. For high data rate transfer from an instrument it is only necessary to write data to an output VC buffer in the SpaceFibre interface. A slower speed interface, e.g. SpaceWire, would be useful for controlling and reading housekeeping information from the instrument.

If the instrument includes an embedded processor it may be preferable to use a memory type interface to write and read data from the SpaceFibre VC buffers in the SpaceFibre interface. This interface can then also be used to access the configuration, control and status registers inside the SpaceFibre interface. In this case it is the responsibility of the instrument to handle the transfer of data to the SpaceFibre interface.

A DMA controller included in the SpaceFibre interface transfers responsibility for data transfer from the instrument controller to the SpaceFibre interface. This may save some important processing power within the instrument controller.

The VHiSSI device is able to provide a SpaceFibre interface for high data rate instruments using a FIFO, memory or DMA type interface to an FPGA or processor. This interface is designed to be able to operate a clock speeds achievable by flight qualified FPGAs while sustaining 2 Gbits/s data transfers. It also is designed to minimise the number of pins required for the interface.

B. SpaceWire to SpaceFibre Bridge

SpaceWire has been used extensively to provide a standard interface to various instruments. To connect these instruments into a SpaceFibre based data-handling network a SpaceWire to SpaceFibre Bridge is required.

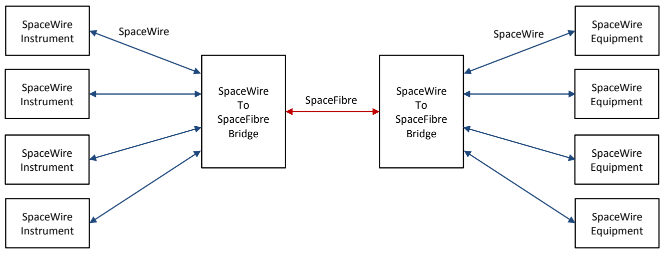


Figure 3 SpaceWire to SpaceFibre Bridge

Figure 3 shows a SpaceWire to SpaceFibre Bridge being used to multiplex several SpaceWire links over a single SpaceFibre link. In this particular example four instruments with SpaceWire interfaces are connected to some other SpaceWire enabled equipment. Bridging between SpaceWire and SpaceFibre is straightforward since both protocols use the same packet format.

The VHiSSI chip can operate as a SpaceWire to SpaceFibre bridge with either LVDS or LVTTL SpaceWire interfaces and includes an internal SpaceWire router.

C. Mass Memory Interface

A mass memory requires several SpaceFibre interface connections to support several high data-rate instruments and instruments with SpaceWire interfaces. This is illustrated in Figure 4.

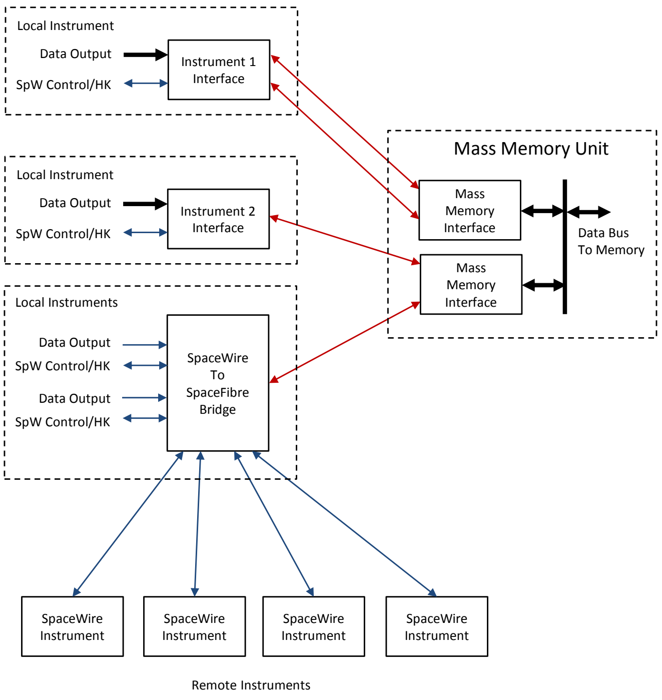


Figure 4 Mass Memory Interface

Two high data-rate instruments are shown, one with a single SpaceFibre link and the other requiring two SpaceFibre links to support data rates of 4 Gbits/s. Several SpaceWire instruments are also connected to the mass memory via a SpaceWire to SpaceFibre Bridge.

The Mass Memory unit provides four SpaceFibre interfaces connected to a common bus or network for accessing the memory modules that are to store the data.

The VHiSSI chip can provide all the SpaceFibre interfaces required in the example network of Figure 3: high-speed instrument interfaces, SpaceWire to SpaceFibre bridge and the interface to the mass memory unit.

D. Control Processor

Configuration and control information can be sent over a SpaceFibre network using individual virtual channels or a virtual network. A SpaceFibre router allows a control processor to access all the instruments and other equipment on the network as illustrated in Figure 5.

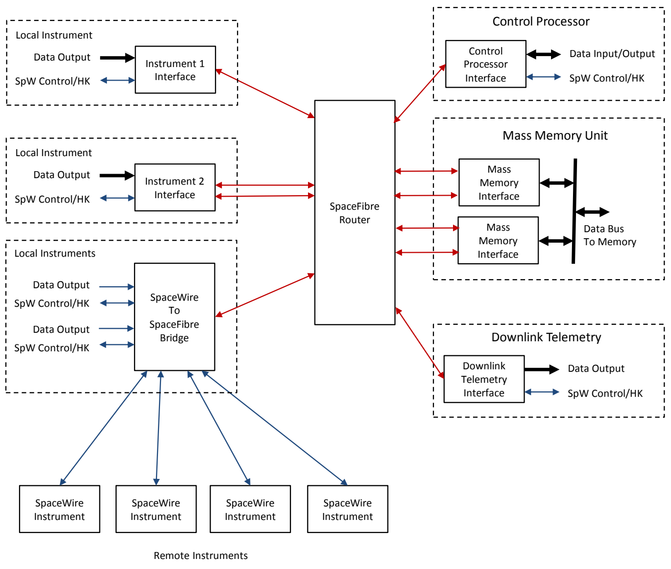


Figure 5 Control Processor on SpaceFibre Network

Figure 5 shows a complete SpaceFibre based on-board data-handling system. A SpaceFibre router is used to interconnect the various units. A control processor is connected to this router. It is able to send configuration, control and status request commands to all of the other units on the network. Typically a virtual network would be used to manage this control and status information, where one virtual channel in each unit is dedicated to control/status and each of them is given the same virtual channel number, e.g. VC0. The control processor then sends SpaceWire packets containing commands over VC0 to another unit. This unit responds over VC0. Since the control processor is the master of the VC0 virtual network, there is no undesirable contention between SpaceWire packets on VC0. This approach leaves all the other virtual channels available for data transfer.

The SpaceWire instruments do not support virtual channels, so control/status packets and data packets have to be multiplexed over the SpaceWire links. The SpaceWire to SpaceFibre Bridge must be able to support this multiplexing of SpaceWire packets containing control information, status or instrument data. This requires a SpaceWire router which could be provided within the SpaceWire to SpaceFibre Bridge. Normally configuration, control and housekeeping requests require small packets and should therefore not have a major impact on data transfer over the single SpaceWire link from instrument to the SpaceWire router in the SpaceWire to SpaceFibre Bridge.

The VHiSSI device together with a SpaceFibre router device can provide all the SpaceFibre network functionality needed for onboard data-handling architectures like that of Figure 5.

IV. Status of VHiSSI Project

A comprehensive set of requirements for the experimental VHiSSI chip have been gathered from the European spacecraft engineering community by Astrium GmbH, focusing on a small device which could be used to provide very high-speed data-links on-board a spacecraft. A versatile chip interface has been designed by University of Dundee which covers many potential applications while keeping the number of pins required on the chip to a minimum. The architectural level design of the experimental VHiSSI chip and its interface definition have been shaped, reviewed and polished and detailed design of this chip is currently underway by STAR-Dundee Ltd.

A critical part of the VHiSSI project is the radiation tolerant serialiser/deserialiser, clock-data recovery circuitry and high-speed serial driver/receiver technology. This is a demanding design activity due to the speed of the interface and the required radiation tolerance. A design has been created by ACE-IC ready for testing.

The use of the IHP chip foundry required a complete radiation tolerant component library to be designed. This has been carried out by Ramon Chips and includes logic gates, IO, LVDS IO, and memory cells. A test chip called RADIC5 has been designed and implemented which includes the critical circuitry designed by ACE-IC and library test components from Ramon Chips. This test chip is currently under test by Synergie-CAD and IHP. The results of this testing will feed into updated component design by ACE-IC and Ramon Chips which will be incorporated into the experimental VHiSSI chip.

The layout of the RADIC5 test chip is shown in Figure 6.

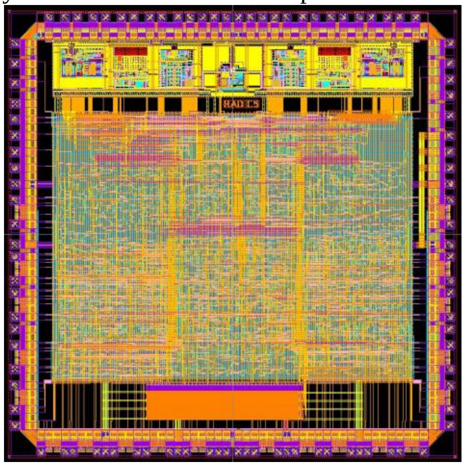


Figure 6 RADIC5 Test Chip

A radiation test board for the RADIC5 is currently being designed and will be used to support the radiation testing of the SerDes and other components on the chip.

An FPGA board is also being designed to support the functional validation and system level validation of the VHiSSI chip design prior to manufacture of the VHiSSI ASIC device. This FPGA board will also be used to provide test signals for functional testing of the VHiSSI ASIC device once it has been manufactured.

The next steps are to complete testing of the test chip, to finalise the design of the experimental VHiSSI chip, to manufacture this chip, and to test it.

V. Conclusions

SpaceFibre is a powerful, multi-Gbits/s networking technology for use on board spacecraft which has QoS and FDIR capabilities built into the hardware. The VHiSSI project is an EU Framework 7 project that is designing an experimental SpaceFibre chip. The VHiSSI chip is implemented in a small 100 pin package but provides complete SpaceFibre interface and SpaceWire to SpaceFibre bridge functionality. This chip is designed to cover the various SpaceFibre network interface requirements envisaged for different onboard systems, including SpaceWire bridging, high data-rate instrument interfacing, and mass memory unit interfacing. An initial test chip (RADIC5) has been produced to test the critical radiation tolerant SerDes technology and the radiation tolerant library components. The RADIC5 chip is currently under test. The experimental VHiSSI chip will be ready for testing during 2014.

Acknowledgment

The research leading to these results has received funding the European Space Agency under ESA contract numbers 4000102641 and from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 284389. We would also like to thank Martin Suess the ESA project manager for the SpaceFibre related activities.

References

[1] S. Parkes, A. Ferrer, A. Gonzalez, & C. McClements, “SpaceFibre Standard Draft E1”, University of Dundee, 28th September 2012.

[2] S. Parkes, A. Ferrer, A. Gonzalez, & C. McClements, “SpaceFibre: Multiple Gbits/s Network Technology with QoS, FDIR and SpaceWire Packet Transfer Capabilities”, International SpaceWire Conference, Gothenburg, June 2013.

[3] S. Parkes, “Never Mind the Quality, Feel the Bandwidth: Quality of Service Drivers for Future Onboard Communication Networks”, paper no. IAC-10.B2.6.6, 61st International Astronautical Congress, Prague 2010.

[4] S. Parkes, C. McClements and M. Suess, “SpaceFibre”, International SpaceWire Conference, St Petersburg, Russia, 2010, ISBN 978-0-9557196-2-2, pp 41-45.

[5] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire, Links, Nodes, Routers and Networks”, Issue 1, European Cooperation for Space Data Standardization, July 2008.

Onboard Equipment & Software (Short)

Prototype of Onboard Mass Storage Device Based on SpaceWire and SpaceFibre Interfaces

SpaceWire Onboard Equipment and Software, Short Paper

Viacheslav Grishin, Petr Eremeev, Sergey Gorbunov
JSC “NII “Submicron”
Zelenograd, Moscow, Russian Federation
grishin@se.zgrad.ru, epm@se.zgrad.ru, aser3000@ya.ru

Yury Sheynin, Elena Suvorova
St.Petersburg University of Aerospace Instrumentation
St.Petersburg, Russian Federation
sheynin@aanet.ru

Tatiana Solokhina, Alexander Glushkov, Ilya Alekseev, Leonid Menshenin, Jaroslav Petrichkovich
“ELVEES” RnD Center
Zelenograd, Moscow, Russian Federation
tanya@elvees.com

Bernard Berne
3D Plus
Buc, France
bberne@3d-plus.com

***Abstract*—In this article the mass storage device prototype for the onboard computing system of a new generation space platform is considered. Its application, structure, information interfaces and etc. are presented. Use of the high-speed SpaceFibre and SpaceWire interfaces is offered.**

***Index Terms*—Mass Storage Device, SpaceWire, SpaceFibre, NAND-Flash, Memory Controller.**

I. Introduction

Onboard control systems of existing spacecrafts, as a rule, are built on the base of the separate systems controlled by the digital computer, and are integrated in a single network by means of the system level interface (usually it is MIL-STD-1553). JSC «NII «Submicron» is to develop a perspective onboard informational computing system (OICS) based on the SpaceWire and SpaceFibre network technologies [1], [2] for the new generation space satellites.

The onboard mass storage device (OMSD) is one of the elements of developed OICS. It is intended for non-volatile storage of digital information from Earth observation equipment.

Existing highly reliable specialized storage devices of leading world manufacturers are based on the NAND-flash memory (e.g. “TCS”, USA; “Galleon Embedded”, Norway) [3], [4]. This trend is justified primarily by the absence of rotating mechanical parts, as in the classic hard disk drives. This causes their increased reliability and service life.

Typically, the volume of target information and its recording (or reading) rate to a drive have special requirements to communication interface bandwidth. Thus baud rate can be up to several Gb / s per channel and transmission line length of up to 10 meters and more.

Typically, onboard specialized storage devices are made for multi-channel connectivity of multiple sources of information. It is caused by limitations of both the dimensions and weight parameters and energy consumption.

The most common used in solid state drives interface is SATA 2 (3Gb/s), SATA 3 (6Gb/s), PCI-Express (4 Gb/s), Fiber Channel (up to 10 Gbit/s) [5].

The most appropriate is the idea of application a specialized gigabit unified digital interface for use in the storage devices for space applications. The aim is to provide unification of cross-platform compatibility of equipment used in various projects in different countries (including in the framework of international cooperation), as well as reducing development time and debugging the finished product.

To solve these problems the most promising interfaces are SpaceWire and SpaceFibre. They are specially designed for use in the on-board equipment of spacecraft. These interfaces allow to work in a wide speed range from several Mbit/s up to several Gbit/s and cover a wide range of solved problems on board [6].

II. Description

The structure of the OMSD and the OICS as a whole suggests applying of SpaceWire and SpaceFibre high-speed interfaces as a communication transmission medium of instructions and data. The SpaceFibre interface will be used for communication of those elements of OICS where data transmission rates reach several Gb/s per channel. The SpaceWire interface is used as the common unified environment for transmission of commands and interaction between all subsystems of OICS.

Structurally the OMSD consists of storage modules (from 2 to 15 modules) and two switch modules (Fig. 1).

Storage modules are intended for reception of input information on two channels of the high speed SpaceFibre interface and it’s saving in NAND-Flash memory. Switch modules are intended for information transfer between storage

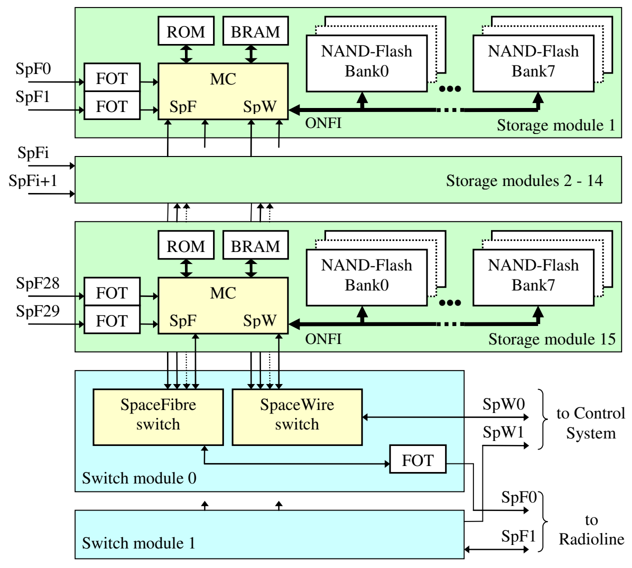


Fig. 1. The structure of the OMSD.

modules and for formation of an output flow to Earth via a high speed radiofrequency line.

Basic elements of the storage modules are the memory controller (MC) and NAND-Flash memory. Radiation-tolerant chip MCT-03D is implemented as a "system on chip" based on CPU IP-core with MIPS32 architecture, developed by "ELVEES" RnD center.

Preliminary MCT-03D microprocessor main characteristics:

- Central Processor Unit (CPU):
  - Architecture – MIPS32 compatible;
  - 16 KB data and 16 KB instruction cache with direct mapped capability;
  - Memory management unit (MMU): with TLB and Fixed Mapped;
  - Multiplier and Divider;
  - FPU ANSI/IEEE Standard 754-1985, “IEEE Standard for Binary Floating-Point Arithmetic.”, Single and Double precision;
  - JTAG IEEE 1149.1 On Chip Debug Unit;
  - 128 KB RAM;
  - five external interrupt, NMI.
- External Memory Port (MPORT):
  - Data bus – 32 bit, Address bus – 24 bit;
  - Built-in controller SRAM, NOR Flash, NAND Flash, SDRAM;
  - Program configuration type and size blocks of memory;
  - separate data buses for both system memory and NAND-Flash controllers;
  - Program configuration wait cycle of SRAM;
  - supporting up to 16 banks of NAND-Flash memory;
  - parallel simultaneous writing to four banks (four channels) of NAND-Flash memory;
  - technical speed of one NAND-Flash Memory Interface channel no less than 33 MB/s;.
- Peripheral units:
  - Two duplex SpaceWire (ECSS-E-50-12C) ports*, from 2 up to 400 Mbod each with built-in the chip LVDS transceivers;
  - Four duplex GigaSpaceWire (SpaceFibre or SpaceWire-RT for future upgrades) ports*, from 5 Mbod up to 1.25-2,5 Gbod each with built-in the chip CML - compatible transceivers;
  - Two Multifunctional Buffed Serial Ports (MFBSP): SPI, I2S, LPORT, GPIO support;
  - Two 4-channels DMA. Flyby mode data transfer (ADSP-TS201);
- Interrupt controller;
  - Two UART (16550);
  - Two 32-bit interval timers;
  - 32-bit Watch Dog timer.
- Additional features:
  - “Radiation Tolerant” 120MHz ASIC for space applications (CMOS, “RadHard by design” process);
  - several built-in the chip PLL;
  - Internal and external memory Error correction: single error correction and double error detection by the Hamming code;
  - Power saving Modes;
  - Development and debugging tools: MCStudio-3M;
  - C, C + + compiler;
  - OS LINUX 2.6.36 and RTOS uOS support.
- Package: CQFP-240;
- ASIC status: experimental IC will be implemented by ELVEES in 2013-2014.

Test samples (prototype) of the MCT-03D chip (Fig. 2) without built-in silicon 4-channels gigabit router (MCT-03P with one gigabit port) are designed for 0.18 µm design rules and manufactured at the Russian factory [7].

NAND-Flash memory is based on modules commercially available from 3D PLUS [8]. The OMSD consisting at 15 storage modules has total capacity up to 1 Tbytes.

III. Main Challenges

The main tasks in the development of storage device are managing the distribution of stored information, ensuring its integrity during transmission and storage.

The problem of data distribution occurs due to the limited memory amount of a single memory module and multi-channel structure of the storage device as a whole, with all its channels operating independently of each other. This problem can be solved by sharing of switching modules and a common file system of storage device. The task of switching modules will include the management of information flows redistribution among storage modules in accordance with the file table (i.e. issuing commands to memory controllers). The task of the memory controller is sending/receiving the information and data exchange with NAND-flash modules (i.e. the transformation of logical addresses in the files table to corresponding physical memory addresses in a specific storage module).

* ELVEES & SUAI IP - cores design

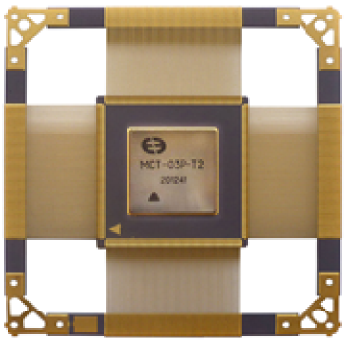


Fig. 2. MCT-03P with one gigabit port – the test sample of the MCT-03D chip without built-in silicon 4-channels gigabit router.

Ensuring the integrity of stored information during transmission (or reading) will be made by using the mechanism of jamfree coding. This may be a Reed-Solomon code, which allows to identify and correct multiple errors in blocks of information. This justified by the fact that the NAND-flash memory cell are subject to wear and damage during operation. This could affect the reliability of a previously recorded original information. Also heavy charged particles may affect the state of memory cells. Studies show that the algorithmic processing reduces the chance of error in the decoded information to values almost comparable with the use of specialized radiation-tolerant memory. Ensuring the integrity of information in the memory controller is through the use of radiation-tolerant MCT-03D processor and the Hamming code protected RAM. To ensure even wear of all memory blocks in memory module, the memory controller will implement their alternation when writing or erasing, and detect bad blocks. Replacing bad blocks with the new ones will come from reserve, which have already been laid by the manufacturer of flash-memory chips in production.

Memory controller will also implement a special accelerated procedure that verify the array of NAND-flash after power-on or after the filing of a special command to the memory controller. Full storage device efficiency (keeping the rate of information exchange) at failure to 25% of the total number of NAND-flash physical blocks is expected to achieve.

Procedures of memory blocks alternating and exchange will take place automatically, providing storage reliable operation with maintaining the required amount of flash-memory. For algorithms and IP-blocks development of these procedures the participation of the St. Petersburg University of Aerospace Instrumentation (SUAI) is involved.

Separate microprocessor's housing CQFP-240 takes a lot of space on a board, in size comparable to two memory modules. Therefore a further step in the development of the developed OMSD we see in increasing of its elements integration, in particular, the establishment of "memory controller - NAND-flash" microassembly (Fig. 3).

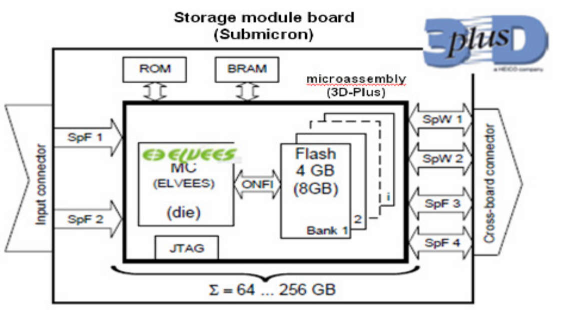


Fig. 4. Application of "memory controller - NAND-flash» microassembly.

This will significantly reduce the size of the storage module by integrating the microprocessor's die to the NAND-flash memory module, which will be achieved through the use of 3D PLUS State-of-the Art stacking technology [9].

IV. 3D PLUS STACKING TECHNOLOGIES

3D PLUS State-of-the Art stacking technologies for SiP (System In Package) allow us to bring the best standard semiconductor devices and technologies in one single highly miniaturized package. The maximum dimension of the 3D microassembly will be 35 mm x 35 mm x 11 mm (L x l x h) (Fig.4).

The electronic parts in the stack of the microassembly will be the MCT-03D in bare die developed by ELVEES RnD center, 64 GByte NAND-flash already Space Qualified from 3D PLUS, some glue logic and the passive components (resistors and capacitors).

A System-In-Package (SiP) consists of a number of dissimilar integrated circuits enclosed in a single highly miniaturized package. The SiP performs all or most of the functions of an electronic system, and, it can contain several silicon components (bare die or package) and passive components.

Key features:

- very small form factor and low profile (more than 80% reductions in size and weight + up-system and in service induced benefits);
- heterogeneous systems : ability to merge different die, package technologies (flip-chip, FBGA, SOT, SOP,….) and form factors;
- improved reliability: space qualified stacking technology, fewer connectors and solder-joints, rugged

Fig. 3. View of the 3D storage module.

to extremely harsh environments;

- improved performance: improved speed and signal integrity (less parasitic elements);
- improved flexibility: modular design enables low-cost system changes, reduce PWB application routing complexity;
- proven "first time right" design and development methodology;
- recognized turn-key design, manufacturing and test.

The Flex Process– SiP Stack (Heterogeneous components and mixed technology stacks) technology flow of 3D PLUS are selected for the design of the microassembly.

This patented process has the unique capability to stack n-High any heterogeneous active, passive, Opto-electronics and MEMS/MOEMS devices in a single highly miniaturized package and with almost no limit for the merging of heterogeneous technologies (standard non modified Die or packages with different sizes).

This 3D technology is based on the stacking of electronic components (chips, commercial packages, sensors) placed on a film layer generally 35mm wide, and so called flex. This solution allows testing and screening the components of each layer before stacking. This is the key feature for building 'n'-High stacks with a very good yield. The flex are then stacked vertically and connected together thanks to a vertical interconnection technique.

This technology allows gaining a factor of at least 10 on weight and volume of the components comparing to existing solutions. This is the most efficient technology for building complex System-In-Packages (SiPs). It enables achieving a combination that cannot be realized with monolithic System-on-Chip (SoC) approaches, and it has a lower development cost and a faster time to market. This capability domain is referenced as FLOW 2 (Fig. 5) and is qualified by European Space Agency (ESA) for Space applications.

3D PLUS has been Capability Approved by European Space Agency (ESA) for the manufacturing of 3D stacked modules.

3D PLUS is qualified as category 1 Manufacturer, the highest qualification level that can be achieved in the ESA specification ECSS-Q-ST-60-05C.

V. Conclusion

There are plans to develop transmission rates of serial I/O duplex SpaceFibre transceivers up to 2.5 Gb/s and 6.25 Gb/s per channel. It is planned to provide an opportunity to work on copper cable with a 50 ohms characteristic impedance and via fiber-optic transceivers (FOT on Fig.1). Further extension the volume of storage device up to 4 Tbytes is under consideration.

References

[1] SpaceWire website, http://www.spacewire.esa.int.

[2] Steve Parkes, Albert Ferrer, Alberto Gonzalez, Chris McClements, SpaceFibre Standard, Draft D, University of Dundee, 2012.

[3] Telecommunication Systems (TCS), "Solid State Drives Overwiev", http://www.telecomsys.com/products/Space-and-Component-Technology/solid-state-drives/solid-state-drives_overview.aspx.

[4] Galleon Embedded Computing, "Product Overview, Rugged Data Recorders", http://www.galleonembedded.com/datarecorders.html.

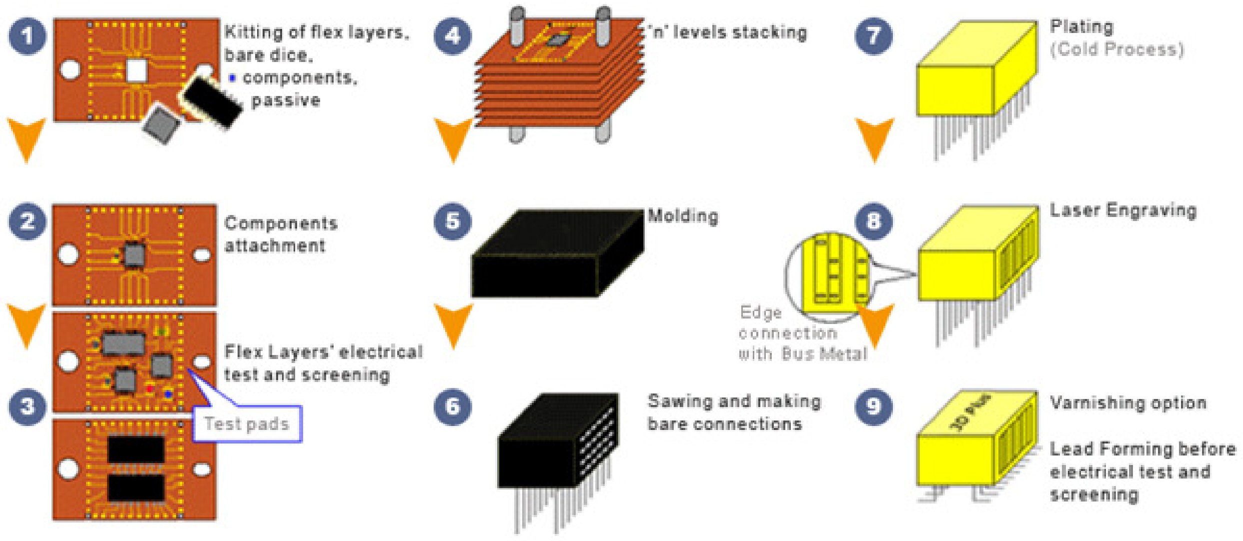
[5] Storage Reviews, "SSD interfaces", http://www.storagereview.com/ssd_interfaces.

[6] Y. Sheynin, T. Solohina and J. Petrichkovich, "SpaceWire technology for parallel systems and on-board distributed complexes", Electronics:Science, Technology, Business, vol. 5, pp. 64-75, 2006.

[7] MCT-xx Radiation Tolerant Microprocessor Data Sheet, ELVEES RnD center, 2013.

[8] 3D PLUS, "Radiation Tolerant Memory, FLASH NAND Product Overview", http://www.3d-plus.com/product.php?type=1&fm=20.

[9] 3D PLUS "System-In-Packages, System-in-Packages (SiP) Product Overview", http://www.3d-plus.com/product.php?type=6.



1
Kitting of flex layers, bare dice, components, passive
4
'n' levels stacking
7
Plating (Cold Process)
2
Components attachment
5
Molding
8
Laser Engraving
Edge connection with Bus Metal
Flex Layers' electrical test and screening
3
Test pads
6
Sawing and making bare connections
9
3D Plus
Varnishing option
Lead Forming before electrical test and screening

Fig. 5. FLOW 2 SiP Process Flow Chart

BepiColombo Solid State Mass Memory employing SpaceWire

Session: SpaceWire Onboard Equipment and Software

Short Paper

Michele De Meo, Giovanni Saldi, Guido Rosani
TAS-I
Gorgonzola (MI), Italy
michele.demeo@thalesaleniaspace.com, giovanni.saldi@thalesaleniaspace.com, guido.rosani@thalesaleniaspace.com

Wahida Gasti, Jeff Noyes, James Windsor
ESA/ESTEC
Noordwijk, The Netherlands
Wahida.Gasti@esa.int, Jeff.Noyes@esa.int, James.Windsor@esa.int

Joachim.Poeckentrup, Reinhard.Eilenberger
EADS Astrium GmbH
Friedrichshafen, Germany
Joachim.Poeckentrup@astrium.eads.net, Reinhard.Eilenberger@astrium.eads.net

***Abstract* — In the frame of the BepiColombo (BC) programme, TAS-I Milano has developed a Solid State Mass Memory unit (BC-SSMM), embedding a SpW network, based on 10 AT7910E SpW router ASICs, connecting 3 internal SpW nodes (Memory, Supervisor A and Supervisor B) to the external ones (9 P/L Instruments, 2 Transfer Frame Generators, On-Board Computer, EGSE). The BC-SSMM behaves as the data exchange centre for the other avionic units of the platform, all interfaced through SpW links (nom. and red.). It implements the RMAP protocol (ECSS-E-ST-50-52C) for router configuration and monitoring, the CCSDS packet transfer protocol (ECSS-E-ST-50-53C) for storage and retrieval of CCSDS packets and a mission specific protocol supporting the exchange of multiple CCSDS packets as cargo of a single SpW packet. The BepiColombo platform is the first flying programme using SpW standard also for the C&C link with the OBC, traditionally based on other mature standards (e.g. Mil 1553 stdbus). Time code distribution supports On Board Time (OBT) synchronisation, replacing the harness needed to distribute dedicated pulse signals. BC gives the opportunity to test the compliance of the AT7910E SpW router ASIC with the ECSS-E-ST-50-12C and to identify important and handy improvements. It also allows to verify the performances of the SpW network, in term of collisions, stalling, routing latency and throughput, in relation to the policy used for logical address mapping and other network configurable features management (timeouts, autostart…etc.). The SpW network is the support to all the higher layer functions implemented inside the BC-SSMM (e.g. PUS services as per ECSS-E-70-41A). A lack inside ECSS-E-ST-50-12C about connectors is finally highlighted.**

I. Scenario

The typical application scenario for the BC-SSMM is a multi-instrument/payload satellite where several independent Users provide data, organized in CCSDS source packets, to be stored on-board into files (named packet stores – PS) and then retrieved, according to the storage and retrieval criteria defined in the Packet Utilization Standard (ECSS-E-70-41A). The BC-SSMM features SpW I/Fs with 9 Payload Instruments (P/L), the On-Board Computer (OBC) and 2 Telemetry Format Generators (TFG), plus an EGSE SpW I/F to support spacecraft assembly integration and test (AIT) on Ground.

The I/O SpW data links and their cross-strap philosophy are shown in Fig. 1. Each P/L, OBC and TFGs can independently operate with either its nominal or redundant SpW I/F at a rate between 10 and 100 Mbps, though in BepiColombo most of these I/Fs run at 10Mbps.

The SSMM exchanges, through these SpW I/Fs, CCSDS packets that can either be TC (from OBC to SSMM, or from OBC to P/Ls through SSMM), or TM (from P/Ls and OBC to SSMM, or from SSMM to TFGs and OBC). Each TM or TC is transferred as cargo of a SpW packet as per ECSS-E-ST-50-53C.

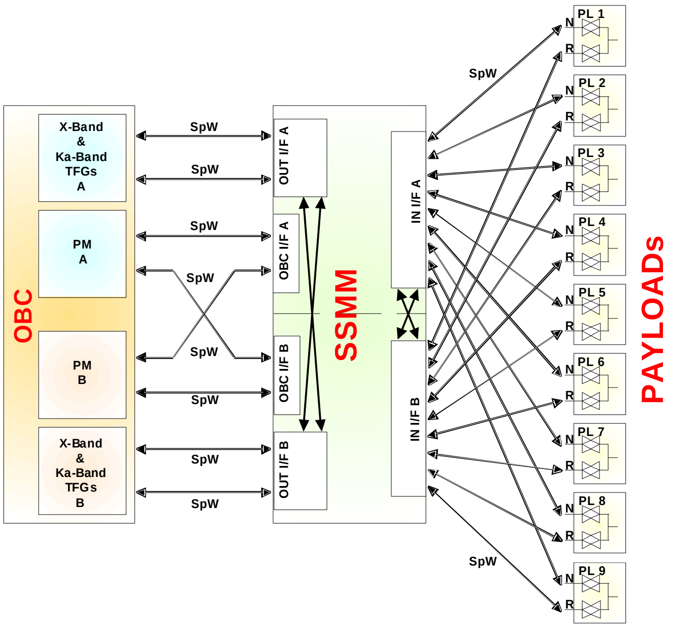


Fig. 1. BC-SSMM SpW links input/output cross-strap philosophy

An exception is represented by non-science CCSDS packets, already collected by the OBC but addressing Ground; many of these CCSDS packets become the cargo of a SpW packet sent from OBC to the Supervisor (SUP). These packets carry in their header a protocol identifier=240, specifically used for the BepiColombo application (as per ECSS-E-ST-50-51C). Another exception is represented by the TM packets towards TFGs, each embedded as cargo of a single SpW packet with only one destination address octet used to represent the down-link Virtual Channel it is transmitted to.

II. Architecture

The SSMM features a self-redundant and Single Point Failure (SPF) free architecture, shown in Fig. 2, consisting of:

- Supervisor modules A/B (managing all the BC SSMM operations) each embedding a Supervisor SpW node and a SpW router
- Memory Array of 3 Memory Modules (MM) as required by mission capacity (384 Gbit EOL) and reliability
- Input modules A/B (managing data storage)
- Output modules A/B (managing data retrieval)
- SUP DC/DC Converter modules A/B each supplying the associated (A or B) Supervisor module
- MEM&IO DC/DC Converter modules A/B supplying (through an SPF free OR) the MMs and the I/O modules.

Redundant functions A and B are housed on separate PCBs. Any combination of "A" and "B" functions is possible except for the Supervisor Module and SUP DC/DC Converter Module; for these last only the A-A or B-B combinations are possible. Any single failure can affect an I/O router of the cross-strap, or the "A" or "B" part of an internal function; the failure remains isolated inside the affected function or router. After recovery of the first failure the SSMM is still fully operational.

Each Input module embeds 3 SpW routers and 1 Write Controller FPGA (WRC-FPGA - ACTEL RTAX2000) which handles storage of the incoming CCSDS packets into the destination PSs inside the MMs. Each Output module embeds 1 SpW router and 1 Read Controller FPGA (RDC-FPGA - ACTEL RTAX2000) which handles retrieval of the outgoing CCSDS packets from the source PSs inside the MMs. From SpW viewpoint the 3 MMs and the 2 I/O modules (without the 8 SpW routers) act as a single SpW node, linked to the parallel ports of the associated 8 SpW routers.

Each Supervisor Module hosts a processor core (ERC32uP, PROM, EEPROM, RAM) running the application software (SW), 1 SpW router and 1 FPGA (OBT&C&C-FPGA - ACTEL RTAX2000), buffering I/O data to/from the SpW network. From SpW viewpoint the processor core and the OBT&C&C-FPGA act as a single SpW node, called the supervisor node, linked to the parallel ports of the associated SpW router.

Therefore the SSMM embeds 3 SpW nodes (supervisor A, supervisor B, and the combined MM & I/O modules) and 10 SpW routers, interfacing the 12 external SpW nodes (plus the EGSE) and providing internal routing paths to allow packet switching between any couple of internal and external nodes.

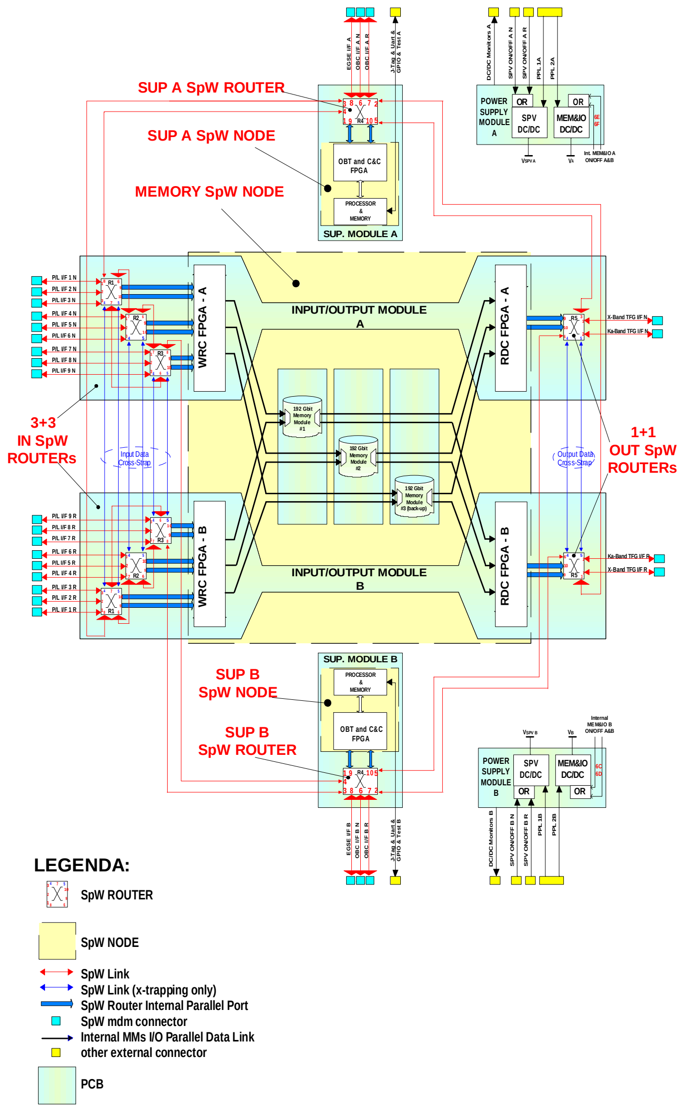


Fig. 2. BC-SSMM block diagram.

Each of the 3 internal nodes is separately supplied; one supervisor node behaves as master and supervises all the memory node operation, while the other supervisor can be activated in service mode only and doesn't interact with the memory node and with the master one; the memory array can therefore save the stored data and file system data even when the master supervisor is switched-off.

The WRC-FPGA of the BC-SSMM splits incoming CCSDS packets (of 4112 bytes max) in segments before recording, while the RDC-FPGA reassembles outgoing CCSDS packets after segments retrieval (128 bytes/segment max.).

Storage and Retrieval of packets occurs in a wormhole fashion through the WRC/RDC-FPGA, from/to a SpW link, to/from the PSs of the Memory Array; therefore the 9 users and the 2 TFGs can concurrently and randomly access the same memory module without latencies, via small temporary buffers inside the WRC and RDC-FPGA. This ensures that the max I/O throughput can be achieved under any random traffic condition even when all the I/O accesses are concentrated onto one MM of the Memory Array.

Therefore the I/O module relieves the SW running on the operational supervisor from any overhead due to storage/retrieval of packets into/from (the PSs of) the Memory Array. Dynamic SW intervention is limited to allocation of memory areas needed for the ongoing storage/retrieval operations (typically assignment of sectors to be written/read for each involved PS). The Application SW higher layer functions implement the data storage according to PUS service 15 and the data retrieval according to PUS service 13 and 15. Data storage is done routing each packet to a PS according to its application process identifier (APID). An exception is the storage of P/L non-science packets which is done into a single cache PS according to the logical address; the same applies for the storage into the PS used as destination of a copy operation.

Figure 3 shows the open BC-SSMM, while Fig. 4 shows the actual BC-SSMM PFM.

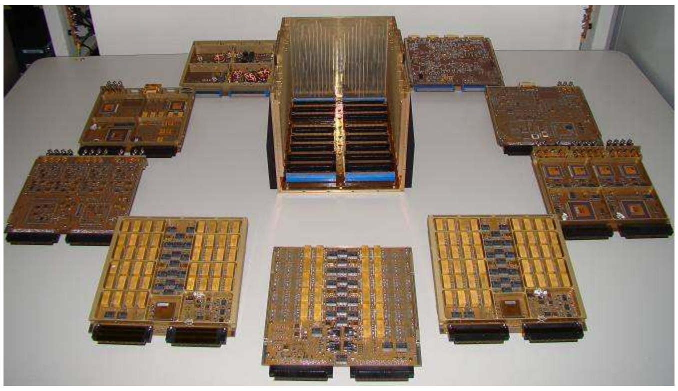


Fig. 3. BC-SSMM PFM open box view

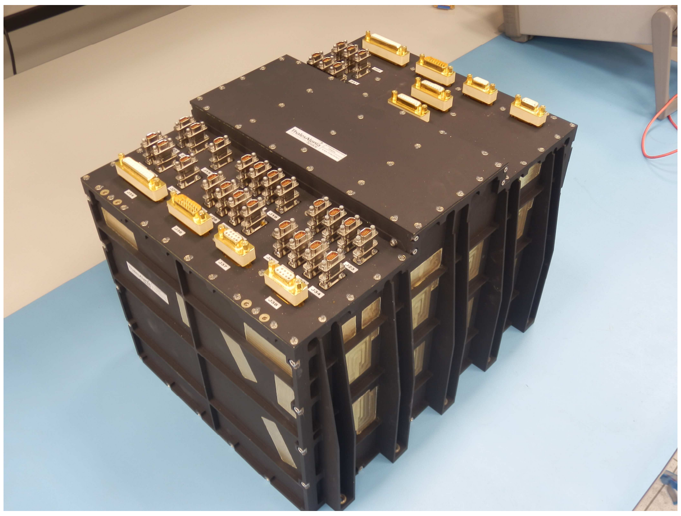


Fig. 4. BC-SSMM Proto Flight Model.

The BC-SSMM consists of 9 boards, plugging into a passive motherboard as follows:

- 1 board for each MM → 3 boards
- 1 board housing 1 I/O module (A/B) → 2 boards
- 1 board housing 1 (A/B) Supervisor module → 2 boards
- 1 board housing 1 SUP DC/DC Conv. and 1 MM&IO DC/DC Conv. → 2 boards.

III. The BC-SSMM SpW Network

A. Topology & Mapping

The network topology is the best compromise satisfying the user requirements in terms of performances (i.e. overall data throughput=~60Mbps, latency<10ms) and budgets (mass<12Kg; power consumption<60W; dimension LxWxH= 340x282x251mm; reliability=0,994 over 7,5 years). It is not based on a common scheme (e.g. tree, ring, star, grid based tor etc), but has the purpose to ensure that each SpW node (P/Ls, OBC, TFGs and the 3 internal ones) can operate its I/O data without suffering the I/O operations of the other nodes; indeed, in the BepiColombo scenario, the BC-SSMM can simultaneously sustains the following packet flows:

- Science TM packets from 9 P/Ls to memory
- TC packets from OBC to 9 P/Ls
- P/L non-science TM packets from 9 P/Ls to memory (these include event packets, acknowledge (ACK) packets to Ground telecommands and ACK packets to OBC telecommands; they are all accumulated into a cache PS for later sorting)
- Non-science TM packets from memory to OBC (these are the event and ACK packets of the previous point; OBC sorts its own packets and assembles multiple TM packets (4112-byte) for Ground into single SpW packets sent to the master supervisor, which then disassembles them and sends the elementary non-science TM packets to memory for later down-link
- Multiple non-science packets from OBC to the master supervisor (these include the ACK to Ground telecommands of the previous point to be stored and later down-linked)
- Non-Science TM packets from master supervisor to memory (obtained after disassembling SpW packets of the previous point, each carrying multiple CCSDS packets re-transmitted by the OBC)
- TC packets from OBC to SUP A
- TC packets from OBC to SUP B
- SSMM TM packets from SUP A to OBC
- SSMM TM packets from SUP B to OBC
- TM packets from memory to master supervisor (to be assembled by application SW into 4112-byte packets named File Data Units - FDU- as per PUS service 13)
- TM packets from master supervisor to memory (4112-byte FDUs as at the previous point)

- TM packets (i.e. 4112-byte FDUs of the previous point) from memory to the TFG supporting PUS service 13
- TM packets from memory to the TFG supporting PUS service 15
- TM packets from memory to memory (copy operations)

Since most of the packet flows reach the memory node, it provides six input ports to sink up to 6 packets at a time from the network (i.e. from 12 source nodes via the input routers). Conversely two output ports are sufficient to retrieve up to 2 packets at a time from memory and forward towards the network (i.e. towards 5 destination nodes via the output routers). Other internal SpW links interconnect the 10 internal SpW routers, to provide routing paths to any packet flow.

Each packet flow is point to point exchanged between a couple of SpW nodes. Though the assignment of more than one logical address per node is unusual, each internal node of the SSMM is associated to as many SpW logical addresses as the number of packet flows it is reached by; this allows the identification of the sender and of the packet type in the receiving node.

The assignment of logical addresses (reflected into the routing tables of the routers crossed by the corresponding packet flow) is such that each packet follows the shortest path to reach its destination, minimising collisions with other packets.

Each packet flow can follow more than one routing path from the same source node to the same destination node. The number of possible routing paths depends on:

- the internal redundancy configuration (A or B functions)
- the external (nom or red) SpW I/Fs used by each P/L, by each TFG and by the OBC
- the SpW routers of the I/O cross-strapping which may be all operating, or 1 of them failed (8 cases).

After detection of an internal failure (affecting any A or B function, or a SpW router of the I/O cross-strapping), the BC-SSMM application SW re-programs the routing tables of the active SpW routers to reallocate all packet flows in the new degraded configuration. Routing tables programming is performed by the master supervisor through command/reply RMAP packets (as per ECSS-E-ST-50-52C) exchanged with each target router, using path addressing. Alternative routing paths from supervisor to each target router are available for RMAP packets too, to cope with possible failure of any intermediate router.

B. Collisions & Latency

Each SpW node (the BC-SSMM internal 3, the 9 P/Ls and the OBC), generates packets asynchronously with respect to the other nodes and therefore collisions may occur. Nevertheless the additional wait time due to a collision is well tolerated by each SpW node since it cannot take more than the transfer time of the longest CCSDS packet (4,1164ms for 4112-byte on a 10Mbps SpW link without null characters). The transfer time is guaranteed by the properly sized TX and RX buffer equipping each source and destination node of the BC platform; for the SSMM internal nodes:

- the memory node directly sinks/sources any incoming/outgoing packet into/from the addressed PS
- each supervisor node does the same via 5KB rx/tx buffers read and written by the application SW

Therefore in case two longest packets generated by node A and node B, addressing the same node C, collide at an output port of a router, the unlucky packet has to wait 4,1164ms max. for the output port to become free. This complies with the latency required for the BC platform (<10ms from P/L to memory).

The selected asynchronous approach saves the implementation of a synchronization process spread over all network nodes (e.g. assigning precise time slots to each source node) to prevent collision of packets at any router output port.

Inside the SSMM, before the first failure of a router or of a FPGA, up to 4 packets may collide onto 2 output ports of an input router (configured in group adaptive routing mode); the last 2 packets shall wait until the first 2 packets free the 2 output ports. Latency is here to be intended as the wait time for the SpW routing path to become free from the source node up to the destination node. Once the routing path is free, then the packet can be forwarded according to the signalling rate of the slowest crossed link. Therefore the actual transfer time of a packet through the SSMM is the sum of the latency and of the transfer time of a packet through a free routing path.

With the purpose to minimise collisions, the non science packets from the 9 P/Ls to the OBC are stored into a cache PS of the SSMM and then forwarded to the OBC rather than being wormhole routed towards the OBC. In the former case the memory node sinks 6 packets at a time and the collision of 9 packets onto six output ports allows the 3 unlucky P/Ls pay a 1x wait time (<=4,1164ms); in the latter case the OBC would sink 1 packet at a time and P/Ls would pay up to 8x wait time. Therefore the selected store&forward scheme, though complex, prevents that a P/L experiences an excessive wait time before transmission of a packet, i.e. the risk to loose internally generated data due to overflow of its TX buffer.

C. Stalling

Once the routing path from source node to destination node is free, additional wait time would be due to the source or destination which, in case of failure, causes a packet to stall on its routing path. Stalling propagates to all the packet flows sharing 1 or more SpW links of their routing path with that of the stalled packet, as established by network topology and logical address mapping.

The SSMM, no matter as source or destination, by design prevents any stalling unless an internal router or node is affected by a permanent failure (hopefully unlikely), however recovered via the foreseen switch-over procedure. Nevertheless the SSMM is equipped to handle and report to the OBC stalling due to the P/Ls. The SSMM doesn't handle stalling due to the OBC (and the TFGs it embeds), but it doesn't hang up and resumes any halted operation as soon as the OBC (i.e. the highest hierarchical level of the BC platform) will recover the stalling; in particular:

- stalling of a TM packet from a P/L towards the Memory Array, due to the P/L, is supported by a timeout mechanism which, causing a temporary disconnection of the P/L SpW link, sets free the path towards the Memory Array for the other P/L packets colliding onto an output port of the same input router. The stalled packet is then received, after the link disconnection, in a truncated shape (EEP terminated) and stored into the Memory Array. The event is reported to the OBC and normal operation immediately restarts.
- stalling of a TC packet from OBC towards a P/L:
  - if caused by the destination P/L (i.e. the P/L sinks no more data), is supported by the timeout mechanism of the input routers, configured in "Watchdog Timer" mode, which flushes automatically the stalled packet; then an event is reported to the OBC. After packet removal the OBC can forward a subsequent TC packet either to the SSMM or to the P/Ls.
  - if caused by the source OBC (i.e. the OBC sources no more data), can be recovered only by the OBC. If a copy operation (memory to memory) is running in background, then it stalls too together with all the other retrieval operations from Memory to TFGs, to OBC (non-science packets) and to master supervisor (packets for FDUs assembly). Though the timeout mechanism of the input router truncates (with an EEP) the TC packet forwarded to the P/L, no timeout mechanism is configured in the supervisor router and output routers to ensure consistency of the ongoing operations, which are immediately resumed as soon as the OBC will remove the stalled packet.
- stalling of a TC packet from OBC to SSMM, due to the source OBC, can be recovered only by the OBC. The stalling has no effect on other packet flows, since it affects only the SpW link between the SSMM and the OBC.
- Stalling of a TM packet towards the OBC, due to the destination OBC, halts the ACK TM from Supervisor to OBC, together with all the other retrieval operations from Memory towards TFGs, OBC (non-science packets), master supervisor (packets for FDUs assembly) and Memory (copy). No timeout mechanism is implemented to ensure consistency of the ongoing operations which are immediately resumed as soon as the OBC will remove the stalled packet.
- Stalling of a TM packet from Memory Array to the TFG supporting service 13 Virtual Channel (VC), due to the destination TFG, halts the background operation inside the SSMM (i.e. file copy and FDU assembly). As the TFGs are part of the OBC, it is up the OBC to remove the stalled packet (e.g. by disconnecting the TFG link)
- Stalling of a TM packet from Memory Array to the TFG supporting service 15 VC, due to the destination TFG, halts the non-science packets from the Memory Array to OBC. As the TFGs are part of the OBC, it is up the OBC to remove the stalled packet (e.g. by disconnecting the TFG link).

In all cases stalling due to OBC or TFGs doesn't affect storage of science packets. Conversely if the OBC stalls for more than ~3sec while sinking non science packets retrieved from the cache PS, this last may overflow and subsequent non-science packets generated by the P/Ls are discarded (but no stalling on any P/L SpW link occurs).

D. Throughput

The average input net data rate (=stored bit/sec) from the 9 P/Ls is expected to be ~50Mbps. The SpW signalling rate is set at 10 Mbps for 8 P/Ls and 100Mbps for 1 P/L. Since the input module multiplexes the 9 P/L SpW links onto the 6 parallel ports of the active WRC-FPGA, it is able to sink 6 packets at a time, 5 at 10mbps and 1 at 100Mbps, equivalent to a peak net input data rate of 120Mbps which can be continuously sustained from the input SpW links up to the SDRAM ICs of the addressed MMs.

The average output net data rate (=retrieved bit/sec) is limited by the actual sink rate of each destination node. It is expected to be ~1,5Mbps towards the 2 TFGs, as limited by the down-link RF bandwidth. The SpW signalling rate is set at 10 Mbps for both TFG SpW links; the available net output data rate towards the 2 TFG SpW links (16Mbps) is never reached (except at TFGs' reset when their input buffers are still empty).

Therefore the SSMM throughput on the retrieval path provides a large margin with respect to 1,5Mbps; this margin is used to retrieve packets in background towards the internal memory (for copy), towards the master supervisor (for FDU assembly as per PUS service 13) and towards the OBC (for non-science packets). The available net retrieval data rate towards internal destination nodes is 16Mbps and is fixed by the single internal SpW link from the output router to the supervisor router running at 20 Mbps (as it occurs for all the internal SpW links).

The retrieval of packets for background operation (copy and FDU assembly as per PUS service 13), is interleaved (through the same RDC-FPGA output port linked to an input parallel port of the output router) with the retrieval of 4112-byte FDUs towards the TFG supporting service 13. The retrieval of an FDU takes from 47ms to 82ms (when the bit rate of the TFG supporting PUS service 13 varies from 700 to 400 Kbps); this is not critical for the background operations which lock their average retrieval rate at 2x the TFG rate, though transferring each packet at 16Mbps (without null characters).

The retrieval of non science packets from the cache PS towards the OBC (flow controlled at 57 pkt/250msec to prevent problems with the OBC central SW) occurs at 8Mbps (i.e. the OBC net sink rate) with high priority through the other RDC-FPGA output port shared with the packets towards the TFG supporting service 15. The throughput on the 10 Mbps SpW link between SSMM and OBC is limited by the capability to sink/source packets of the application SW running in both units. SSMM stand-alone test with all the SSMM SpW links running at a 100Mbps SpW signalling rate has been successfully performed.

IV. AT7910E SpW Router ASIC experience

The SpW Router has demonstrated to be a well conceived and robust design. The BC-SSMM, exploiting its routing capabilities, features now performances (mass, power consumption, reliability….) which could have been never reached with alternative state of the art devices. The application SW of the BC-SSMM doesn't suffer any overhead due to the routing of packets to/from P/Ls and towards TFGs, since it handles only the SpW packets sunk /sourced by the master supervisor SpW node.

Nevertheless the experience of TAS-I Milano has highlighted few minor problems solved with a work-around and few nice to have improvements which would have simplified the BC-SSMM development; in particular:

- a true data sheet of the device is still missing, though this lack is partially compensated by the user manual
- the initial signalling rate of a SpW output port may not be 10Mbps, unless 10 Mbps is initialized inside the corresponding port control register
- the following features, though useless for applications with stand-alone remote routers, would be welcome for any unit embedding routers controlled by a host microprocessor:
  - a dedicated interrupt pin, asserted each time an error is sensed on a router port (with mask capability)
  - possibility to read/write from/into any internal register without any RMAP packet exchange (e.g. via a host interface obtained by combining the current Status Interface and Time-code Interface)
  - option to automatically append the CRC octets to each RMAP packet injected into the parallel port and addressed to the same router or to other routers of the same SpW network.

V. SpW Std Connector

The ECSS-E-ST-50-12C doesn't deal with SpW cables linking two SpW nodes placed in two locations "A" and "B", physically separated by an intermediate barrier "C" as it occurs for:

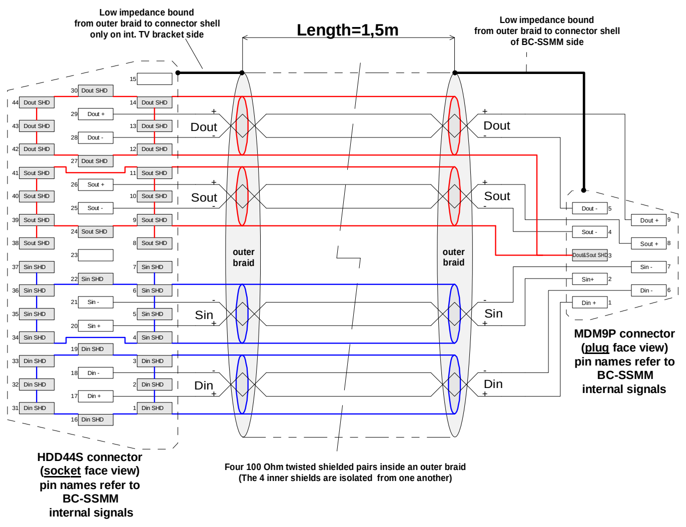
- a unit in a thermal-vacuum chamber linked to an external test equipment during AIT on Ground
- two units of the same satellite platform with a wall in between.

A SpW link like this consists of three main sections:

- the cable from "A" to the left side of "C"
- the wiring through "C"
- the cable from the right side of "C" to "B"

The 9-contact micro-miniature D-type (MDM) connector is not suitable for both sides of "C" since in case the 4 shields of the twisted pairs would be tight to pin 3 of the MDM connector (on each side of "C"), the signal grounds of the SpW nodes "A" and "B" would be shorted together through the link. This contrasts with figure 5-3 in chapter 5.4 of the ECSS-E-ST-50-12C std.

With the purpose to maintain separation of the 4 inner shields, still matching the 100 Ohm cable impedance also through the intermediate barrier (the shield of each twisted pair has been tight to 8 or 9 contacts around the 2 contacts of the associated differential signal), TAS-I Milano developed a specific SpW cable, terminated on one side with a 9-contact MDM connector and on the opposite side with a 44-contact high density connector, as shown in Fig. 4.



Low impedance bound from outer braid to connector shell only on int. TV bracket side
Length=1,5m
Low impedance bound from outer braid to connector shell of BC-SSMM side
Dout
Sout
Sin
Din
outer braid
outer braid
MDM9P connector (plug face view) pin names refer to BC-SSMM internal signals
HDD44S connector (socket face view) pin names refer to BC-SSMM internal signals
Four 100 Ohm twisted shielded pairs inside an outer braid (The 4 inner shields are isolated from one another)

Fig. 5. BC-SSMM specific SpW cable for Thermal Vacuum Chamber.

The BC-SSMM tests in the TV chamber have been successfully performed up to 100Mbps using two cables like this (1,5m + 5m) for each SpW link connecting the internal SSMM to its external test equipment.

VI. Conclusion

The use of the SpW standard is becoming widespread over many programmes, though often limited to the implementation of point to point interconnections replacing alternative solutions (e.g. RS-422, Mil1553-std…..etc.). Conversely the BC-SSMM is the first representative implementation of an actual SpW network with routers and nodes fitting the purposes of the ECSS-E-ST-50-12C. The flexibility of a design based on a SpW network, is confirmed by the immediate reuse of the BC-SSMM in Solar Orbiter, applying little adjustments to house one more instrument and to map additional packet flows. The SSMM internal SpW network is also suitable to implement additional services like the CCSDS File Delivery Protocol (already implemented as customization of PUS Service 13 and 15).

Networks & Protocols (Short)

Deterministic Scheduling of SpaceWire Data Streams

Networks and Protocols, Short Paper

Dmitry Raszhivin, Yuriy Sheynin, Alexey Abramov
St.Petersburg University of Aerospace Instrumentation
Saint-Petersburg, Russia
dmitry.raszhivin@guap.ru

Abstract—**The topology of an onboard computer network is determined by the physical location of nodes (sensors, computer modules, databases, etc). Data flows in network are determined by a sender and receiver pair (or set of receivers in case of multicasting). A packet flow could be organized through a number of intermediate switches. Time division multiple access allows several users to share the same channel by dividing their channel access time into different time slots. Operation of a network is managed by a schedule that defines which node is allowed to initiate a transmission at any particular time. This table shall be compiled on the one hand to prevent conflicts in the network resources usage, and on the other hand to utilize the resources as the highest level as possible. The aim of the work presented in this paper is to develop a method of finding the best routes for all data flows in a network and compiling an optimal schedule table, which guarantees deterministic data transmission.**

Index Terms— **SpaceWire-D, Scheduling**

I. Introduction

Nowadays time-scheduled protocols over SpaceWire [1] (SpaceWire-D [2], SpaceWire-T [3]) are actively discussed and developed. These protocols can be developed as a network service running at existing SpaceWire equipment [8]. Deterministic data delivery with predictable characteristics is achieved by using time-division multiplexing, that is managed by a schedule table. SpaceWire-D provides several types of schedule table, but does not offer any algorithm of constructing such a tables. By routing data streams through one or another path using different intermediate switches different scheduling tables with discrepant characteristics would be obtained.

II. Graph coloring approach to solve scheduling task

In the graph theory, graph coloring is a procedure of the assignment of labels traditionally called "colors" to the elements of the graph, in such a way that no two adjacent vertices share the same color [5]. The chromatic number of a graph G is the smallest number of colors K needed to correctly color all the vertices of G, i.e. the smallest value of k possible to obtain a k-coloring.

Scheduling task can be referred to vertex coloring task. Consider n data flows $\{J_i\}_{(i-1)}^{N}$ and competition matrix $M \equiv \{m_{ij}\}$, where m_{ij} is equal to one if J_i competes with J_j, and to zero otherwise. Representing every flow J_i with node A_i and connect with undirected edge e_{ij} nodes A_i and A_j if $m_{ij} = 1$, schedule table making task can be considered as a coloring task for graph G witch consist of nodes $V(G) \equiv \{A_i\}_{(i-1)}^{N}$ and edges $E(G) \equiv \{e_{ij}\}$. Minimal value of time-slots, required for deterministic schedule organization, is equal to the chromatic number $k(G)$ of a graph G [6].

Examine a the network shown in Fig. 1. This network consists of five nodes and includes five data flows defined in table I.

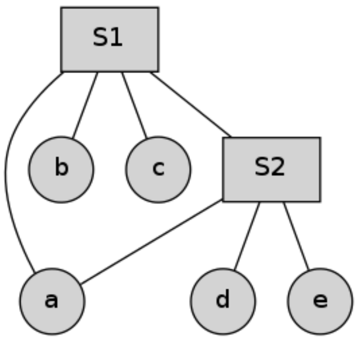


Fig. 1 Example network

TABLE I. CORRESPONDENCE BETWEEN FLOWS AND NODES

Flow	Source	Destination
X_1	a	b
X_2	a	c
X_3	a	d
X_4	b	d
X_5	c	e

A competition matrix for this data flows is presented in table II. In this example flow X_1 competes with flows X_2 and X_4, flow X_2 competes with flows X_1 and X_5 etc.

TABLE II. SAMPLE OF COMPETITION MATRIX

	X_1	X_2	X_3	X_4	X_5
X_1	*	1	0	1	0
X_2	1	*	0	0	1
X_3	0	0	*	1	0
X_4	1	0	1	*	1
X_5	0	1	0	1	*

This competition matrix can be interpreted as adjacency matrix for a new graph *H*. The result of coloring procedure for this graph *H* is shown in Fig. 2. For given example coloring is absolutely trivial and chromatic number is two. The network resources are allocated for data flows X_1, X_3 and X_5 in first time-slot and for data flows X_2 and X_4 in the second time-slot.

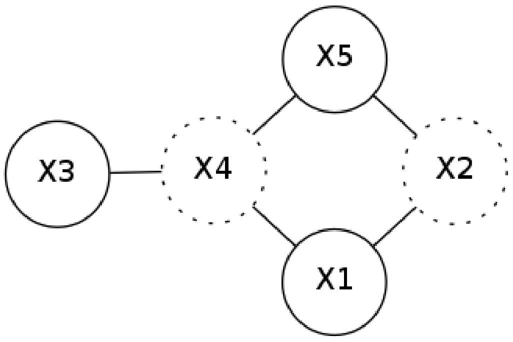


Fig. 2 Competition matrix colored graph

III. COLORING

During our work we studied and developed several algorithms for graph coloring.

A. Greedy coloring

A greedy coloring is a coloring of the vertices of a graph formed by a greedy algorithm that considers the vertices of the graph in sequence and assigns to each vertex an available color. Generally, greedy colorings do not use the minimum number of possible colors, however they have been deployed in mathematics as a technique for proving other results about colorings as well as in computer science as a heuristic to find colorings with few colors.

B. Liquid scheduling

The capacity of a network can be called as its liquid throughput. The liquid throughput corresponds to the flow of a liquid in an equivalent network of pipes. E. Gabrielyan and R.D. Hersch [7] proposed to schedule the data flows of a network in accordance with a schedule yielding the liquid throughput. Such a schedule called liquid schedule takes into account the underlying network topology and ensures an optimal utilization of all bottleneck links. In order to build a liquid schedule the traffic is partitioned into time slots comprising mutually non-congesting flows which keep all bottleneck links busy during all time slots. The search for mutually non-congesting flows utilizing all bottleneck links is of exponential complexity. An efficient algorithm presented in [7] traverses the search space non-redundantly and limits the search to only those sets of flows, which are non-congesting and use all bottleneck links.

C. Heuristic coloring with QoS requirements

Time division of a bandwidth guarantees deterministic delays, but it is very important to predict throughput as well as transmission delays. Greedy coloring and liquid scheduling don't take in mind any throughput requirements for data flows, so we developed an algorithm that builds scheduling table based on several QoS requirements. Several steps are performed to reduce a search tree of the NP-complete brute-force problem:

1. Find all possible routes for all data flows. Different routes can use different switches and channels.

$$X_i = R_{i1}, R_{i2}, R_{i3}, \ldots$$

2. Group non-congesting routes with close transmission times (Fig. 3)

$$G_\alpha = R_{i1}, R_{j3}, R_{k1}, \ldots$$

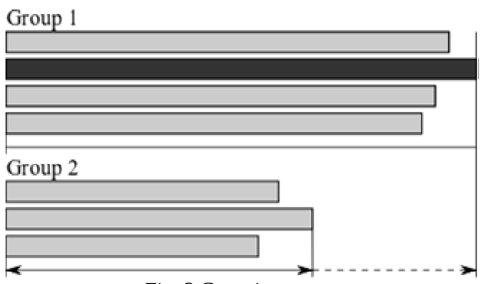


Fig. 3 Grouping routes

This grouping consists of two stages:

- Calculating transmission time for all data flows through all routes with formula above:

$$t = \sum_{i=0}^{N} \frac{d_i}{c_i} + \sum_{i=1}^{N} (m_i \times bt_{i-1} + T\omega_i + Tb_i) + L \times \max(bt_0, \ldots, bt_N)$$

where: N is the number of transit channels, d_i is the length of channel i, c_i is the propagation speed of transmission medium of channel i, m_i is size of buffer in switch i, $T\omega_i$ is the processing time in switch i, Tb_i is the blocking time in switch i, L is the length of packet in bits, and bt is the one bit transmission time in channel i. Tb_i considered to be zero at SpaceWire-D.

- Calculating normalized transmission time for all data flows through all routes:

$$t_i^{'} = \frac{t_i}{\max(t_{1 \ldots N})}$$

where $t_i^{'}$ - normalized transmission time for route i.

- Calculating normalized average transmission time for all data flows:

$$T_{average} = \frac{\sum_{i=1}^{N} \sum_{j=i+1}^{N} \left| t_i^{'} - t_j^{'} \right|}{\frac{N \times (N-1)}{2}}$$

- Creating groups, using breadth-first search. Resulting groups must not compete one with other and satisfy next condition:

$$\left| t_i^{'} - t_j^{'} \right| \le T_{average}$$

3. Form a schedule table from a set of groups, using one group as one time-slot. Breadth-first or depth-first search are used to find acceptable for all QoS requirements schedule table.

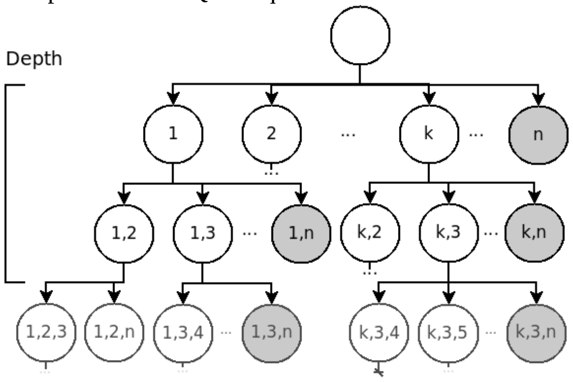


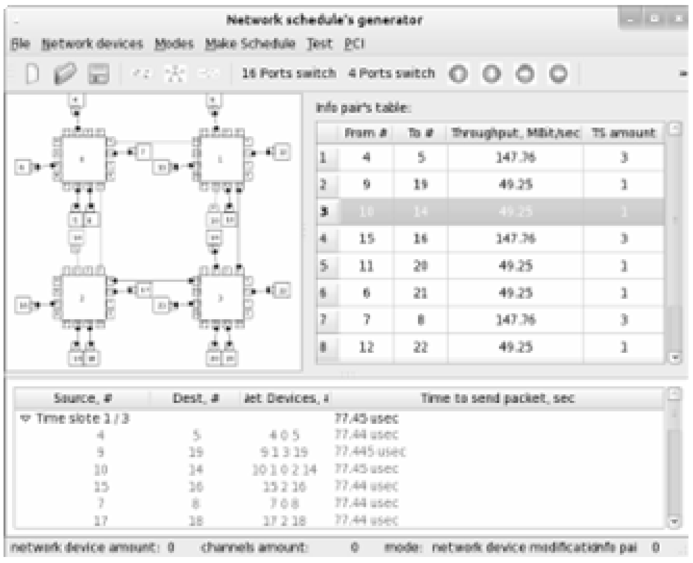
Fig. 4 Schedule table creating

IV. Software

A software tool "Network TDMA Scheduler" implements the algorithms described above. It has GUI (Fig. 5) and allows compiling schedule table and predicting time characteristics of the obtained schedule.

The workflow consists of the following steps:

1. Creating a network with one of several ways:
   - drawing with build-in primitives,
   - loading from a XML file,
   - using build-in network generator with several topologies: star, tree or ring.
2. Defining data flows with one of several ways:
   - choosing by hands,
   - loading from XML file,
   - using build-in random generator.
3. Calculating a network schedule with one of the coloring algorithms.
4. Computing characteristics of the obtained schedule.



Info pair's table:

	From #	To #	Throughput, MBit/sec	TS amount
1	4	5	147.76	3
2	9	19	49.25	1
3	10	14	49.25	1
4	15	16	147.76	3
5	11	20	49.25	1
6	6	21	49.25	1
7	7	8	147.76	3
8	12	22	49.25	1

Source, #	Dest, #	Net Devices, #	Time to send packet, sec
Time slote 1 / 3			77.45 usec
4	5	4 0 5	77.44 usec
9	19	9 1 3 19	77.445 usec
10	14	10 1 0 2 14	77.45 usec
15	16	15 2 16	77.44 usec
7	8	7 0 8	77.44 usec
17	18	17 2 18	77.44 usec

network device amount: 0 channels amount: 0 mode: network device modificationfs pai 0

Fig. 5 Network TDMA Scheduler interface

All schedule tables, compiled with the algorithms examined above, have some characteristics that affect network performance. The time-slot length, the number of time-slots in the epoch, and the number of data flows transmitted during one time-slot are related to the schedule table characteristics. A packet transfer delay and the aggregate throughput are related to the network characteristics.

The aggregate network throughput can be calculated as following:

$$s = \sum_{i=0}^{k} \frac{(L_i \times t_i)}{E}$$

where: k is the number of streams, t is the number of time-slots assigned to the stream i, E is the epoch length in sec.

Build-in network generator allows creating networks with star, tree or ring topologies. Automated tool permits compute different network characteristics on dynamic topologies. Fig. 6 shows growth of aggregate throughput on dynamically scaling ring network. At first step the network consists of 4 switches, 8 nodes and 4 data flows between nodes. At every new step one switch, two nodes and two data flows are added to the network. Aggregate throughput of the network at each step is shown at Fig. 6, Fig. 7 shows number of time-slots (lower values are the best).

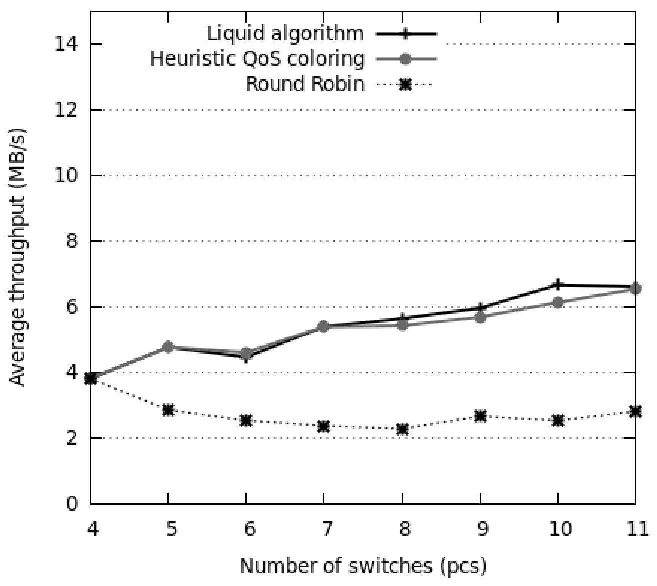


Fig. 6 Aggregate throughput of network

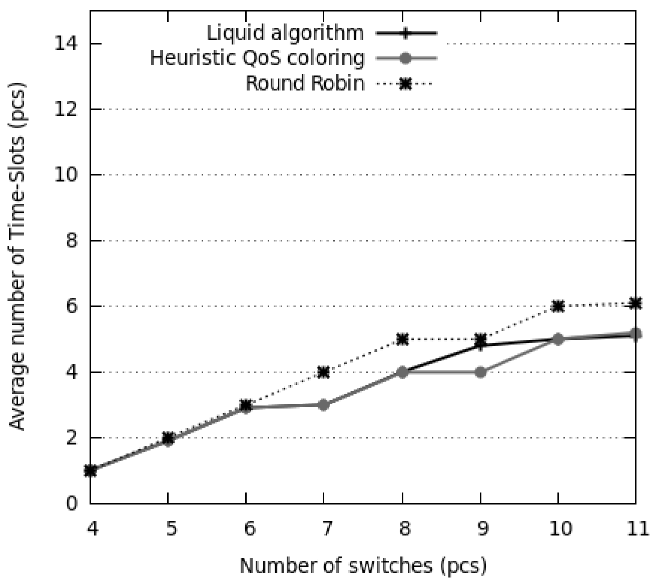


Fig. 7 Number of Time-Slots

Liquid schedule gives results close to results by heuristic coloring as it is seen above, but it is important to remember that such QoS requirements as throughput and latency can be guaranteed only by heuristic coloring.

V. Conclusion

In the paper the task of deterministic data transmission with guaranteed delivery time in SpaceWire network was studied, methods of scheduling tables construction for networks with time division multiplexing were reviewed. Algorithms of compiling such tables were proposed. A software tool "Network TDMA Scheduler" implements the algorithms described above. It allows compiling schedule table and predicting time characteristics of the obtained schedule. All schedule tables, compiled with the algorithms examined above, have some characteristics that affect network performance. The time-slot length, the number of time-slots in the epoch, and the number of data flows transmitted during one time-slot are related to the schedule table characteristics. These characteristics were studied for several networks with standard topologies, advantages and disadvantages of algorithms were discovered.

References

[1] ECSS, "SpaceWire - Links, nodes, routers and networks", ECSS-E-ST-50-12C, July 2008

[2] S.Parkes, A. Ferrer "SpaceWire-D: Deterministic Data Delivery with SpaceWire", Proceedings of the 3rd International SpaceWire Conference, St. Petersburg, 2010

[3] S.M. Parkes, A. Ferrer-Florit, "SpaceWire-T Initial Protocol Definition", Draft 3.1, August 2009.

[4] "Remote Memory Access Protocol", ECSS-E-50-11, Draft E, December 2005.

[5] N. Christofides. Graph Theory - an Algorithmic Approach. Academic Press 1975.

[6] Marx, Daniel (2004), "Graph colorings problems and their applications in scheduling", Periodica Polytechnica, Electrical Engineering, 48, pp. 11-16

[7] E. Gabrielyan and R.D. Hersch, "Efficient Liquid Schedule Search Strategies for Collective Communications", ReCALL, 2004, pp. 760-766.

[8] Liudmila Koblyakova, Yuriy Sheynin, Dmitry Raszhivin. Real-time signaling in networked embedded systems. International SpaceWire Conference, St.Petersburg 2010. Conference Proceedings. ISBN: 978-0-9557196-2-2, p 385-388

A network Device driver Framework for SpaceWire

Session: SpaceWire Network and Protocols, Short Paper

Qiang Wan, Baokang Zhao, Bo Liu, Chunqing Wu
School of Computer Science
National University of Defense Technology
Changsha, Hunan, CHINA
wanqiang @nudt.edu.cn
bkzhao @nudt.edu.cn

***Abstract*—SpaceWire is becoming more and more popular in space applications due to its technical advantages, including reliability, low power and fault protection, etc. SpaceWire networks also provide an efficient approach to connect on board equipments and function units. Yet, comparing with widely deployed terrestrial network protocols such as Ethernet, it is difficult to develop software for SpaceWire buses since it is relatively a new type of network. Therefore, it will be very useful if the SpaceWire interfaces can be regarded as common network interfaces.**

Towards this end, in this paper, we propose vSpWNet, a new network device driver framework to build virtual SpaceWire Network Interfaces. vSpWNet consists of several routines, including the SpaceWire hardware access encapsulation functions, Network frame packaging and unpacking functions, packets encoding and decoding functions, etc.

We have integrated our proposed vSpWNet into the protocol stack of VxWorks. Experimental results show that, our proposed vSpWNet platform performs well in a real OBC board. Moreover, our vSpWNet approach can be ported into other Operation Systems, including RTEMS, eCOS, etc.

***Keywords*— SpaceWire, vSpWNet, VxWorks.**

I. Introduction

SpaceWire[1, 2] is a spacecraft communication network based in part on the IEEE 1355 standard of communications. It is coordinated by the European Space Agency (ESA) in collaboration with international space agencies including NASA, JAXA and RKA. Within a SpaceWire network the nodes are connected through low-cost, low-latency, full-duplex, point-to-point serial links and packet switching wormhole routing routers. SpaceWire covers two (physical and data-link) of the seven layers of the OSI model for communications.

Yet, comparing with the widely deployed terrestrial network interfaces such as Ethernet[9], there are very few software for SpaceWire buses since it is relatively a new type of bus standard[6, 7]. Therefore, it will be very useful if the SpaceWire interfaces can be used as a common network interface.

In this paper, we are the first to propose an idea of vSpWNet, a new network device driver framework to build virtual SpaceWire Network Interfaces. It can package the SpaceWire buses, simulate SpaceWire buses as Ethernet interfaces. We named this new network device driver framework vSpWNet. In the premise of availability, vSpWNet can be ported into other operation system easily, including RTEMS, eCOS and so on.

The implementations of vSpWNet can greatly enhance its practicality. Most of the applications based on Ethernet interfaces can works on vSpWNet directly. Then, a lot of time and energy resources can be saved.

The problems we need to solve can be summarized as follows,

a) How to package the SpaceWire and simulate it as Ethernet interfaces, since SpaceWire buses and Ehternet interfaces are quite different.
b) How to make vSpWNet can be ported into other operation system easily, since protocol layer interfaces are different on different operation system.

The rest of this paper is organized as follows. in section II, the vSpWNet is proposed. The design of vSpWNet is represented in section III. We describe the implementations of vSpWNet in section IV. And the test result is shown in section V. Finally, we conclude in section VI.

II. Our Proposed vSpWNet Framework

SpaceWire is becoming more and more popular in space applications due to its technical advantages, including reliability, low power and fault protection, etc[3]. However, comparing with the widely deployed terrestrial network protocols such as Ethernet, it is difficult to develop software for SpaceWire buses since it is relatively a new type of network interface while Ethernet network interfaces are already very mature. If we develop all the application over again, it surely will consume huge resource. Taking all these into consideration, we propose vSpWNet.

vSpWNet is short for virtual SpaceWire network. Fig 1 shows the position of vSpWNet in OSI[10] model. The vSpWNet covers Datalink layer and Physical layer. The physical interface is SpaceWire buses and the Datalink layer mainly is SpaceWire driver framework.

By means of vSpWNet, we can use Ethernet in space. That way we can not only take advantage of SpaceWire's reliability, low power, and fault protection but also Ethernet's large amount of mature application software.

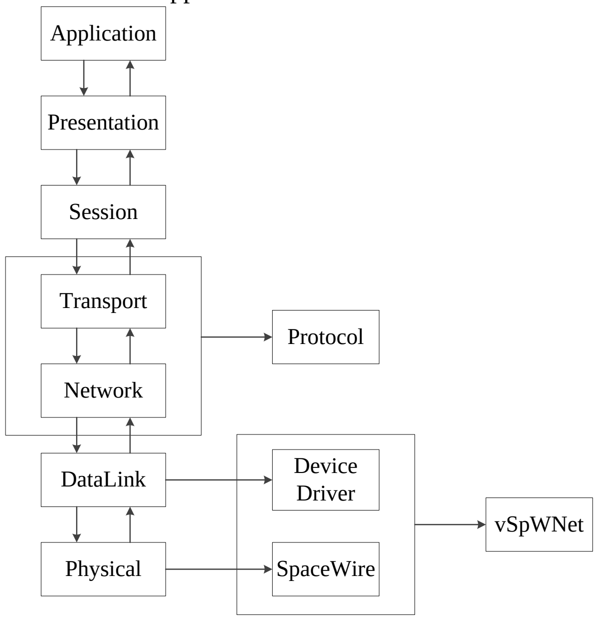


Fig. 1. vSpWNet's structure

So our main work is the design and implement of vSpWNet.

III. The Design and ImplementION of vSpWNet

In this section, we describe the design and implementation of vSpWNet.

First, we should decide the main structure of vSpWNet. Generally, the device driver and network layer connect directly as shown on Fig 2.

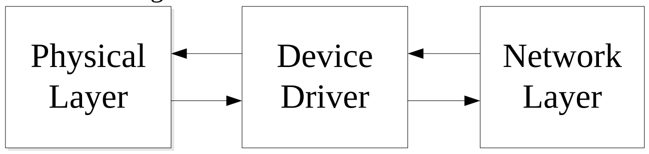


Fig. 2. Physical layer, device driver and network layer

This is the traditional structure. But it is hard to be ported into other operating system. So we add a Vsp layer between the device driver and network layer. The new structure is shown in Fig 3.

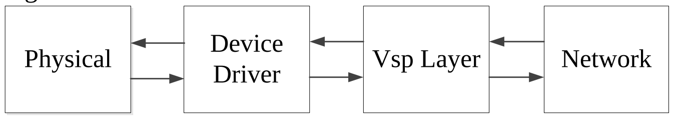


Fig. 3. the structure of vSpWNet

The Vsp layer contains the SpaceWire hardware access encapsulation functions, Network frame packaging and unpacking functions, packets encoding and decoding functions. The work models of Vsp layer is shown in Fig 4.

The SpaceWire buses receive the electric signal and convert it into data[4, 5]. Then the device driver commits the packages to the Vsp layer. Vsp layer dispose the packages and commit them to the network layer.

And the sending packet procedure is just the reverse.

Fig 5 shows the main function interfaces of the Vsp layer.

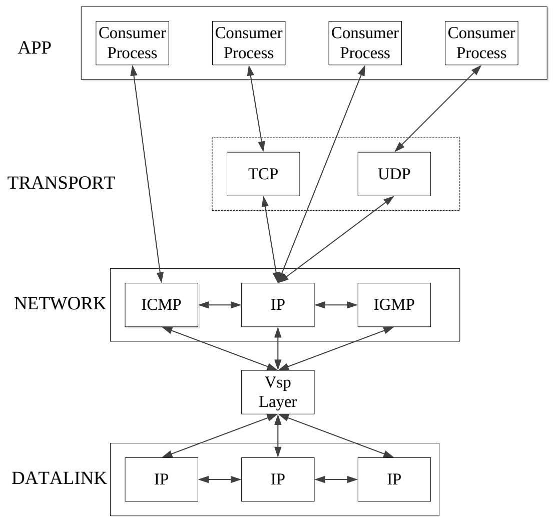


Fig. 4. The work models of Vsp

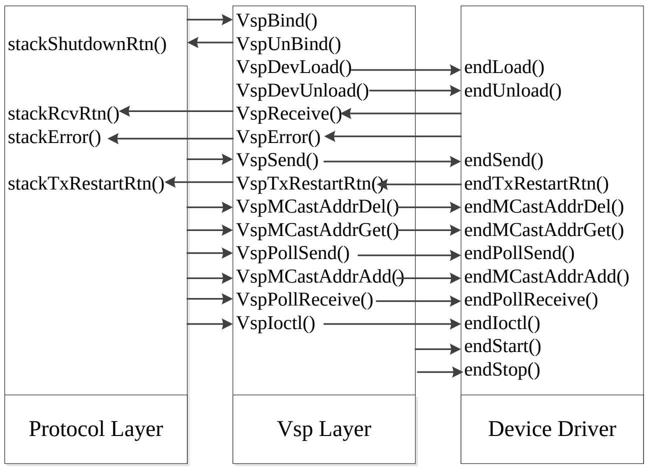


Fig. 5. the main function interfaces of the Vsp layer

There are 14 important function interfaces as follows:

- VspBind()：bind a network device;
- VspUnBind()：Unbind a network device;
- VspDevLoad()：Load a network device drive to the operation system;
- VspDevUnload()：Unload a network device from the operation system;
- VspReceive()：the receive data function;
- VspError()：return error to the upper layer;
- VspSend()：the send data function;
- VspTxRestartRtn()：restart the network device；
- VspMCastAddrDel()：delete the multicast address;
- VspMCastAddrGet()：get the multicast address;
- VspPollSend()：the polling send function;
- VspMCastAddrAdd()：add the multicast address;
- VspPollReceive()：the polling receive function;

- VspIoctl()：Device settings.

The Vsp layer separates the network layer and the datalink layer and improves the portability of vSpWNet.

IV. THE EXPERIMENT OF VSPWNET

We verify feasibility of vSpWNet by experiments and test the transmission speed and reliability.

The experiment is based on VxWorks 5.5[9] operation system. VxWorks is a real-time embedded operating system developed by WindRiver Company.

The MUX layer in VxWorks can be mapped to our Vsp layer. Fig 6 shows the position of MUX in OSI model.

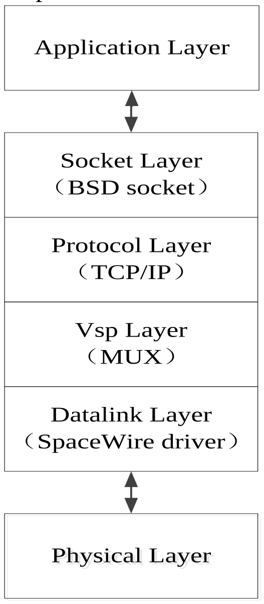


Fig. 6. The position of MUX

Having to MUX layer, we can develop the network device driver easily. Our SpaceWire driver is based on MUX layer and its important functions are listed as follows:

a) Device Loading Function: sysEtherEndLoad
Device Loading Function is the entrance of every network device drivers. Fig 7 shows the flow chart of sysEtherEndLoad:

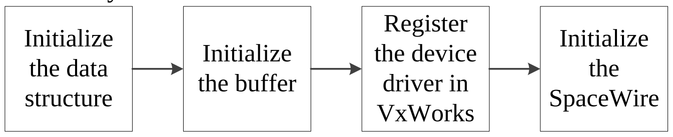

Initialize the data structure
Initialize the buffer
Register the device driver in VxWorks
Initialize the SpaceWire

Fig. 7. The flow chart of sysEtherEndLoad

b) Device Unloading Function: SPWUnload
SPWUnload releases all the data structure and unloading the device from VxWorks.

c) Device Start Function: SPWStart
SPWStart sets SpaceWire buses in work mode. It always calls function sysIntConnect to register interrupt in VxWorks. Fig 8 shows the flow chart of SPWStart:

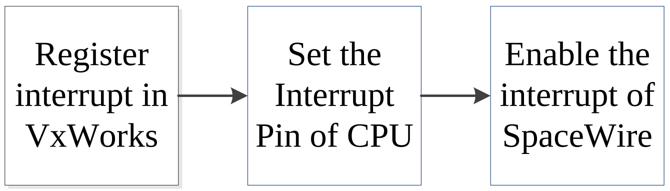

Register interrupt in VxWorks
Set the Interrupt Pin of CPU
Enable the interrupt of SpaceWire

Fig. 8. the flow chart of SPWStart

d) Sending Package Function: SPWSend
When there are packages to send, MUX will call SPWSend. Fig 8 shows the flow chart of SPWSend:

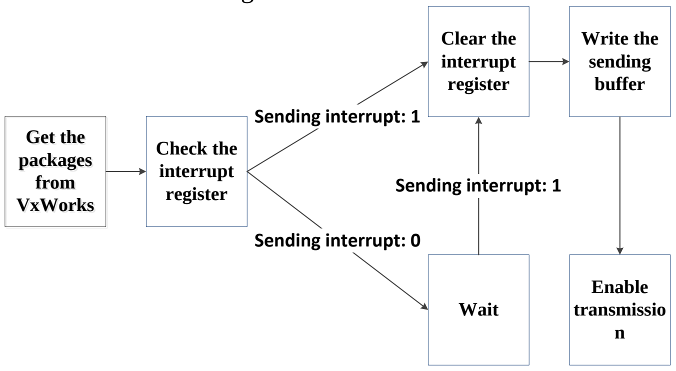


Fig. 9. The flow chart of SPWSend

e) Interrupt Handler Function: SPWInt
When there is an interrupt, MUX will call SPWInt to handle the interrupt. Fig 10 shows the flow chart of SPWInt:

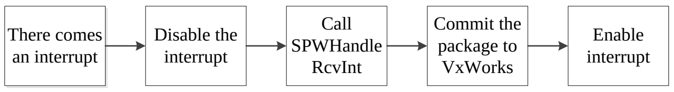


Fig. 10. The flow chart of SPWInt

f) Receiving Interrupt Handler Function: SPWHandleRcvInt()
This Function can handle the receiving interrupt. Fig 11 shows the flow chart of SPWHandleRcvInt:

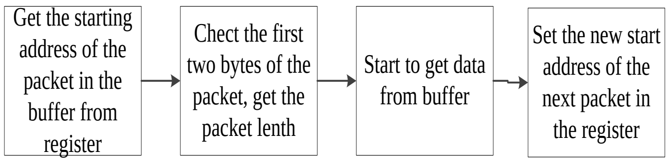

Get the starting address of the packet in the buffer from register
Chect the first two bytes of the packet, get the packet lenth
Start to get data from buffer
Set the new start address of the next packet in the register

Fig. 11. The flow chart of SPWHandleRcvInt

V. THE RESULT OF THE EXPERIMENT

Before testing, we should build the VxWorks Boot Image which integrates the vSpWNet driver. Then we burn it into the target board. The test scenario is shown in figure 12:

After the image loading successfully, we start to test the vSpWNet. The experiment shows that it works well. From the test results, we proved the feasibility of vSpWNet and it can meet most of the application transmission requirements.

VI. Conclusion and Future work

In this paper, we studied the advantages and disadvantages of SpaceWire buses systematically. Then we came up with the idea of vSpWNet and proposed the concept of Vsp layer to improve the portability of vSpWNet. Finally, we designed and have implemented it based on VxWorks embedded operation system.

From experiments results, it proved that our vSpWNet can meet the actual requirements properly. Most of the applications based on TCP/IP protocol work well on vSpWNet. So SpaceWire will be more easy to use. Moreover, our vSpWNet approach can be ported into other Operation Systems, including RTEMS, eCOS, etc.

In the future, we will improve our vSpWNet to make it work more efficiently.

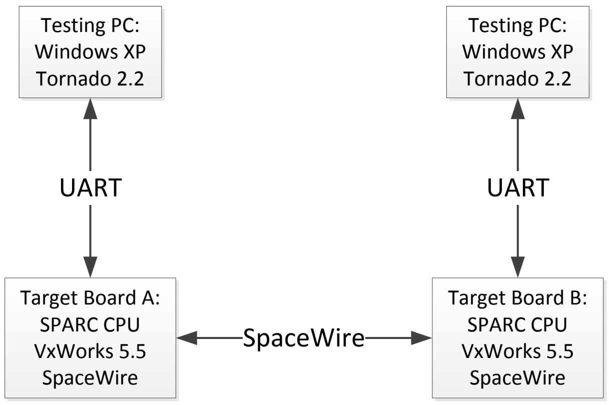

Testing PC: Windows XP Tornado 2.2
Testing PC: Windows XP Tornado 2.2
UART
UART
Target Board A: SPARC CPU VxWorks 5.5 SpaceWire
SpaceWire
Target Board B: SPARC CPU VxWorks 5.5 SpaceWire

Fig. 12. Test Scenario

ACKNOWLEDGEMENT

The work described in this paper is partially supported by the grants of the National Basic Research Program of China (973 project) under Grant No.2009CB320503, 2012CB315906; the project of National Science Foundation of China under grant No. 61070199, 61103189, 61103194, 61103182, 61202488, 61272482; the National High Technology Research and Development Program of China (863 Program) No. 2011AA01A103, 2012AA01A506, the Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20114307110006, 20124307120032, the program for Changjiang Scholars and Innovative Research Team in University (No.IRT1012), Science and Technology Innovative Research Team in Higher Educational Institutions of Hunan Province("network technology"); and Hunan Province Natural Science Foundation of China (11JJ7003).

REFERENCES

[1] http://en.m.wikipedia.org/wiki/SpaceWire.

[2] Glenn Rakow, Richard Schnurr, Steve Parkes, SpaceWire Protocol ID: What does It Means to You, Aerospace Conference, 2006. IEEE. 2006, 24(6):3~4.

[3] Sandra G. Dykes, Buddy Walls, Mark A. Jonhson, KristianPersson. A Non-Broadcast Address Resolution Protocol for SpaceWire Network. Aerospace Conference. Washington DC, ACM Press, 2006:4~5.

[4] TukkaHowkola, Sari Lppanen, Modeling the SpaceWire Architecture with Lyra. Proceedings of the Fifth International Conference on Application of Concurrency to System Design, Los Angeles, 2004. New York, IEEECOMPUTER SOCIETY, 2004:10~11.

[5] R. Marshall, W. Berger, C.Rodgers, Reconfigurable Processing Susysterms in Spaceborne Application. IEEE Aerospace Conference Proceedings, Big Sky, Montana, 2004. New York, IEEE CPOMPUTER SOCIETY, 2004:5~6.

[6] S.M.Parks, SpaceWire a satellite on board data handing network. Aircraft Engineering and Aerospace Technology, 2001.

[7] Dr SM Parkes, SpaceWire: SERIAL POINT-POINT LINKS, University of Dundee, 2000.

[8] WindRiver, VxWorks Network Programmers' Guide, 1999.

[9] Steven Vaughan-Nichols, The Birth and Rise of Ethernet: A History, 2012.

[10] Douglas E. Comer, Internetworking with TCP/IP-Principles, Protocols and Architecture, 2006.

[11] WindRiver, VxWorks Programmers' Guide, 1999.

Toolset for SpaceWire Networks Design and Configuration

Session: SpaceWire networks and protocols

Short Paper

Alexey Syschikov, Elena Suvorova, Yuriy Sheynin, Boris Sedov, Nadezhda Matveeva, Dmitry Raszhivin
Saint-Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russian Federation
{alexey.syschikov, suvorova, sheynin, boris.sedov, n.matveeva, dmitry.raszhivin}@guap.ru

***Abstract*— SpaceWire networks design and configuration is a complicated design problem that includes a set of different tasks to be solved: terminal nodes selection, workload tasks mapping, logical channels definition, interconnection topology design and switches selection, configuration space settings and routing table generation, etc. In design one should take into account many characteristics and requirements: terminal nodes performance for workload, channels latency and throughput, communication system capacity to cover logical channels requirements, etc. Every design step and the whole network should also fit into other user and technological constraints: area, mass, power consumption, heating also.**

Design tasks are complicated; some of them are NP-hard. “Manual” design of SpaceWire networks with dozens of nodes becomes very complicated, time-consuming and has high probability of design errors and misses.

To deal with the problem a through design flow for SpaceWire Networks Design and Configuration has been developed. Toolset includes network architecture design tool, communication structure forming tool and tool that forms logical structure and routing information. All these tools are highly automated and produce results corresponding to the defined user requirements and technological constraints. Network modeling tool is also provided for simulation of network functioning and collection of network statistics for investigation purposes.

***IndexTerms*— SpaceWire network, automated design, design space exploration, SpaceWire configuration**

I. Introduction

A network synthesis problem for systems with dozens and hundreds of nodes in general case is NP-hard. Nowadays in a number of network synthesis methodics affordable computation complexity is reached due to significant constraints in the problem statement only.

Some input parameters for SpaceWire network generation are determined at the architecture stage on the base of tasks parameters. Tasks should be allocated to nodes. The network should transmit data packets between source and destination terminal nodes with required throughput and timing parameters.

In system level design a network should be generated of devices (routers) from a system components library. The system components library can include different types of routers with various number of ports, mass, power, timing parameters.

The network should also meet user requirements in total equipment mass and power, in fault tolerance, considers specific requirements based on spatial placement of nodes.

Network synthesis methodic, such as based on Stainer three synthesis [1,2,3,4] and other classical methodics for wireless networks synthesis [1,5,6], do not deal with these constraints and can’t be used for SpaceWire networks synthesis.

Development of a SpaceWire network structure correspondingly to architecture and others user’s requirements are supported by the suggested methodology and developed tools.

II. Architecture

The system design techniques and supporting software are based on the design space exploration methodology. This approach is productive and is widely used in modern R&D. For example, communication architectures with big number of channels generate exponentially growing number of possible mappings of logical channels to the communication paths in the physical structure of the communication network. It makes impossible a straightforward exhaustive exploration of variants. Interrelation between possible mappings and communication protocols characteristics increases complexity of actions in design space exploration.

The proposed instrument is an automated tool for the design of SpaceWire networks.

The methodology of a network structure design includes four main stages:

- Analysis of the workload for the network to be design;
- Synthesis of the network architecture, including the terminal nodes and logical channels;
- Synthesis of the communication system, including switches and physical channels.
- Synthesis of the Logical Network structure and correspondent setting in nodes and switches.

Design of SpaceWire network architecture is performed from a given set of basic system components from a network system components library according to user-defined network characteristics.

During forming of the SpaceWire network architecture it is necessary to verify the possibility of allocation of computation workload to the particular library of system components. To achieve this, the allocation of computing workload to synthesized architecture should be done.

Formed structure should correspond to the user requirements: performance, throughput, command transmit delay, signal of real time transmit delay, failure tolerance of the network, etc.

The problem is solved with a restricted search. Methodology generates a set of possible solutions, which includes alternative problem solutions. Search is restricted by using a number of criteria for discarding not satisfying solutions at each stage.

The assignment problem for the network synthesis has a task graph G as the input data.

$$G = \{N, E\}, \quad \text{where}$$

N –set of graph vertexes, which are computation tasks,
E – set of graph edges.

Additionally we define the overall throughput of the network input/output ports in each direction (for input and for output).

$$\forall t \in Tt.IOth = \sum_{io \in t.IO} io.qty * io.th\ .$$

Task graph vertexes are mapped to the system library components and a set of possible solutions is formed. At this stage the job is to define the number of nodes of each type and allocate task graph vertexes to the SpaceWire network nodes (end nodes). The total network characteristics should not override the user-defined constraints.

The allocation of the task graph vertexes on the network nodes is represented by the

$$\mathrm{Allocation} = \{\mathrm{A}\}, \ \text{where}$$

$\forall \mathrm{i} \le \mathrm{k} \ \ \mathrm{a_i} \in \mathrm{A} = \mathrm{Allocation(i)} = \{\mathrm{n_i} \in \mathrm{N}, \mathrm{nb} \in \mathrm{Nb}\}$, where
k –total number of nodes in the task graph;
$\mathrm{n_i}$ –node of the task graph allocated to the node nb of the network.

The function for allocation of a task graph vertex to the network node is:

$$\mathrm{alloc(n_i)} = \mathrm{Allocation(i).nb}.$$

The function of a network node allocation that defines which task graph vertexes are allocate to this network node:

$$place(b) = \bigcup Allocation(n){:}\ Allocation(n).nb = b$$

Formula for the total weight of the SpaceWire network is:

$$Weight = \sum_{i=1}^{k} nb_i.t.w\,, \qquad where$$

k – the total number of nodes in the network;
$nb_i \in Nb$ – the$i$-the node of the network.

To limit the search space we apply the criteria:

1. The total weight of the SpaceWire network limit.

$$Weight \le W$$

2. The limit of the throughput of output logical channels of the network node.

The required output logical channels throughput is:

$$Througput_{nb_{i_{out}}} = \sum_{\substack{eb \in Eb: \\ eb.nb_{src} = nb_i \text{ и} \\ eb.nb_{dst} \ne nb_i}} eb.e.t\,,\ where$$

$nb \in Nb$ – node of the network.
The criterium is:

$$nb_i \in Nb, \qquad Througput_{nb_{i_{out}}} \le \sum_{n=1}^{k} nb_i.t.IOth$$

3. The limit of the throughput of input logical channels of the network node.

The required input logical channels throughput is:

$$Througput_{nb_{i_{in}}} = \sum_{\substack{eb \in Eb: \\ eb.nb_{dst} = nb_i \text{ и} \\ eb.nb_{src} \ne nb_i}} eb.e.t\,,\ where$$

$nb \in Nb$ – node of the SpaceWire network.
The criterium is:

$$\forall nb_i \in Nb, \qquad Througput_{nb_{i_{in}}} \le \sum_{n=1}^{k} nb_i.t.IOth$$

4. The limit of the requirement to the network node memory.

$$\forall nb_i \in Nb, \qquad \sum_{n \in place(nb_i)} n.m \le nb_i.t.m$$

5. The limit of the requirement to network node computation resource (processor).

$$\forall nb_i \in Nb, \qquad \sum_{n \in place(nb_i)} nb_i.t.P[n.f] \le 100$$

In the process of possible solutions building the best ones are selected with the complex set of minimization parameters.

1. Total requirement for the network throughput.

$$Min_1 = \sum_{\substack{eb \in Eb: \\ eb.nb_{src} \ne eb.nb_{dst}}} eb.e.t$$

2. Maximum requirements for the throughput of network node input ports.

$$Min_2 = \max_{nb_i \in Nb}\left(Througput_{nb_{i_{in}}}\right)$$

3. Maximum requirements for the throughput of network node output ports.

$$Min_3 = \max_{nb_i \in Nb}\left(Througput_{nb_{i_{out}}}\right)$$

As a result we get the architecture of a SpaceWire network that is based on the computing/communication workload requirements and satisfy the user-defined constraints.

III. SpaceWire network synthesis

We use decomposition of a system to subsytems. Decomposition is used for support of spatial constraints and specific timing constraints (jitter) for some tasks. It helps also to decrease algorithm complexity.

A. Spatial constrains

Technology constraints related to spatial placement of terminal nodes and specific of cable-laying typically are relevant for a spacecraft. Certain groups of devices should be placed locally due to their functionality (sensors, locator) or structural reasons. Typically quantity of cables that connect such group with other parts of the network is strongly constrained.

The methodic takes into account this type of user constraints. User can specify such groups of nodes as clusters. Our tools generate subnetwork structure for each cluster independently (routers of one subnetworks are not used in others) and with constrained (required) quantity of interconnections to other network parts.

B. Specific timing constraints

We use network structure patterns in our methodic for specific timing constraints support.

SpaceWire networks are often used for data transmission from sensors to computer or from computer to visualization or telecommunication subsystems. Not only packet delivery time but jitter is important for these applications also. Therefore an appropriate network structure for such subsystems is symmetric (that is important for jitter parameter) tree.

The tool generates symmetric trees of routers from the system components library that meet mass and power constraints. Adaptive routing can be used for simultaneous throughput utilization of some links, which directly connects two devices (e.g. in fat trees).

Such subsystems can include up to 90% of terminal nodes. Therefore this approach also allows to essentially decreasing the computation complexity of the algorithm.

At a lower layer of the tree our tool could generate daisy chains of nodes (if nodes supports this functionality), to decrease hardware cost of the network.

The tool automatically selects subgraphs in a logical interconnection graph, for which tree based subnetworks could be generated.

Others typical for a SpaceWire networks structures are used for distributed computing. In the tasks graph such structures are usually represented by subgraphs with peer to peer connections between tasks. Typical requirement is equal data transmission time between all components of a distributed computing platform.

A designer should select subgraphs that correspond to distributed computing in a logical interconnections graph because rules of such subgraphs detection strongly depends on the tasks that are processed in system.

A good structure for such network fragments corresponds to bipartite graphs is Banyan network [7,8,9]. Banyan networks provide path with equal length between all nodes from one set of nodes from a bipartite set to another set.

In our methodic Irregular Banyan subnetworks are built of routers. The Banyan subnetwork generation algorithm can use the adaptive routing to utilise total throughput of some links that directly connect a pair of devices (nodes, routers) and takes into account requirement of connections with other parts of the SpaceWire network.

Subnetworks for logical interconnections subgraphs with peer-to-peer interconnections are generated as Banyan networks also.

A Banyan network in this case should connect not only nodes from different sets but nodes from one set with same timing parameters of interconnections also. We use coupled Banyan networks in what in the left half of the network we append interconnections mirrored interconnections of the right half; in the right half we append interconnections mirrored interconnections of the left half.

Example of such a network is represented in Fig.1. Black lines correspond to interconnections between nodes from different sets, gray lines correspond to interconnections between nodes from one set.

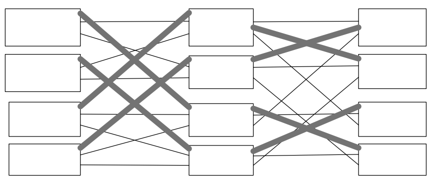


Fig.1 Example of doubled banyan network

IV. Synthesis of Logical Network structure

Synthesis of a logical network structure includes mapping of logical channels (created at the architecture synthesis stage) to physical paths (in the basic variant of network structure) and generation of configuration settings for all terminal nodes and routers in the network structure.

In the developed algorithm, which maps logical channels to physical paths, logical channels could be mapped not only to shortest paths, by to others paths that correspond to throughput and timing constraints. Usage of the adaptive routing for throughput is also considered.

Further generation of configuration parameters for terminal nodes and routers (routing tables' content, adaptive routing mode, and link transmission rate) is performed.

Link transmission rate is configured correspondingly to required throughput and packet transmission time. Adaptive routing could be configured for pairs of nodes or routers that are directly connected via some links (for throughput).

Logical addresses of the terminal nodes should be defined before the routing table content generation. Our tools assign a logical address to every application (task) in a terminal node.

If the quantity of addresses is more than 224, then the regional addressing is used. Regions (or groups of regions) correspond to subgraphs that are extracted in the logical interconnection graph structure.

Routing tables' content is generated after logical and regional address assignment, in correspondence to mapping of logical channels to physical paths.

V. Fault tolerance support

Hardware redundancy is used for providing fault tolerance.
Fault tolerance support is realized by to hardware replication (routers and links replication). We suggest two types of redundancy policy: path replication based and dynamic path reconfiguration.

A. Path replication based redundancy

If tolerance to N-1 faults is need, the whole SpaceWire network structure should include N independent paths between source and destination terminal nodes. The whole network structure (Fig.2) includes N copies of the basic structure (configurations of corresponding routers in all replicas are identical). Every terminal node is connected to all replicas of the basic network structure. Source terminal nodes should send copies of every packet to all replicas of the basic structure. Destination terminal nodes should correctly interpret data flows with proper and faulty copies of received packets.

This redundancy policy is recommended for systems with strong requirements to packet delivery reliability and real time constraints.

Hardware cost of buffering scheme in terminal nodes is essential for this redundancy policy.

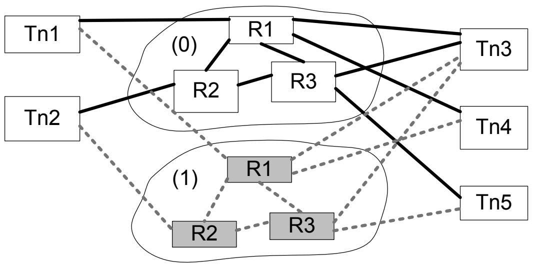


Fig.2 Example of a path replication based network structure (N=2)

If fault tolerance is required, at the first stage our tools generate a basic network structure and logical configuration for it and next this network structure is expanded for fault tolerance.

User should reserve resources (mass, power, number of device's used ports that can be utilized for a basic network structure, should be decreased correspondingly to quantity of faults and selected fault tolerance policy) before a basic network generation. In the basic network should be used not more than 1/N allowable mass and not more than 1/N of every terminal node port's quantity for tolerance to N-1 faults.

For practical reasons it is typical to apply FT-requirements to some fragments of the network only, to some of its subnetworks, clusters. Thus the strategy of network redundancy by path replication is applied to these individual parts of the network.

B. Dynamic path reconfiguration redundancy

Further, dynamic path reconfiguration redundancy could be used for a N-1 faults tolerant network that includes N replicas of a basic network (Fig.3). All routers Ri(0) and Rj(0) in the basic network (marked by "0") that are directly connected, have connections with routers Rj(k) and Ri(k) for all network replicas (represented by gray dotted lines in the figure).

Adaptive routing configuration for direct interconnections of devices by some links is identical in the basic network and all the replicas.

For dynamic path reconfiguration additional adaptive routing configuration is generated for interconnections between network replicas in whole network structure.

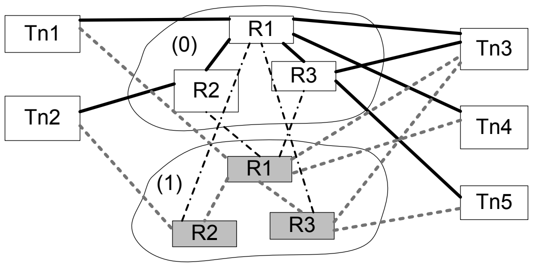


Fig.3 Example of dynamic path reconfiguration based network (N=2)

In this case a terminal node sends to the network only one copy of a packet. It sends packet to any link that corresponds to this packet's path and is currently in the work state. Then every next transit router send packet to any link that corresponds to this packet's path and is ready. If a fault occurs in a router or a link when a packet transmitted, this packet would be lost or corrupted and destination node would not receive correct copy of it. Also fault can impact to transmission time of others packets in network.

However in this case designer doesn't need additional hardware for packet buffering in terminal nodes. Therefore this fault tolerance policy is recommended for networks without guaranteed delivery packets, without strict real time requirements and when packets buffering in terminal nodes is impossible.

When a designer plans to use this fault tolerance policy, he should reserve resource of network equipment mass and resource of terminal nodes and routers port's quantity before a basic network generation. In the basic network should be used not more than 1/N allowable mass and not more than 1/N of every router and terminal node port's quantity for tolerance to N-1 faults.

VI. Conclusion

The paper describes the methodology and toolset for SpaceWire network design. It provides the design space exploration mechanism for synthesis of large SpaceWire networks. The design includes the set of terminal nodes, communication structure, switches and links to meet the computation requirements and user-defined constraints. The high level of automation allows making changes in requirements and reconfiguration of SpaceWire network rapid and easy. The methodology also allows generating fault tolerant networks with variable level of tolerance.

REFERENCES

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, CliffordStein, Introduction to algorithms, Second Edition, Massachusetts Institute of Technology, 2009.

[2] Hazewinkel, M. (2001), "Steinertreeproblem", inHazewinkel, Michiel, EncyclopediaofMathematics, Springer, ISBN978-1-55608-010-4

[3] Algebraic Graph Theory" byChrisGodsilandGordonRoyle, publishedbySpringer-Verlag, 2001, 439 pp

[4] The Steiner three problem: a tour through graphs, algorithms, and complexity Hans Jürgen Prömel, Angelika Steger. Amer Mathematical Society, 2002 241 pp.

[5] Wu, BangYe; Chao, Kun-Mao (2004). Spanning Trees and Optimization Problems. CRC Press. ISBN1-58488-436-3

[6] A Walk ThroughCombinatorics: An Introduction to Enumeration And Graph Theory. MiklósBóna. World Scientific, 2006 – 469 pp.

[7] Principles and Practices of Interconnection Networks William James Dally, Morgan Kaufmann Publishers is an imprint of Elsevier 2004 581 pp

[8] Computer NetworksV.S.Bagad, I.A.Dhotre, Technical Publications, 2009 г. – 512 pp.

[9] Data And Computer CommunicationsюWilliam Stallings. Pearson Education, 2007.852

Delay Guarantee for Real-time Message in SpaceWire-D Network

Session: Networks and Protocols, Short Paper

Qiang ZHOU, Chunming ZHANG
School of Electronics Information Engineering
BeiHang University
Beijing, China
zhouqiang_ee@buaa.edu.cn, 446022801@qq.com

Hengqing LIN
System Engineering Research Institute
China State Shipbuilding Corporation
Beijing, China
wu342@163.com

***Abstract*—SpaceWire is a standard for on-board satellite networks as the basis for future data-handling architectures. However, it can't meet the deterministic requirement for safety/time critical application in spacecraft, where the delay of real-time (RT) message streams must be guaranteed. Therefore, SpaceWire-D is developed that provides deterministic delivery over a SpaceWire network. This paper researches on computer simulation for real-time performance in SpaceWire-D network. Based on the principal of SpaceWire-D protocol, we set up its network model and simulation model where an event-driven scheme is adopted, give an implementation of simulation program based on VC, where the Time Slot scheme and static schedule method are also implemented, and then present a validation of the simulation model, and finally develop several simulation cases on typical application. The simulative results indicate that the schedule table plays an important role on delay of the message in SpaceWire-D networks. To increase the real-time performance of SpaceWire-D, an effective schedule table should be researched and provided.**

***Index Terms*—SpaceWire-D, real-time, simulation, VC.**

I. Introduction

SpaceWire is a standard for on-board satellite networks chosen by the ESA as the basis for future data-handling architectures [1]. However, it can't meet the deterministic requirement for safety/time critical application in spacecraft, where the delay of real-time (RT) message streams must be guaranteed. Therefore, SpaceWire-D is developed that provides deterministic delivery over a SpaceWire network [2].

Aiming to evaluate the SpaceWire/SpaceWire-D protocol, network designers may take several measures such as numerical analysis [3], simulation, and hardware test. An analytical method in [3] to compute an upper-bound on the worst-case end-to-end delay of a packet in a SpaceWire network is proposed. However, the analytical method is limited by many assumptions. Also, hardware test is sometimes impractical because of long research cycle and expensive cost.

This paper researches on computer simulation for real-time performance in SpaceWire-D network. Based on the principal of SpaceWire-D protocol, we set up its network model and simulation model where an event-driven scheme is adopted, give an implementation of simulation program based on Visual C++, where the Time Slot scheme [4] and static schedule method are also implemented, and then present a validation of the simulation model, and finally develop several simulation cases on typical application.

II. Overview of the SpaceWire-D

A. *SpaceWire-D Protocol*

SpaceWire provides versatile network architecture for onboard data-handling using switches and bi-directional serial links. It delivers the high throughput required for payload data with low implementation cost. However, it does not provide guarantees in the packet latency due to network congestion. Besides, the use of wormhole switching increases the worst case latency of packets that use shared links on the way to their destination.

SpaceWire-D is a protocol that provides deterministic delivery over a SpaceWire network [4] (the protocol stack for SpaceWire-D is illustrated in Fig.1). SpaceWire-D delivers data within predetermined time constraints. Deterministic data delivery is the delivery of data within predetermined time constraints: not too early and not too late.

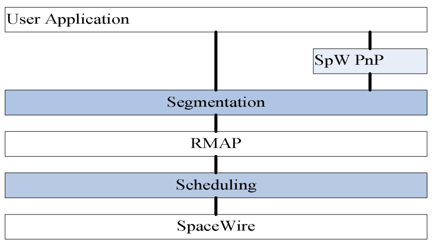


Fig.1. SpaceWire-D protocol stack

B. *SpaceWire-D Scheduling*

In SpaceWire-D networks, it is often required that data is delivered within certain time constraints. One promising solution is to schedule the network using time division multiplexing. With a *schedule table*, there is no network contention and packet delivery time is deterministic. Then it is possible to obtain latency and throughput guarantees for the user data. The required periodic synchronization signal is easily provided using SpaceWire *Time-Code (TC)* characters. Time is divided into discrete time intervals or *time-slots (TS)* determined by the arrival of a Time-Code.

SpaceWire uses wormhole switching, so packets are typically not buffered within the routers. Therefore, the scheduling is implemented at each transmitting node or network terminal using a local schedule table. Each local table must be configured following a global network scheduling, assuring that contention can not occur when no errors are present in the network.

III. SIMULATION MODELLING

Aiming to evaluate the SpaceWire/SpaceWire-D protocol, a simulation method is adopted. And simulation modeling of the SpaceWire networks covers some issues including event-driven mechanism, time-code synchronization, and etc.

A. Event-driven Mechanism

There are two time advance approaches in simulation [5]: a) time-driven simulator, and b) event-driven simulator. In this paper, the event-driven approach is adopted because of its effective simulation performance. Figure 2 presents the flow diagram of event-driven simulation.

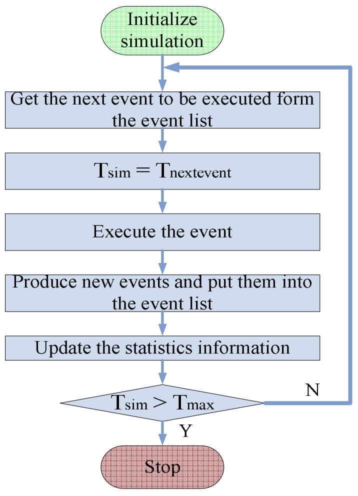

Initialize simulation
Get the next event to be executed form the event list
$T_{sim} = T_{nextevent}$
Execute the event
Produce new events and put them into the event list
Update the statistics information
$T_{sim} > T_{max}$
N
Y
Stop

Fig.2. A Typical event-based simulation model

Where,

Initialize simulation is that some node information and events are generated when the simulation is started, and an event list is built.

Get the next event to be executed from the list is that those events are arranged chronologically in the event list, and the simulation program gets the next event which should be executed from the list.

Tsim= Tnext-event is that once the next-event is got, the event-driven simulator progresses time with the next-event time.

Execute the event is that the event is triggered and executed step by step.

Produce new events and put them into the list is that the executed event may triggers new events. If there are some new events triggered, they will be inserted into the event list.

Update the statistics information is that when an event is executed, there may lead to some changes of states and parameters in the nodes or routers. So the changes should be updated.

Tsim>Tmax is that the simulation program would judge if the current time exceeds the simulation time we set. If no, the simulation will go on, or else, the simulation will finish.

B. Event Example

The implementation process of the simulation program is described by corresponding events. Figure 3 proposes an example of event, named as Time-Code generation event. In the SpaceWire-D network, there is a unique node that can allocate system time to other nodes to ensure synchronization. It is named as Time-Master. Time-Master periodically triggered the Time-Code generation event. The flowchart of this event is shown in fig.3. This event is triggered periodically to generate Time-Code and insert the Time-Code into the output queue. While sending the Time-Code, the corresponding router event is triggered. In this way, all the information transferred in the SpaceWire-D network can be scheduled.

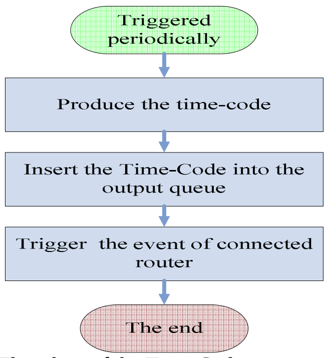

Triggered periodically
Produce the time-code
Insert the Time-Code into the output queue
Trigger the event of connected router
The end

Fig.3. Flowchart of the Time-Code generation event

C. SpaceWire-D Simulation Based on Event-driven Scheme

Figure 4 presents the working process of the SpaceWire-D

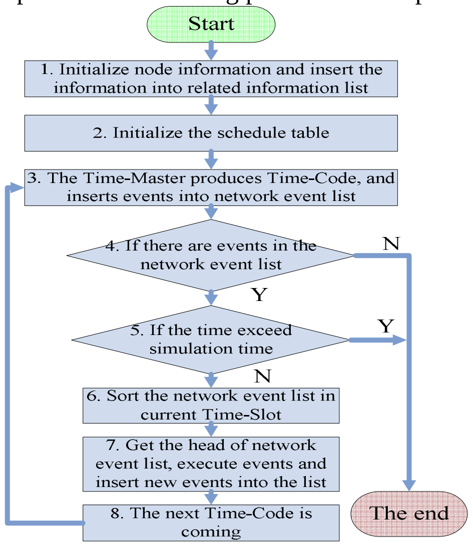

Start
1. Initialize node information and insert the information into related information list
2. Initialize the schedule table
3. The Time-Master produces Time-Code, and inserts events into network event list
4. If there are events in the network event list
N
Y
5. If the time exceed simulation time
Y
N
6. Sort the network event list in current Time-Slot
7. Get the head of network event list, execute events and insert new events into the list
8. The next Time-Code is coming
The end

Fig.4. Flowchart of SpaceWire simulation model based on event-driven scheme

network model based on event-driven scheme. As described in section II, the Time-Code and schedule are two key issues in the SpaceWire-D network. Therefore, the simulation model needs to provide event scheduling scheme based on Time-Code. In fig.4, Step 2 indicates that scheduling of every event can be implemented only if the event is in its Time Slots.

D. the Simulation Software Based on VC + +

On the basis of the simulation modeling, a simulation program based on VC++ [6] is developed to research the performance of SpaceWire-D network. And the simulation software includes several modules such as *message parameter setting module*, *topology configuration module, simulation control module, and data survey module*.

IV. Verification

In this section, a mathematical model is proposed to analyze and verify the effectiveness of the aforementioned simulation model.

A. SpaceWire Network Topology

The SpaceWire Network is setup according to the topology described in fig.5. The network topology consists of seven nodes, two routers and several SpaceWire links. Where, LA41, LA52~ LA54, LA60, LA70, and LA80 are working as nodes. Among them, LA70 is a mass storage, the destination of LA41, LA52 ~ LA54, LA60 and LA80. As a processor, LA80 is used to transmit the packets coming from LA54 to LA70 for storage. LA60 collects packets from sensors, and then sends them to LA70 mass storage [4].

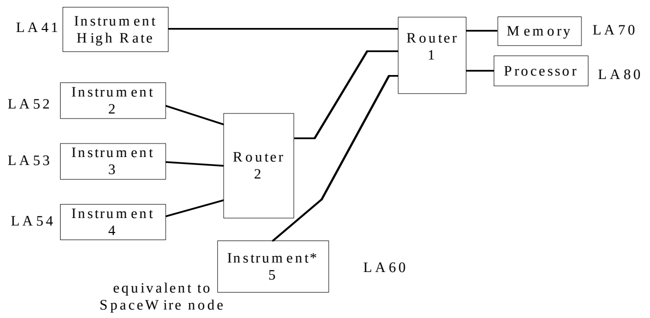


Fig.5. SpaceWire-D network topology architecture

B. the Scenario Parameters

The scenario parameters and the events are set in Table I and Table II, respectively.

TABLE I. The Scenario Parameters

Quantity	Value
Simulation time	100s
Link bandwidth	200Mbps
Scheduling delay of router	0.5us

TABLE II. EVENTS AND THEIR TYPES

Number	*Event*	Event Type
M1	LA41→LA70	Write
M2	LA60→LA70	Read
M3	LA80→LA70	Write
M4	LA52→LA70	Read
M5	LA53→LA70	Read

C. Model Verification

Ref. [4] proposes a mathematical model to compute the event delay in a SpaceWire-D network. Ordinarily, an event contains two processes, one is the process of source node sending message and the process of destination node sending response message. So the event delay is the sum of message delay and response message delay, i.e., the delay is the time to finish both processes.

The delay of SpaceWire-D is denoted as $D_{i,m}$

$$D_{i,m}=\begin{cases}\left(P-(T_{im}\backslash C)\backslash P\right)+\left(N-1-\frac{T\backslash C}{P}\right)*P+(i-1)*P+D_e & \frac{T\backslash C}{P}\geq i-1, P-(T\backslash C)\backslash P<D_e \\ D_e & \frac{T\backslash C}{P}=i-1, P-(T\backslash C)\backslash P\geq D_e \\ \left(i-2-\frac{T\backslash C}{P}\right)*P+P-(T\backslash C)\backslash P+D_e & \frac{T\backslash C}{P}<i-1\end{cases} \tag{1}$$

Where,

N	is the number of nodes in the SpaceWire network.
i	is the sequence number of each node.
T_{im}	is the generation time of the *m*-th message in the *i*-th node.
D_e	is the operation delay, which represents the delay of event such as READ and WRITE defined in [4].
\	is an operator getting remainder. The remainder is number left over when one integer is divided by another.

D. Results

Table III shows the deviation between the simulation results and mathematical analysis results.

TABLE III. Comparison on ETE Delay of Different Events

Events	Maximum delay			Minimum delay		
	exp.(1) (us)	sim. (us)	deviation (%)	exp.(1) (us)	sim. (us)	deviation (%)
M1	188.450	188.548	0.052	38.450	38.452	0.005
M2	233.650	233.748	0.042	83.650	83.748	0.117
M3	203.450	203.549	0.049	53.450	53.549	0.185
M4	248.650	249.750	0.442	98.650	99.750	1.115
M5	218.450	219.550	0.504	68.450	69.550	1.607

Table III illustrates that the simulation results are basically consistent with mathematical analysis results, which indicates the correctness of the simulation model. Moreover, the deviations of delay between two methods are less than 2%, which verify the validity of the simulation model.

V. Simulation And Analysis

In the simulation program, the delays of different events are discussed when the schedule is set to *Simple Schedule*, *Concurrent Schedule* and *Optimization Concurrent Schedule*, respectively.

A. Simulation Scenario & Message Parameters

Here, we use the aforementioned network topology and the events in Sec.IV. The scenario parameters are set in table IV.

TABLE IV. THE SCENARIO PARAMETERS

Quantity	Value
Simulation time	100s
Time-slot	45us
Time code	0~63

i) Simple Schedule (SS)

SS gives an initiator full control of the network for one or more specified time-slots, which is illustrated in Table V.

TABLE V. SIMPLE SCHEDULE

Time-Slot	0	1	2	3	4	···	58	59
Event	LA41	LA60	LA80	LA52	LA53	···	LA52	LA53

ii) Concurrent Schedule (CS)

CS makes more usage of network bandwidth by allowing more than one initiator to initiate transactions in a time-slot. This gives much possibility that two initiators might attempt to use the same network resources (links) at the same time. So in CS, LA52 can share the time slot with LA80, which is illustrated in Table VI.

TABLE VI. CONCURRENT SCHEDULE

Time-Slot	0	1	2	3	4	···	58	59
Event	LA41	LA60	**LA80 LA52**	**LA52 LA80**	LA53	···	LA52 LA80	LA53

iii) Optimization Concurrent Schedule (OCS)

OCS builds on the concurrent schedule to improve network efficiency further. Different from CS in table VI, where the time schedule of LA80 and LA52 take two TCs, OCS makes the two TCs become one, so the distribution period will cut down by a time slot.

TABLE VII. OPTIMIZATION CONCURRENT SCHEDULE

Time-Slot	0	1	2	3	4	···	58	59
Event	LA41	LA60	**LA80 LA52**	LA53	LA41	···	LA53	LA41

B. *Simulation*

Figure 6 presents the delays of five events when the scheme is set to SS, CS and OCS, respectively.

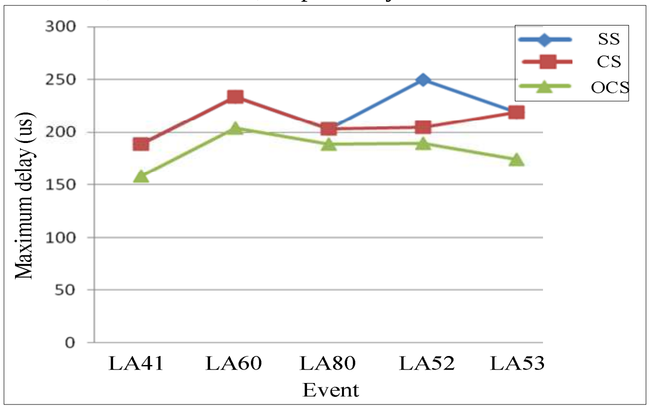


Fig.6. Delay comparison among three schedules

Figure 6 shows that, i) For SS, the delays of five events (190, 230, 200, 250, and 220 (unit: us)) are all higher than those of the other two schedules. ii) For CS, the delay of event LA52 is smaller than that of SS, i.e., the delay is decreased from 250us to 200us. The reason is that LA52 and LA80 share the same time-slot. The two nodes can use the same links at the same time. iii) For OCS, the delays of five events are the smallest among the three schedules. The reason is that the schedule period decrease from 5 to 4, so the delays of all events decreased.

C. *Result*

Based on the above simulation, we can infer the following result that the schedule table plays an important role on delay of the message in SpaceWire-D networks. To increase the real-time performance of SpaceWire-D, an effective schedule table should be researched and provided.

VI. CONCLUSION

In this paper, we set up SpaceWire-D network model and simulation model where an event-driven scheme is adopted, give an implementation of simulation program based on VC++, where the times slot scheme and static schedule method are also implemented, and then present a validation of the simulation model, and finally develop several simulation cases on typical application. The simulative results indicate that an effective schedule table can obtain better real-time performance of SpaceWire-D networks.

ACKONWLEDGMENT

This work was supported by Beijing Natural Science Foundation (4133089).

REFERENCES

[1] ECSS-E-50-12-C. SpaceWire Engineering: SpaceWire-Links, node, routers and networks ESA-ESTEC. November 2008.

[2] S. Parkes and A. Ferrer-Florit, “SpaceWire-D Deterministic Control and Data Delivery Over SpaceWire Networks”, ESA Contract No. 220774-07-NL/LvH, University of Dundee, April 2010.

[3] T. FERRANDIZ, F. FRANCES, C. FRABOUL. A method of computation for worst-case delay analysis on SpaceWire networks [J]. Institute of Electrical and Electronics Engineers (IEEE), SIES '09; 8 - 10 July 2009, Ecole Polytechnique Federale de Lausanne, Switzerland. IEEE, Piscataway, pp.19-27.

[4] SpaceNet-SpaceWire-RT Initial Protocol Definition. Space Technology Centre School of Computing University of Dundee, DD1 4HN Scotland, UK. October 2008.

[5] Euiyul Ko, Hanjin Park, and Ikjun Yeom. A New Event-Driven Network Simulator for Delay-Tolerant Networks(DTNs), Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, Brussels, Belgium. page 59-64, (2010)

[6] Microsoft Development Network-MSDN Library Visual Studio 6.0 Release. Microsoft Company,2000 April.

SpaceFibre Quality of Service Features Support in the Network Level

SpaceWire Networks and Protocols, Short Paper

Nadezhda Matveeva, Elena Suvorova, Valentin Olenev, Irina Lavrovskaya, Ilya Korobkov, Artur Eganyan
Saint-Petersburg State University of Aerospace Instrumentation
SUAI
Saint-Petersburg, Russian Federation
n.matveeva88@gmail.com, wildcat15@yandex.ru, Valentin.Olenev@guap.ru, Irina.Lavrovskaya@guap.ru, Ilya.Korobkov@guap.ru, eganyan.artur@gmail.com

The novel SpaceFibre standard provides quality of service features, which are implemented by means of virtual channels. However, the SpaceFibre standard covers only "point-to-point" connections and does not specify the network layer.

In order to provide quality of service in a network operating with SpaceFibre links it is necessary to support quality of service mechanisms in the routing switch functionality.

Nowadays, routing switches can support various implementations of virtual channels mechanisms (architectures and structures of port controllers and switch matrix). These implementations provide different performance and latency characteristics for packet flows. But at the same time, they lead to different hardware costs.

One of the key aspects affecting both achievable performance and latency characteristics and hardware costs is ratio of the virtual channels quantity in a port and switch matrix's channels quantity connected to every port (connection point). The connection points quantity can vary from one to a quantity of virtual channels in this port.

The case when switch matrix's channel quantity connected to one port is less than quantity of virtual channels in this port can lead to additional packet transmission latency which is necessary for release of a connection point. Moreover, with growing quantity of connection points the hardware cost of switch matrix and port controllers increases dramatically.

In this paper we evaluate additional data packet transmission latency introduced by a switch matrix and hardware costs for different routing switch parameters: ports quantity, virtual channels quantity in ports, connection points quantity; and different parameters of packet flows through each virtual channel.

***Index Terms*—SpaceFibre, SpaceWire, Networking, Virtual channels, Quality of Service.**

I. Introduction

SpaceWire-RT Network layer is responsible for routing SpaceFibre packets over a SpaceFibre network, comprising SpaceFibre routing switches, SpaceFibre links, and SpaceFibre nodes. SpaceWire-RT provides compatibility with SpaceWire at Network and Packet Layers [1]. The current specification of the Packet and the Network layers is not included in the SpaceWire-RT standard. But the Network layer implementation is described in the SpaceFibre presentation [5].

Quality of service parameters that can be provided by routers with SpaceFibre ports [1] depend not only on the SpaceFibre protocol characteristics and port implementation specific but on a network layer implementation also.

In this paper we consider some classical ways of the network layer implementation that can be used for a router with SpaceFibre ports. We evaluate the hardware costs and timing characteristics for these implementations.

We do not consider an impact of low layers of SpaceFibre (from the Retry Layer and lower) to timing characteristics of data flows. In our analysis we suppose that SpaceFibre connection is established and data transmission errors do not occur (data frames are not retransmitted). We take into account the necessity of auxiliary information's transmission by reserving of some physical channel throughput.

Let us consider which parameters are important for different qualities of service (QoS) that are provided by SpaceFibre.

For the Scheduled service a packet should be transmitted from the source to the destination during a corresponding time slot (the packet can be transmitted through one or several routers).

Thus the short transmission time from the input port to the output port via the network (with only one router) is very important for the scheduling service.

Therefore, for evaluation of achievable parameters for this type of service we need to know the maximal packet transmission time via the network layer.

For the Bandwidth reserved service the network layer should support the corresponding throughput for its channels. For this type of service the following additional requirement to a jitter size [2] often exists: the difference between minimal and maximal packet transmission time via the network (with only one router) should not exceed a required value. Consequently, it is necessary to know the minimal and maximal packet transmission time through the network.

If the Priority service is used, the network layer should provide a priority processing scheme without priority inversion. Priority inversion could appear only when data packets with different priorities are transmitted via one network layer resource (one channel). In this case the packet with the higher priority can wait until a lower priority packet would be transmitted if the lowest priority packet goes first.

II. *Considered router's structures*

A. 1st way of router's network layer structure

Router's Switch matrix includes a separate channel for connection of each input virtual channel with the correspondent output virtual channel in this way. Quantity of connection points to the switch matrix (hereinafter – connection points) for every port of a router is equal to the virtual channels amount in this port, *Fig. 1* (only one data transmission direction is represented). This way was recommended by the SpaceWire-RT specification draft [5].

In such router structure data flows can compete with each other only within one virtual channel in output port of router. In this case timing characteristics in the network layer depend only on arbitration rules. In all other cases timing characteristics of data flows are not influenced by the router network layer.

However, such router structure results in an essential hardware cost.

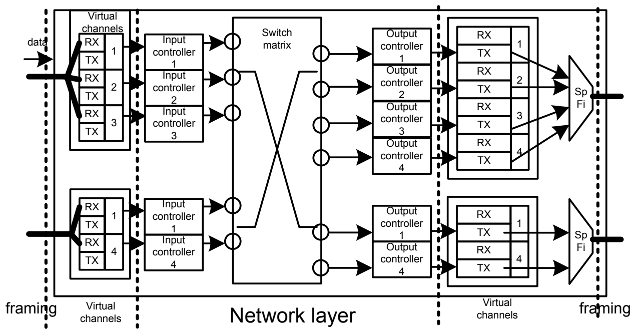


Fig. 1 The first way of router's network layer implementation

B. 2nd way of router's network layer structure

According to this router structure, the quantity of connection points for every port is less than amount of virtual channels in the port. In our research we suppose that data flows from every virtual channel can be transmitted via any connection point of the correspondent port, *Fig. 2*.

Hardware cost of this router structure is essentially less, than hardware cost of the previous one. But in this way, data flows from different virtual channels share switch matrix channels. Therefore, an impact between data flows and corresponding disturbance of its timing characteristics in this case in this router structure is more essential than in the previous one.

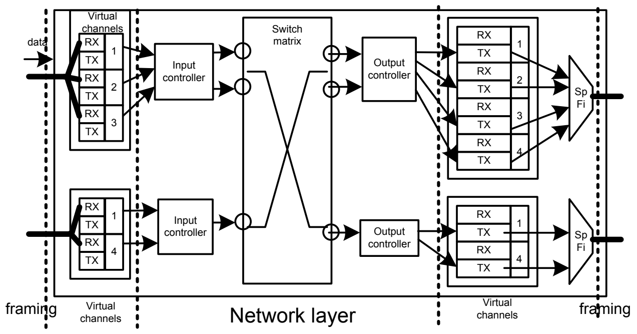


Fig. 2 The second way of router's network layer implementation

III. Evaluation of hardware Costs

We are using Cadence RTL Compiler and Encounter and UMC 120 nm technology library for evaluation of router's switch matrix hardware cost.

We performed a logical and a physical synthesis of the switch matrixes with different number of channels that correspond to different router implementations (different amount of ports and connection points).

Results of the logical synthesis are represented in *Fig. 3*. As shown in this figure, if quantity of connection points is bigger than 4, hardware cost grows essentially. The logical synthesis becomes impossible when quantity of ports is 16 and quantity of virtual channels is 16 or bigger (256 channels of the switch matrix). The physical synthesis is problematic if quantity of ports is bigger than 8 and of virtual channels is bigger than 8 (64 channels of switch matrix). This amount of switch matrix channels is boundary of hardware resources for the 1st way of a router structure. The 2nd way can be implemented with the greater amount of virtual channels if 2 – 4 connection point for every port is used.

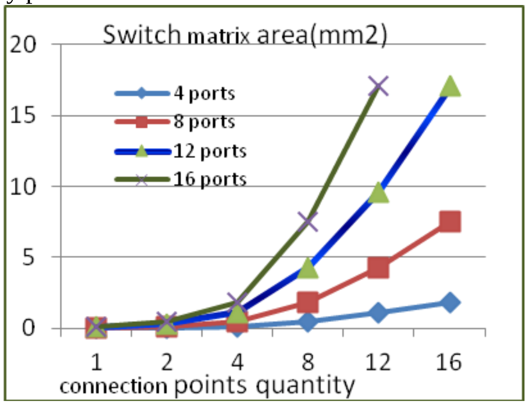


Fig. 3 The switch matrix hardware cost

Thus the 1st way of a router structure hardware is essentially constrained.

IV. *Theoretical parameters evaluation*

Maximum/minimum delay and jitter are calculated for the proposed router structures. The following assumptions were made during calculations: the packet size for the virtual channel was the same for every source; for every virtual channel data transmission is enabled in every time slot; Nchars are written to TX and RX buffers of each port at the same amount of time; the packet size for every virtual channel is less then frame size; the frame size is less then buffer size for every virtual channel; $\mathrm{TcalcPrec}_k$ for every port of a device has the same value.

Notation:

i - an identifier of a virtual channel;

k - an identifier of a node (a terminal node or a router);

l - an identifier of a link;

p - an identifier of a port ;

h - an identifier of a virtual channel with the highest priority;

sizeF - the frame size in bytes;

$\mathrm{sizeP\_VC}_i$ - a packet size for the virtual channel i in bytes;

$\mathrm{sizeB\_VC}_i$ - a buffer size for the virtual channel i in bytes;

$countSw\_VC_i$ - the number of routers which should be passed for transmission of data of the virtual channel *i*;

$countLink\_VC_i$ - the number of links which should be passed for transmission of data of the virtual channel *i*.

$countLink\_VC_i = countSw\_VC_i + 1$

v_l - a data rate in the link *l*, Gb/s. We assume that it is equal to 1 Gb/s for all links.

$v = 1024^3 bit/s = 128 \cdot 1024^2 byte/s$

$Tbyte_l$ - the transmission time of 1 Nchar (1 byte) through the link *l*.

$Tbyte_l = \frac{1}{v_l}$; for the data rate of 1 Gb/s

$Tbyte_l = \frac{1}{v} = \frac{1}{128 \cdot 1024^2} \approx 7{,}5 \cdot 10^{-9} s$

f_k - an operating frequency of the node *k*, MHz.

$jitter\_VC_i$ - jitter of data packets for the virtual channel *i*;

$minDelay\_VC_i$ - the minimal packet's transmission delay for thr virtual channel *i* for the whole transmission path;

$maxDelay\_VC_i$ - maximal packet's transmission delay for virtual channel *i* for the whole transmission path;

$jitter\_VC_i = max Delay\_VC_i - min Delay\_VC_i$

$\{SourceVC_i\}$ - a set of source nodes for the virtual channel *i;*

$\{DestinVC_i\}$ - a set of destination nodes for the virtual channel *i*;

$TwrByteTX_k$ - time of writing of 1 Nchar into the TX buffer of the node *k;*

$TwrByteRX_k$ - time of writing of 1 Nchar into the RX buffer of the node *k* (for this implementation it is equal to the transmission time of 1 byte through the SpaceFibre link, i.e. $TwrByteRX_k = Tbyte_l$)

$TcalcPrec_k$ - time of the Precedence calculation for all virtual channels in node *k*. This parameter is defined by the developer of the system.

$DelaySwMatrix_k$ - the delay of accessing to routing table and selection of connection points in a router with identifier *k*. This time is necessary to connect the input port with the output port for data transmission.

Calculation of the minimum data transmission delay for the virtual channel *i*:

$\{minLinkVC_i\}$ - a set of physical links, which constitute the shortest data transmission path for the virtual channel *i*.

$\{minSwVC_i\}$ - a set of routers, which constitute the shortest data transmission path for the virtual channel *i;*

$minDelaySource\_VC_i$ - the minimal processing delay in a packet's source of the virtual channel *i*

$$minDelaySource\_VC_i = \min_{k \in \{SourceVC_i\}} (sizeP\_VC_i \cdot TwrByteTX_k + TcalcPrec_k)$$

$min DelayDestin\_VC_i$ - the minimal processing delay in a receiver of the virtual channel i

$$min DelayDestin\_VC_i = = \min_{k \in \{DestinVC_i\}} (sizeP\_VC_i \cdot TwrByteRX_k)$$

$minDelaySw_k\_VC_i$ - the minimal delay in a router for packets of the virtual channel i. We assume that there is no competition between packets of one virtual channel and that different virtual channels do not compete in the router's output port.

$$min DelaySw_k\_VC_i = sizeP\_VC_i \cdot TwrByteRX_k + + DelaySwMatrix_k + TcalcPrec_k$$

$$\begin{aligned} minDelay\_VC_i &= minDelaySource\_VC_i + \\ &+ \sum_{l \in \{minLinkVC_i\}} Tbyte_l \cdot sizeP\_VC_i + \\ &+ \sum_{k \in \{minSwVC_i\}} minDelaySw_k\_VC_i + \\ &+ minDelayDestin\_VC_i \end{aligned}$$

Calculation of the maximum data transmission delay for the virtual channel *i*:

$\{max LinkVC_i\}$ - a set of links, which constitute the longest data transmission path for the virtual channel *i;*.

$\{max SwVC_i\}$ - a set of routers, which constitute the shortest data transmission path for virtual channel *i;*

$maxDelaySource\_VC_i$ - the minimal processing delay in a packet's source of the virtual channel *i;*

$\{allVC_p\}$ - a set of virtual channels, which are supported in the port with identifier *p* of a node.

$\{allVCHighPriority_i\}$ - a set of virtual channels, which are supportes in the port with identifier *p* of a node and have a higher priority than the priority of the virtual channel *i;*

$\{allPortVC_i\}$ - a set of node's ports which support data transmission via the virtual channel *i*;

$$\begin{aligned} maxDelaySource\_VC_i &= \\ \max_{k \in \{SourceVC_i\}} \{ \max_{p \in \{allPortCV_i\}} &(sizeP\_VC_i \cdot TwrByteTX_k + \\ &+ TcalcPrec_k + \\ &+ \sum_{j \in allVC_p, j \neq i} (sizeP\_VC_j \cdot TwrByteTX_k + TcalcPrec_k))\} \end{aligned}$$

$maxDelayDestin\_VC_i$ - the maximum processing delay in a destination node for packets of the virtual channel *i*

$$\max DelayDestin\_VC_i =$$
$$= \max_{k \in \{DestinVC_i\}} (sizeP\_VC_i \cdot TwrByteRX_k)$$

$\max DelaySw_k\_VC_i$ - maximum delay in a router for packets of the virtual channel *i* for the case when the competition exists between the packets of one virtual channel and the packets of different virtual channels for the switch output port..

$$\max DelaySw_k\_VC_i =$$
$$= sizeP\_VC_i \cdot TwrByteRX_k + DelaySwMatrix_k +$$
$$+ TcalcPrec_k +$$
$$(|\{allPortVC_i\}| - 1) \cdot$$
$$\cdot (sizeP\_VC_i \cdot TwrByteRX_k +$$
$$+ \sum_{j \in \{allVCHighPriority_i\}, j \neq i} (sizeP\_VC_j \cdot TwrByteRX_k + TcalcPrec_k) +$$
$$+ TcalcPrec_k)$$

$$maxDelay\_VC_i =$$
$$= maxDelaySource\_VC_i + \sum_{l \in \{maxLinkVC_i\}} Tbyte_l \cdot sizeP\_VC_i +$$
$$+ \sum_{k \in \{maxSwVC_i\}} maxDelaySw_k\_VC_i + maxDelayDestin\_VC_i$$

Calculation of the data transmission delay and jitter for the virtual channel with the highest priority.

The following restrictions were made during calculations: for every virtual channel data transmission is enabled in every time slot; all routers contain only one connection point for each port.. The connection point is shared by all virtual channels of the corresponding port.

$jitter\_VC_h$ - jitter for packets of the virtual channel with the highest priority.

$minDelay\_VC_h$ - the minimal packet transmission delay for the virtual channel with the highest priority for the whole transmission path.

$maxDelay\_VC_h$ - the maximal packet transmission delay for the virtual channel with the highest priority for the whole transmission path.

$$jitter\_VC_h = \max Delay\_VC_h - \min Delay\_VC_h$$

The value of the **minimal** packet transmission delay for the virtual channel with the highest priority is equal to the value of the minimal packet transmission delay for the virtual channel of an arbitrary priority.

$$minDelay\_VC_h = minDelay\_VC_i$$

The value of the **maximal** packet transmission delay for the virtual channel with the highest priority is not equal to the minimal packet transmission delay for the virtual channel of an arbitrary priority.

$maxDelaySource\_VC_h$ - the maximal packet processing delay for the virtual channel with the highest priority in a source node.

$$maxDelaySource\_VC_h =$$
$$\max_{k \in \{SourceVC_h\}} \{ \max_{p \in \{allPortVC_h\}} (sizeP\_VC_h \cdot TwrByteTX_k +$$
$$+ (size\ F - 1) \cdot TwrByteTX_k + TcalcPrec_k \}$$

$maxDelayDestin\_VC_h = maxDelayDestin\_VC_i$ - the maximal packet processing delay in a destination node for the highest priority virtual channel is equal to the maximal packet processing delay in a destination node of an arbitrary virtual channel priority.

$\max DelaySw_k\_VC_h$ - the maximal delay in a router for packets from the virtual channel with the highest priority. We assume that the frame of the lower priority packet is already being transmitted and there is a competition between the packets of virtual channels of the same priority in output port of router.

$$\max DelaySw_k\_VC_h = sizeP\_VC_h \cdot TwrByteRX_k + DelaySwMatrix_k$$
$$+ TcalcPrec_k + (sizeF - 1) \cdot TwrByteRX_k +$$
$$(|\{allPortVC_h\}| - 1) \cdot (sizeP\_VC_h \cdot TwrByteRX_k + TcalcPrec_k)$$

$$maxDelay\_VC_h = maxDelaySource\_VC_h +$$
$$+ \sum_{l \in \{maxLinkVC_h\}} Tbyte_l \cdot sizeP\_VC_h +$$
$$+ \sum_{k \in \{maxSwVC_h\}} maxDelaySw_k\_VC_h + maxDelayDestin\_VC_h$$

$$jitter\_VC_h = \max Delay\_VC_h - \min Delay\_VC_h =$$
$$= (sizeF - 1) \cdot TwrByteRX_k +$$
$$+ \sum_{k \in \{maxSwVC_h\}} \begin{matrix} ((sizeF - 1) \cdot TwrByteRX_k + \\ (|\{allPortVC_h\}| - 1) \cdot \\ (sizeP\_VC_h \cdot TwrByteRX_k + TcalcPrec_k)) \end{matrix}$$

Connection point is not allowed to switch between virtual channels.

$$jitter\_VC_h = \max Delay\_VC_h - \min Delay\_VC_h =$$
$$= (sizeP\_VC_l - 1) \cdot TwrByteRX_k +$$
$$+ \sum_{k \in \{maxSwVC_h\}} \begin{matrix} ((sizeP\_VC_l - 1) \cdot TwrByteRX_k + \\ (|\{allPortVC_h\}| - 1) \cdot \\ (sizeP\_VC_h \cdot TwrByteRX_k + TcalcPrec_k)) \end{matrix}$$

$sizeP\_VC_l$ - the maximum packet size for the virtual channel with the lower priority *l*.

V. Timing Characteristics Estimation

A. Network models

Timing characteristics estimation was done on the basis of the models, which are depicted in *Fig. 4* and *Fig. 5*.

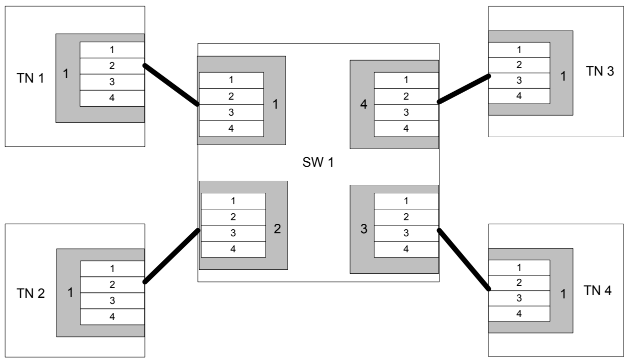


Fig. 4 Network model 1

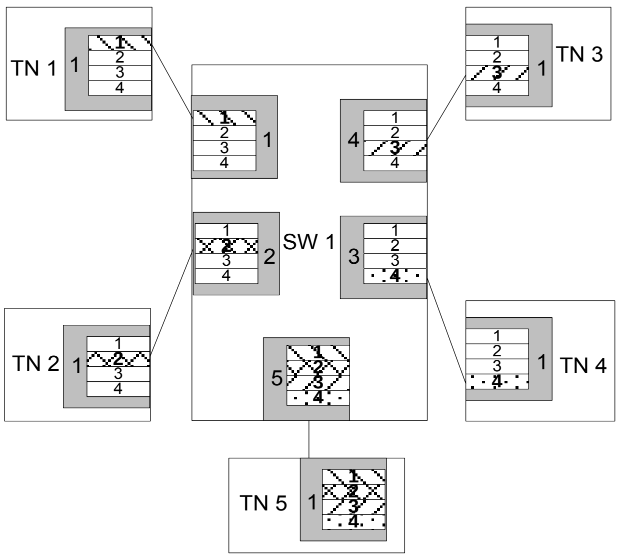


Fig. 5 Network model 2

The Network model 1 comprises a router with 4 ports, each of which can work with 4 virtual channels. Terminal nodes generate packets in a random time moments. At these random moments the terminal node sends the generated packets to each virtual channel. The destination nodes for each virtual channel are also chosen randomly and can be different for the virtual channels. This configuration can lead to a potential possibility of data packets flow concurrency in the output port. The second way of the router organization can also result in the concurrency for the connection points in the input ports.

In its turn the Network model 2 comprises a router with 5 ports. Four of these ports are connected with the terminal nodes, which are sources of packets, and the fifth port is connected to the terminal node, which is a destination node for all packets flows. According to this model, each packet source generates packets for only one particular virtual channel. Such model gives an ability to investigate the characteristics in case of the concurrency of packets flows in the output port 5. In contrast to the first network structure, the distortion of data flows characteristics can be more considerable as all data flows shall be transmitted through the same output port.

These particular models were chosen in order to investigate how the common virtual channels recourses utilization on the network layer of a router can affect the timing characteristics of the virtual channels as well as degradation of the quality of service.

B. Models' implementation and simulation results interpretation

The results of the simulation can significantly depend on the router model implementation features such as local clock frequency and link capacity within the router. We used two different router models for our investigation and built the network models on the basis of these routers.

The first model was created on the basis of the SystemC SpaceWire-RT Network model, which was implemented in the scope of the SpaceWire-RT project [3]. According to the SpaceWire-RT Outline Specification, SpaceWire-RT standard takes SpaceFibre as the basis. This model defines a router which implements the protocol layers from the Serialisation Layer up to the Network Layer. The link bandwidth in the model is set to 2 Gbits/s.

Also the concerned network was simulated on the adapted DCNSimulator model [4]. In this case we used the router and node models which comprise only the Virtual Channel and the Network Layers (this gave an opportunity to reduce the simulation time and to obtain the more detailed results). The link bandwidth in the model is set to 1 Gbit/s.

The simulation of the Network 2 model resulted in the following latencies which are given in Table 1. The obtained latencies for two models differ by two times. This can be explained by the two times difference in the link bandwidth in the models.

TABLE I. THE COMPARISON OF SIMULATION RESULTS

Virtual Channel number	Latency in the SpaceWire-RT Network model, μs	Latency in the DCNSimulator, μs
1	3,43	6,09
2	4,64	8,12
3	5,85	10,15
4	7,064	12,18

Therefore, the absolute values of the timing characteristics are scaled proportionally to the change of the link bandwidth in case of router characteristics alternation such as a local frequency and/or link capacity. In spite of scaling the general relation remains the same.

C. Estimation of achievable characteristics of the Network model 1

Let us consider the case when each virtual channel has its own particular priority level, which corresponds to the virtual channel number: VC1 – the highest priority, VC4 – the lowest. The packet length does not exceed the frame length. *Fig. 6* shows the simulation results for the 1st way of router implementation, *Fig. 7* – for the 2nd way of router implementation with only one connection point for each port.

According to the diagrams, the reduce of the connection points quantity leads to a minor increase of the average time of packet transmission and to a considerable increase of jitter, especially for the low priority levels. As for the highest priority level, there is almost no difference.

However, if the packet length exceeds the frame length the achievable characteristics for the 2nd way (average transmission time and jitter) would be considerably worse than for the 1st way especially for the highest priority levels (*Fig. 8 – Fig. 11*). The packet length in this simulation was set to 750 bytes.

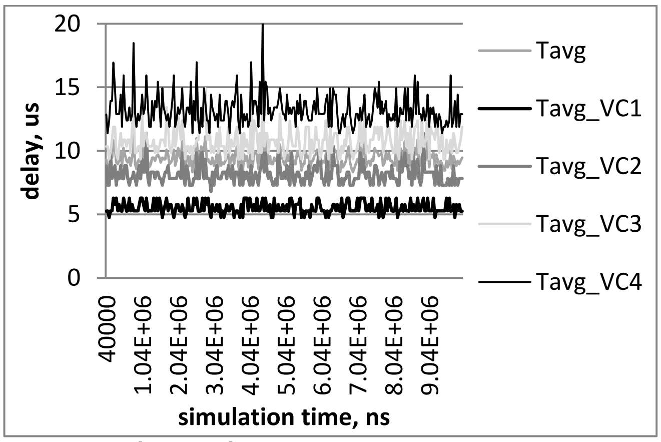


Fig. 6 Simulation results 1

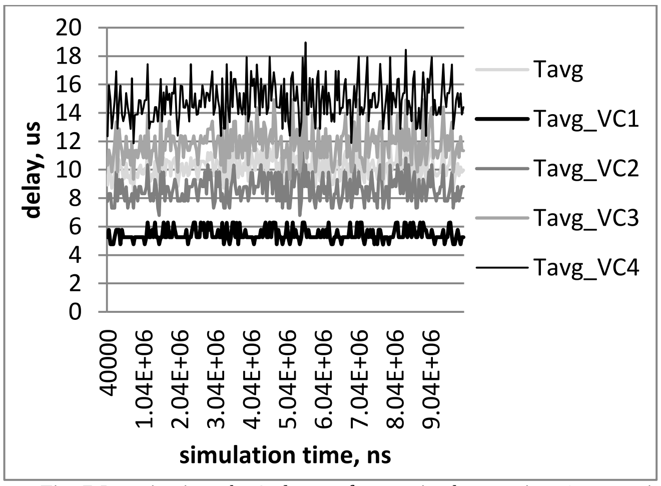


Fig. 7 Investigation: the 2nd way of router implementation, 1 connection point

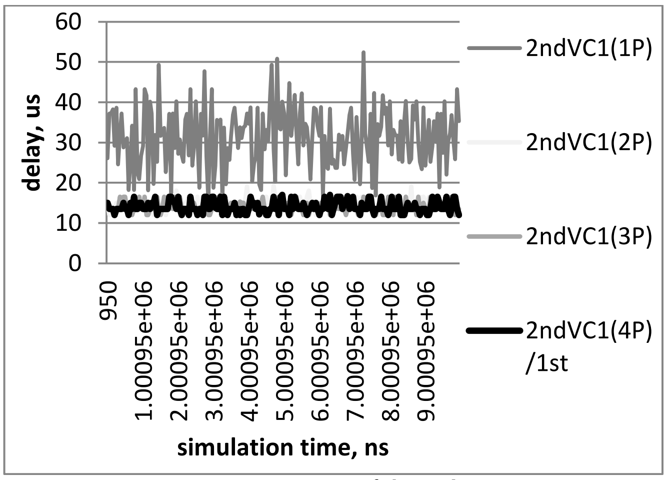


Fig. 8 Investigation 9. Comparison of the packet transmission time via VC1 in case of different connection points quantity

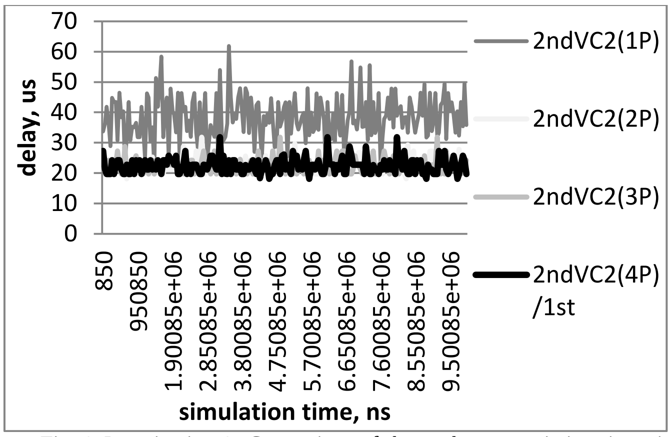


Fig. 9 Investigation 9. Comparison of the packet transmission time via VC2 in case of different connection points quantity

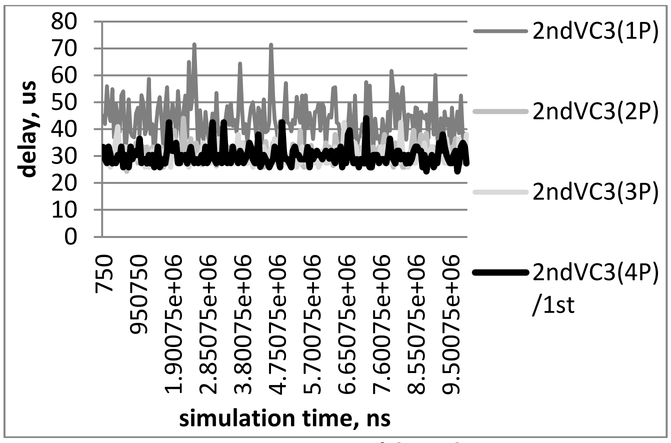


Fig. 10 Investigation 9. Comparison of the packet transmission time via VC3 in case of different connection points quantity

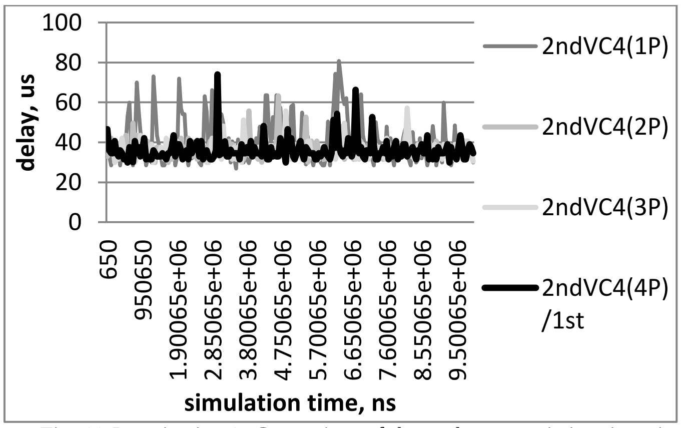


Fig. 11 Investigation 9. Comparison of the packet transmission time via VC4 in case of different connection points quantity

The average transmission time and jitter grow proportionally with the increase of the packet length while using one connection point (*Fig. 12*- packet length – 1000 bytes, *Fig. 13* – packet length – 1500 bytes)

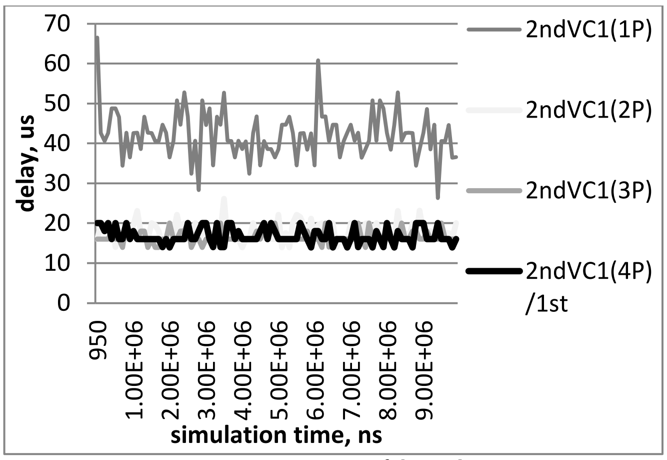


Fig. 12 Investigation 14. Comparison of the packet transmission time via VC1 in case of different connection points quantity

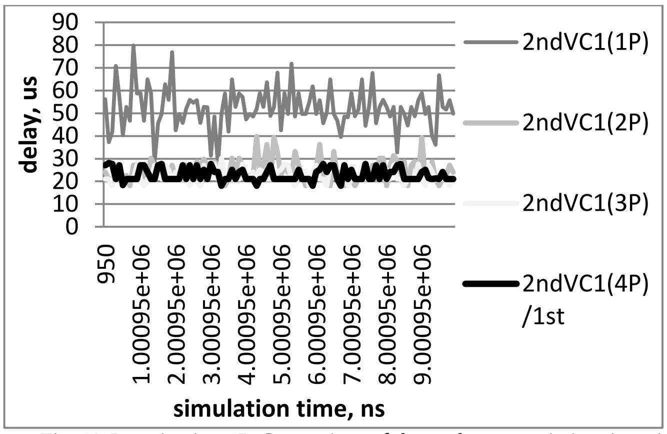


Fig. 13 Investigation 15. Comparison of the packet transmission time via VC1 in case of different connection points quantity

The increase of the connection points' quantity up to two points gives an ability to significantly reduce the difference between the characteristics. The difference for the 2nd way with two and three connection points is very small and the characteristics, obtained for them are very close to the 1st way characteristics.

This investigation shows that the 2nd way of router implementation with one connection point has a significant drawback. The addition of one more connection point (i.e. two connection points for the port) provides a possibility for improvement of the average transmission time and jitter values. The average transmission time values are almost similar for the 1st way of the router implementation and for the 2nd way with two connection points. As for jitter, it is 20% higher for the 2nd router structure.

D. Estimation of achievable characteristics of the Network model 2

In contrast to the Network model 1 in this case the competition between data flows exists only in the output port, but it is stronger because of the essential data flows intensity.

If a packet length is smaller than the frame length then the average packet transmission time for the 1st and the 2nd ways of the router implementation is almost the same, *Fig. 14*. This result is coinciding with results obtained on the Network model 1.

Similarly to the Network model 1, the average transmission time and jitter grow proportionally with the increase of the packet length in case of using the 2nd way of the router implementation with one connection point (*Fig. 15* – packet length – 1000 bytes, *Fig. 16* – packet length – 1500 bytes).

The increase of the connection points' quantity up to two points gives an ability to significantly reduce the difference between the characteristics of the 1st and 2nd ways. Similarly to the Network model 1 in this case the difference of jitter between these router's implementations is not more than 20%.

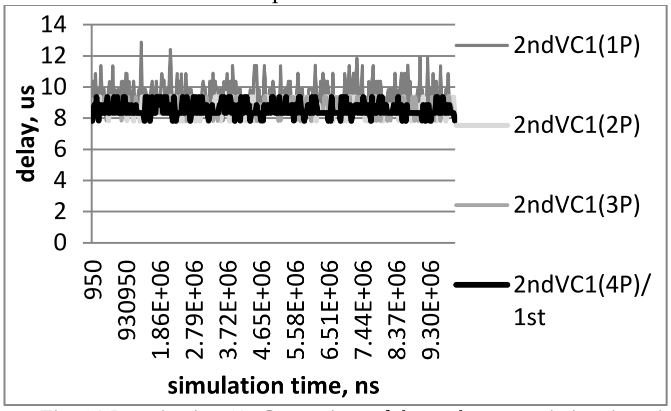


Fig. 14 Investigation 13. Comparison of the packet transmission time via VC1 in case of different connection points quantity

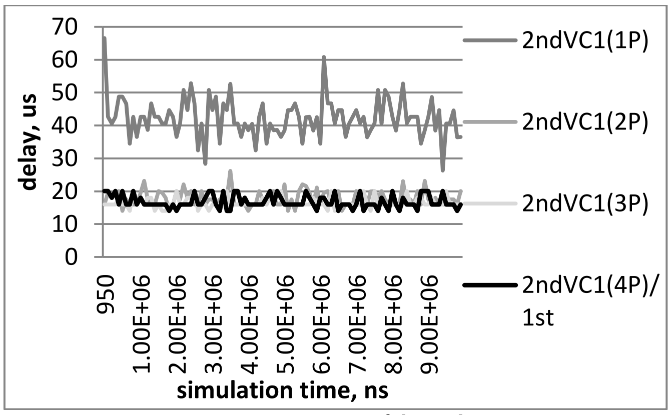


Fig. 15 Investigation 14. Comparison of the packet transmission time via VC1 in case of different connection points quantity

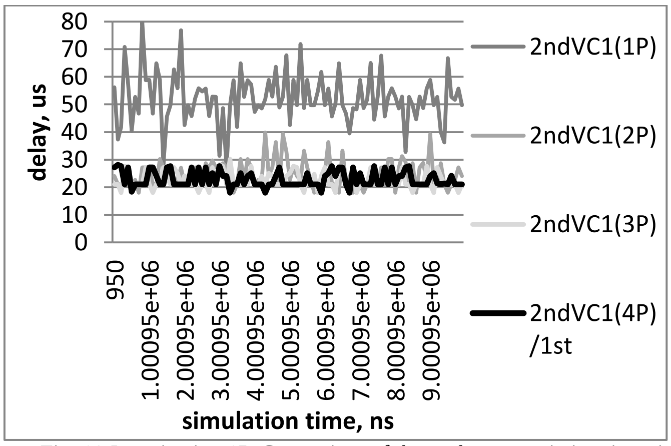


Fig. 16 Investigation 15. Comparison of the packet transmission time via VC1 in case of different connection points quantity

VI. Conclusion

According to the investigations made the 1st way of the router organization results in the limitations in hardware implementation. The comparison of the achievable timing characteristics for different ways of router implementation showed that if a packet size is smaller than the frame size then the average packet transmission time for both ways is almost similar. Jitter of the low priority traffic grows faster for the 2nd way of the router implementation.

Therefore, the 2nd way of the router implementation with one connection point can be used for the networks with the packet length shorter than frame size. In this case it will provide scheduled, bandwidth reserved and priority qualities of service.

The packet lengths larger than the frame size while using the 2nd way of the router implementation result in degradation of the timing characteristics in comparison with the 1st way.

This degradation of characteristics grows proportionally to the packet's length of the virtual channels of low priorities. Consequently, the 2nd way of the router implementation with one connection point in networks where long packets are transmitted is possible only when there are no hard real time requirements and jitter constraint's.

In such systems the achievable link utilization will be less than the physical link throughput. The link can stay and wait for the rear frames of the transmitted packet (frames from others virtual channels could not go to this output port because the connection point is occupied by the current packet).

The 2nd way of the router implementation with two connection points essentially decreases these disadvantages. The average packet transmission time and achievable link utilization in this case are almost similar to the 1st way of the router implementation.

Jitter is 20% bigger for the 2nd way of the router implementation with 2 connection points than for 1st way. Therefore, the achievable characteristics for the scheduled service and jitter value for this 2nd way of router implementation are 20% lower.

The achievable characteristics for the priority and bandwidth reserved (without jitter constraint) qualities of service are practically the same for the 2nd way of router implementation with 2 connection points and for the 1st way.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under grant agreement n° 263148.

REFERENCES

[1] *S. Parkes, A. Ferrer, A. Gonzalez, C. McClements, D3.2-SpaceWire-RT Updatated Specification. Annex 1 SpaceFibre Standart, Febrary 2013*

[2] *H. Kopetz, Real-Time Systems Design Principles for Distributed Embedded Applications, Second edition, 2011*

[3] *V. Olenev, I.Lavrovskaya, I. KorobkovH. SpaceWire-RT/SpaceFibre Specification and Modeling, 2013*

[4] *A. Eganyan, L. Koblyakova, E. Suvorova. SpaceWire network simulator. SpaceWire-2010. Proceedings of international SpaceWire conference St.Petersburg 2010, p.403-406. ISBN^ 978-0-9557196-2-2*

[5] *Steve Parks, Chris McClements, Martin Dunstan Albert Ferrer, Alberto Gonzalez. SpaceFibre.SpW WG October 2012*

Poster Presentations

1 Tbits of data serviced by SpW

Christophe Delay
SYDERAL SA
Gals, Switzerland

Stephane Humbert
SYDERAL SA
Gals, Switzerland

***Abstract*—The GAIA Mass Memory receives asynchronous scientific data from the instruments via seven Spacewire links @ 40 Mbps. These 7 SpW are included as IPs in a single FPGA. GAIA launch is planned end 2013.**

***Index Terms*—Mass Memory, file system, reliability**

I. Introduction

The Gaia mission, lead by ASTRIUM, is an ESA mission that will allow the creation of a precise three-dimensional map of about one billion stars throughout our Galaxy and beyond.

Seven 'Video Processing Units' collect the astrometric and photometric observations of objects passing across the Gaia focal plane. The data produced by these units must be stored prior to transmission to the ground. Due to the large amount of data produced and limited visibility of ground stations, Gaia contains a dedicated data storage unit - the Payload Data Handling Unit (PDHU).

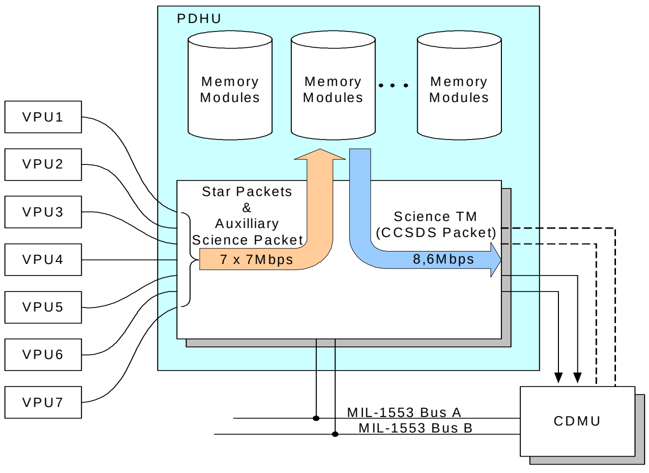


Fig. 1. GAIA PDHU DATA FLOW

II. Description

The PDHU is mass memory equipment which offers a storage capability of more than 1Tbits beginning of life and more than 800Gbits end of life.

The PDHU accepts and handles stores and reads the following data flow in parallel:

- "Science Data Packets" Data packets from the 7 VPUs by the SpW links
- "Auxiliary Science Data Packets" Transmitted by the 7 VPUs by the SpW links
- "Telemetry science Data" Packets sent by the PDHU to the CDMU

In addition to this task, the PDHU manages the TM/TC links with the CDMU (Command and Data Management Unit), maintains its internal reference time (periodically re-synchronized with the On Board Timing via MIL-1553), controls the sectors deletion and recovery processes. The equipment includes both hot and cold redundancy depending on the functionalities. The Hardware-Software functionalities require a very high level of interactions.

III. Main Challenges

The design, development manufacturing and tests of the equipment have led to some major challenges successfully achieved leading to the PDHU being recognized as one of the most challenging electronic equipment of the GAIA satellite. During the design phase the reliability was carefully looked at in order to reach a 0, 99 figure over the mission life time of 5,5 years.

Due to the particular location of the equipment the mechanical design faced a thermal request for no conductive thermal dissipation. Thermal Vacuum tests have shown that the maximum power of 35 Watts could be dissipated via thermal radiation only.

In terms of mass memory operation the two main topics are the file management with a very fast response time and the asynchronous nature of the data inputs. Packets handling of sizes between 17bytes and 96 Kbytes is performed in a typical response time of 3,5 µsec.

All 7 SpW inputs are handled with IPs integrated in a single RTAX FPGA with the fully asynchronous data handling performed at a typical data rate of 40Mbps (useful data at 30Mbps), with possible maximum speed above 100Mbps.

IV. EQUIPMENT DESCRIPTION

The PDHU has the following major characteristics:

- Mass: 14 kg
- Power : 26 Watts
- Volume : 2,3 liters

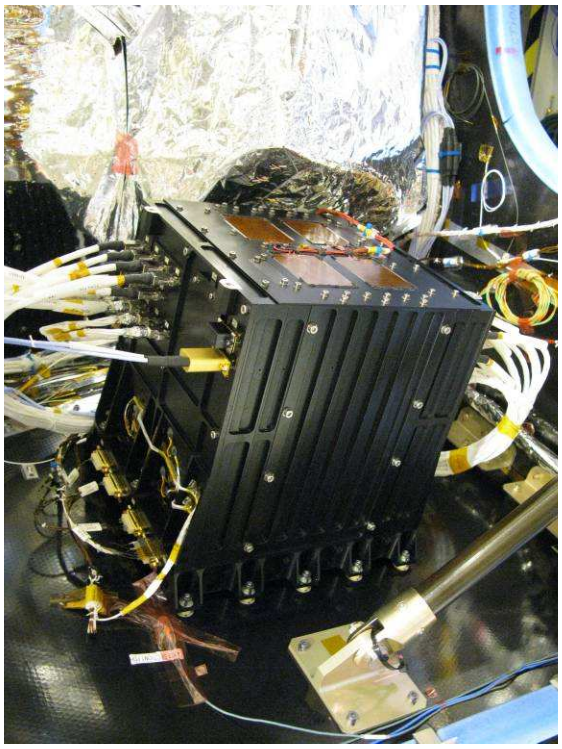


Fig. 2. PDHU FM in the satellite (Credit:ASTRIUM)

The PDHU model “Fig. 2” is composed of three different types of boards:

- Memory Boards
- Controller Boards (see “Fig. 3”)
- Power Supply Boards

The controller board, based on a LEON 2 FT ASIC, includes all input-output-command-control functions. All aspects related to the SpW inputs are included within this single board.

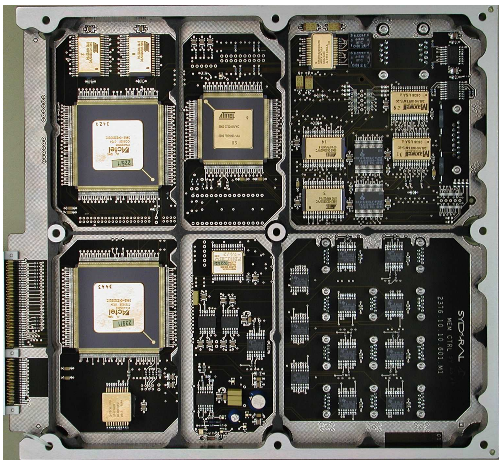


Fig. 3. CONTROLLER BOARD

The specificity of the equipment stands also in the fact that the file system in largely based on an Hardware approach rather then on usual software ones . This has been triggered by the very short file-system response time required (below 5µs in the worst case) due to the inputs signal speed. Further details on this very specific implementation is provided in § 5.

The software services implemented are equivalent to the so called PUS Service 13.

The equipment is based on mixed of cold and hot redundancy concept. The power supplies are used in hot redundancy while the controller and memory functions are in cold redundancy.

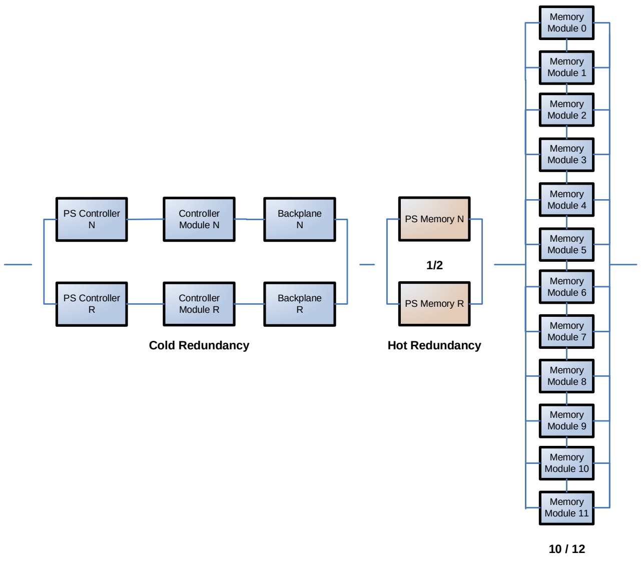


Fig. 4. REDUNDANCY CONCEPT

V. HARDWARE BASED FILE SYSTEM

Every packet received from the VPU or from the LEON is divided in Blocks of 64 Bytes, whatever the size of the incoming packet. That is, the Block containing the end of the packet will probably not be full and the padding of the Block is not used.

A Block contains the data of the packet and the corresponding Reed-Solomon control bits. Considering a Reed-Solomon code 64 / 60, the data payload are 60 Bytes, 4 Bytes are reserved for the RS control Bytes. This code can correct up to 2 Bytes anywhere in the data field.

When a packet is coming, the Packet size is sent to the FSM by the mean of the data link interface. The FSM has the responsibility to calculate the real packet size that will be taken in the mass memory with the formula:

- Number of Blocks = (Star Packet size + 59) / 60

The File management is Hardware implemented inside the so called Management FPGA.

The PDHU file system management is mainly, bit not only, based on two tables, the File Header Table (FHT) and the Sector Link Table (SLT), which allows the memory allocation of the entire Memory Array. The File System Manager has the responsibility to manage all the 256 files stored in the Memory Array.

The File Manager is responsible to process each request, access and update the file system tables. As FHT is frequently accessed, and in order to reduce the file system request latency, it is stored in the embedded FPGA memory while the Sector Link Table is stored in the external SRAM memory. All the memories are protected by EDAC.

The requests to the file system are issued from the VPU I/F located inside the Data Link FPGA.

Each time a new star packet or an auxiliary science data is received from a VPU, the VPU Interface extracts the File ID and packet size from the incoming packet and performs a file system request to the File System Manager, sending these two data.

The FSM has to define the number of Blocks to be implemented in the Memory Array. The Number of Blocks corresponds to: (Star Packet size + 59) / 60.

Based on the File ID, the File System Manager access the File Header Table. The FSM returns, according to the number of Blocks, the corresponding Sector address and Block offset. Further more it updates the File Header Table, the Sector Link Table (SLT), the Block Counter Table (BCT), the Useful Header Table (ULT) and the Sector Size Table (SST).

The deletion process is controlled by the processor through the PCI bus. The File Header Table and the Sector Link Table are accessed by the LEON which performs the necessary tables update to free the selected sectors of files.

The File System Manager is connected with the internal AMBA bus to handle the file system request from the LEON processor and to access the Sector Link Table and the Block Counter Table, which are located in an external memory.

The access of the FHT by the LEON is controlled by the FSM and gives the priority on the access to its own controller.

The “First free sector” register is implemented in an APB register, in order to be also accessible by the LEON.

The “Last free sector” register, used only by the LEON for file deletion, is memorized in the SRAM of the LEON

In the cyclic files, a Sector must be closed on request. A dedicated command sent by the LEON using a register containing the file number where the current sector must be closed

VI. SPW FPGA IMPLEMENTATION

The FPGA including all inputs data processing is dedicated to data link and consists of the following blocs:

- Reset and Clock
- VPU Interface (7x)
- VC Interface
- Time Keeper
- Management Transfer Interface
- Memory Bus Interface
- Memory Module Switch Decoder and Leon Status

The design is based on 7 commercial Spacewire IP (SpaceWire-b CODEC from University of Dundee) included in a single Actel RTAX2000S FPGA with the following architecture:

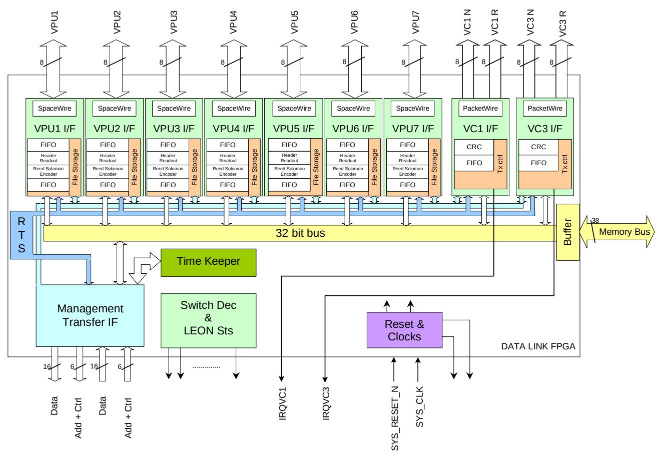


Fig. 5. SPW FPGA ARCHITECTURE

The system requires the implementation of seven (7) SpaceWire interfaces for the GAIA VPU data reception. The requested speed of each SpaceWire link is 40Mbps (RX_CLK = 40MHz) with an average useful data rate over one second of 30Mbps. Due to board design constraints, only one FPGA was foreseen for the seven SpaceWire interfaces.

The main challenge of this FPGA design was the management of the clock networks, since Actel's RTAX FPGA only provides eight (8) dedicated clock buffers (Hardwired Clock and Routed Clock Buffers), and the implementation of the SpW clock recovery logic for the seven links.

Due to design constraints (several clock domains), most of the dedicated clock buffers were already used preventing the implementation of the SpaceWire clock recovery logic as recommended by Actel. .Therefore, in order to provide each SpW IP with a dedicated clock network, local clock feature provided by the FPGA has been used. Each SpW IP is placed in a separate FPGA tile and uses part of the global clock network.

To increase the SpW link speed and clock signal integrity, the clock recovery logic (XOR and first Flip-flops stage) has been manually placed in the FPGA to precisely control delay. The post-layout static timing analysis (**worst-case**) reports RXCLOCK frequency between **114**MHz and **128**MHz on the 7 SpaceWire Interfaces while 40MHz was required.

ACKNOWLEDGMENT

The authors would like to thank P. Lelong, P. Norridge, C. Lloyd and S. King from ASTRIUM as well as T. Paulsen from ESA for their valuable contribution to the development and realization of the equipment.

SpaceWire Validation Test Plan & Conformance Test Bench Prototyping

Session Test & Verification

Fabien Vigeant
Onboard Data Handling Department
CNES
France, Toulouse
fabien.vigeant@cnes.fr

Thierry Parrain
Delta Technologies Sud-Ouest
France, Toulouse
t.parrain@delta-technologies.fr

Abstract — **The aims of the present study were first to establish a SpaceWire Validation Test Plan (SVTP), commonly with Astrium-F and TAS-F, in order to cover all the ECSS standard specifications, addressing in particular the physical and electrical compliance requirements, in which some additional tests were specified. The methodology offered through this SVTP is substantially based on the approach used for the MIL-STD-1553 Validation Test Plan.**

Subsequently, this SVTP has provided the basis to define and realize a prototype of a SpaceWire Validation Test bench aiming to fulfill most of the SVTP specifications. For an optimized covering of these requirements, this test bench has been built by using both the SpaceWire Conformance Tester from STAR Dundee company, associated with a custom tool (called "HOST SpaceWire Traffic Generator" from Delta Technologies Sud-Ouest (DTSO)) developed especially for this validation test plan and providing a large panel of physical and low-level/electrical compliance tests.

Index Terms—**SpaceWire standard, Compliance Testing, LVDS, BER, Eye diagram, design margins, test procedures.**

I. Introduction

SpaceWire is based on two existing commercial standards, IEEE-1355 and LVDS which have been combined and adapted for use on-board spacecraft. Since its introduction in the early 2000s, SpaceWire applications have grown steadily and have led to the development of a large variety of SpaceWire implementations based on different ICs and CODEC IP, conceived independently by different agencies, labs and industrials of the Space Community.

Taking into account that the ECSS-E-ST-50-12C specification lacks detailed requirements concerning in particular the lower levels of the standard, it has become a real challenge to assess the compliance of a SpaceWire interface. In his current state, the chapter 5 of the ECSS standard only provides some basis for the SpaceWire Physical Layer (as highlighted in previous equivalent analysis like [10]).

It is even more essential that newly higher protocols have been added since (RMAP, CPTP, RDDP, …) or are currently under development (SpaceWire-D, SOIS, …), allowing more complex or critical communications to take place on SpaceWire links or networks.

It is in this context that this study has been carried out, aimed at complementing the current standard and providing complete and suitable requirements and tests procedures to insure the compliance of SpaceWire interface for space applications, from Physical Layer to Packet Level.

II. SpaceWire Standard Basis

The SpaceWire Standard ECSS-E-ST-50-12C calls for a Low Voltage Differential Signaling (LVDS) physical layer as defined in ANSI/TIA/EIA-644, Electrical Characteristics of Low Voltage Differential Signaling Interface Circuits. LVDS is the most common differential signaling interface. The low power consumption, minimal EMI, and excellent noise immunity are the features that have made LVDS an interface of choice for many applications.

In its current definition, SpaceWire standard covers up to the network level of the layer-based OSI model, with slight differences in its organization:

- Physical layer which gathers the signal and physical level specifications in a ISO/OSI sense, including : signal voltage levels, signal encoding, noise margins, data rates Connectors and PCB wiring
- Data link layer which lists all the character, exchange and packet level specifications in an ISO/OSI sense, including data and control characters specification, flow control, error detection and link error recovery.

There are different means to ensure compliance with the standard, from a physical and electrical tester (as the SpaceWire Margins Tester presented in [4]) to Base Functional Model-oriented approach for testing and verification of SpaceWire IP-Blocks Interface, as in [1]. Although SpaceWire standard has helped reduce incompatibility problems at the data link and physical layers, there is still the potential for problems at this level. In particular, in some cases the physical layer specification defined in the standard is not fulfilling all requirements of a specific application, and remains incomplete (for instance with regard to common mode range, jitter, bit error rate, …).

Moreover test and verification of interfaces includes different subsets of SpaceWire standard layers, which makes even harder to cover all the possible implementations and application's specific needs.

III. SpaceWire Validation Test Plan

From this perspective, the first task of this study was to establish a SpaceWire Validation Test Plan (SVTP), through a working group led by DTSO and composed by CNES and the two main space-industrial actors Astrium-F and TAS-F, in order to cover all the ECSS standard specifications (from character level to packet level), and also to complement it with regard to the physical and electrical requirements. In the latter case, some additional tests were specified in the SVTP, which introduces a group test classification, at signal level, based on input and output signals. The main requirements added in that section of the SVTP concern common mode, noise, Rise / Fall time, Amplification / Attenuation, Skew, Jitter, BER and dynamic output signal balance tests (see below for further details).

The methodology offered through this SVTP is willingly and substantially based on the approach used for the MIL-STD-1553 Validation Test Plan. This document is built on these main sections:

- Section 2 is the presentation of the documentary reference system, and Section 3 shows a general description of SVTP perimeter,
- Section 4 shows the set of requirements that a SpaceWire interface shall fulfill, and also deals with guideline rules for designing and implementing a SpaceWire Interface (referring either to ECSS standard (ECSS-Q-ST-70-08 or ECSS-Q-ST-70-26) or implementation feedback gathered among the SpaceWire users community). It provides useful recommendations for PCB and backplane tracking as well as conceptual advises to system designer (such as fail-safe extension feature like described in [11]).
- Section 5 presents hardware setup for test execution, section 6 lists the corresponding test procedures and section 7 gathers through a table all requirements and the method that will be applied on each test,
- Section 8 contains the traceability matrix between SVTP requirements and ECSS requirements, and section 9 is for appendixes.

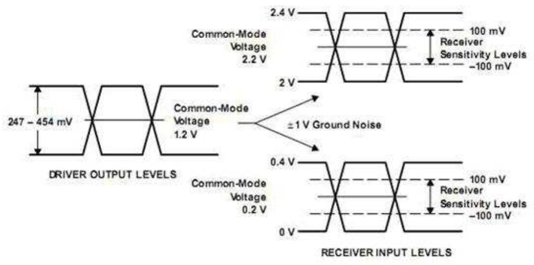

2.4 V
Common-Mode Voltage 2.2 V
100 mV
Receiver Sensitivity Levels
–100 mV
2 V
247 – 454 mV
Common-Mode Voltage 1.2 V
±1 V Ground Noise
DRIVER OUTPUT LEVELS
0.4 V
Common-Mode Voltage 0.2 V
100 mV
Receiver Sensitivity Levels
–100 mV
0 V
RECEIVER INPUT LEVELS

Fig. 1. SVTP Common mode offset requirements

Taking the example of the common mode offset test at the input level of the UUT, according to the SVTP (based itself on LVDS standard), each LVDS input shall be tolerant to a common mode voltage through a range from 0.2V to 2.2V with a maximum of ±1V ground noise. The recommended voltage applied to the receiver is between ground and 2.4 V with a common mode range of 0.05 V to 2.35 V, like depicted in Fig. 1.

For testing the compliance of the UUT, the hardware setup corresponding to this test is based on the SVTP general test setup configuration for testing SpaceWire Inputs, and relies on a specific signal disrupter (offset generator injected at the input level) as illustrated in Fig. 2. below.

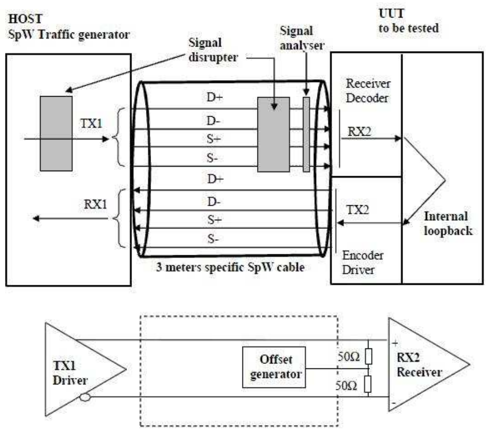
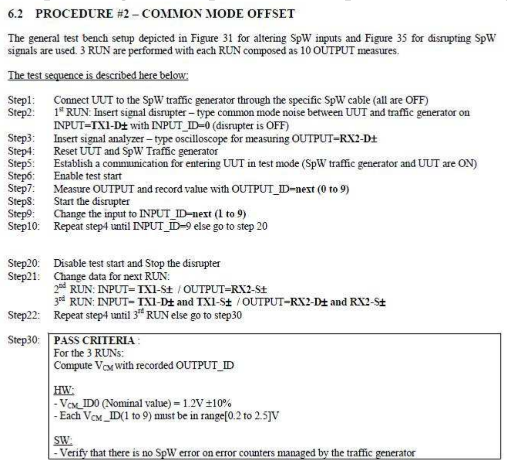


Fig. 2. Hardware setup and signal disrupter for SVTP Common mode offset test

Offset values with 250mV ±10% step are injected onto TX1-D± and/or TX1-S± and expected VCM common voltage read at RX2-D± and/or RX2-S± inputs is measured. VCM is computed as (RX2-D+ + RX2-D-) / 2 or (RX2-S+ + RX2-S-) / 2). The corresponding SVTP procedure is presented in Fig. 3. .


6.2 PROCEDURE #2 – COMMON MODE OFFSET

The general test bench setup depicted in Figure 31 for altering SpW inputs and Figure 35 for disrupting SpW signals are used. 3 RUN are performed with each RUN composed as 10 OUTPUT measures.

The test sequence is described here below:

Step1:	Connect UUT to the SpW traffic generator through the specific SpW cable (all are OFF)
Step2:	1st RUN: Insert signal disrupter – type common mode noise between UUT and traffic generator on INPUT=**TX1-D±** with INPUT_ID=0 (disrupter is OFF)
Step3:	Insert signal analyzer – type oscilloscope for measuring OUTPUT=**RX2-D±**
Step4:	Reset UUT and SpW Traffic generator
Step5:	Establish a communication for entering UUT in test mode (SpW traffic generator and UUT are ON)
Step6:	Enable test start
Step7:	Measure OUTPUT and record value with OUTPUT_ID=**next (0 to 9)**
Step8:	Start the disrupter
Step9:	Change the input to INPUT_ID=**next (1 to 9)**
Step10:	Repeat step4 until INPUT_ID=9 else go to step 20
Step20:	Disable test start and Stop the disrupter
Step21:	Change data for next RUN: 2nd RUN: INPUT= **TX1-S±** / OUTPUT=**RX2-S±**; 3rd RUN: INPUT= **TX1-D± and TX1-S±** / OUTPUT=**RX2-D± and RX2-S±**
Step22:	Repeat step4 until 3rd RUN else go to step30
Step30:	**PASS CRITERIA** : For the 3 RUNs: Compute V_{CM} with recorded OUTPUT_ID HW: - V_{CM}_ID0 (Nominal value) = 1.2V ±10% - Each V_{CM} _ID(1 to 9) must be in range[0.2 to 2.5]V SW: - Verify that there is no SpW error on error counters managed by the traffic generator

Fig. 3. SVTP Procedure for Common mode offset test

The next section of the SVTP gathers in a table all the validation methods applicable for this test, with standard terminology (A: Analysis - I: Inspection - D: Demonstration - F: Frequency = sYstematic/Unitary/Several).

IV. Group Test Classification & Description

Two main categories have been specified in the SVTP, for considering independently the input and output level of a SpaceWire Interface, mainly based on [6]. The main tests addressed in the Input Level Group Test concern:

- Voltage span or differential input voltage threshold

- Bias tolerance or common mode voltage tolerance
- Rise and Fall time tolerance
- Common and differential mode noise tolerance
- Common and differential impedance/ground properties
- Failsafe properties verification

The latter test is not treated as an "execution test" like the others but rather with a design rule due to the black-box testing principles (the UUT is only accessible through his outside connector and not on his internal interface).

The Output Level Group Test gathers the following compliance tests:

- Offset voltage and balance measurements: this measure is of particular interest for measuring design margin since SpaceWire isn't DC balanced.
- Output swing
- Jitter and skew
- Overshoot and undershoot
- Rise and Fall time characterization
- Dynamic Output Balance
- Common and differential impedance/ground properties

Furthermore two more tests have been added in the SVTP:

- Eye pattern test: as presented in Fig. 4. , there is a number of measurements that can be made and extract information from the eye diagram at termination resistor level (offset voltage, output swing, jitter, rise/fall time,). In the SVTP, this is used to correlate the previous individual measurements and also to measure margins on these parameters.

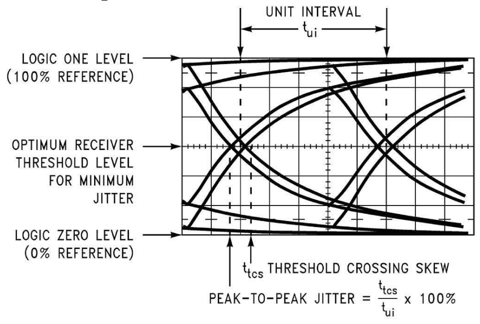

UNIT INTERVAL
t_{ui}
LOGIC ONE LEVEL (100% REFERENCE)
OPTIMUM RECEIVER THRESHOLD LEVEL FOR MINIMUM JITTER
LOGIC ZERO LEVEL (0% REFERENCE)
t_{tcs} THRESHOLD CROSSING SKEW
PEAK-TO-PEAK JITTER = $\frac{t_{tcs}}{t_{ui}}$ x 100%

Fig. 4. Eye pattern measurements

- BER test

BER testing is very time-intensive: the time length of the test is determined by the data rate and also the desired performance bench mark. For example : for achieving a correct BER (Bit Error Rate) < 10^{-12} without noise injection, the test has to be run for @ 1,38 hours at maximum data rate (200 Mbit/s) or @ 27,7 hours at default data rate (10Mbit/s).

To minimize the time required for the BER Test, method similar to MIL-BUS procedure has been adopted [5] : the test is launched while stressing this communication link. Every bit error, time test is increased by transferring more data bit or stop test. The noise test shall run continuously until the total number of data received by the UUT exceeds the required number for acceptance of the UUT or is less than the required number for rejection of the UUT. The measurement is computed automatically by the tester: each data transmitted by the tester is returned by UUT on the worst path, i.e. a cable of maximum length (defined for the equipment) at maximum data rate with pseudo-pattern sequence for data for optimizing Inter Symbol Interference.

It should be noted that prior to executing the compliance tests provided by the SpaceWire Validation Test Bench, the UUT shall enter a specific Test Mode at SpaceWire Interface Level, in which the UUT returns all data packets received on his receive end-point: as recommended in [4], the loop-back is implemented at the SpaceWire CODEC level (Fig. 5.) to insure better results, in particular for BER testing. Currently the SVTP specifies that this Test Mode shall be entered either when receiving a specific time code sequence from the HOST (referred to as Initialization Sequence) or simply by setting a bit in a configuration register after power-up.

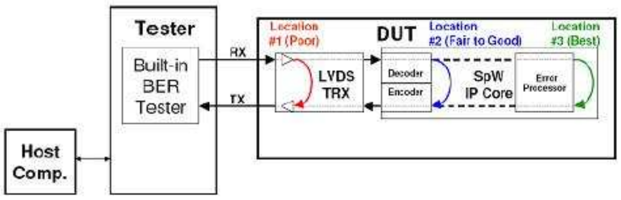


Fig. 5. Loopback implementation for error reporting level & measurement method

V. Validation Approach

Considering the needs for assessing the compliance of a SpaceWire Interface (flight-model or board interface validation, AIT-AIV procedures, including late investigations allowing limited access to the SpaceWire interface), a black-box approach has been adopted: the SpaceWire interface is only available at his external interface (connector only) and not internally at component level.

For that matter, the SpaceWire Conformance Tester (SCT) from Star-Dundee provides a wide range of tests to probe and to insure the compliance with the higher levels/layers of the standard (from character to exchange levels). Indeed even if some low-level (link or bit-level) tests are proposed through this tool, it doesn't provide an acceptable coverage of the physical and signal layers : [2] it is more likely intended for debugging purposes, to the attention of hardware and software engineers developing and using SpaceWire systems.

Using SCT to cover upper layers of the standard is coherent with the black-boxing testing philosophy: the conformance tester has no knowledge of the internal structure or behavior of the UUT and can only investigate conformance by stimulating the UUT in different ways and then analyzing the visible behavior.

Taking into account the features and capabilities of a the SCT, the remaining key need towards SpaceWire Interface Validation resides in physical and electrical layers testing, according to the SVTP requirements previously established, which can't be addressed with the Star-Dundee tool.

VI. SPACEWIRE VALIDATION TEST BENCH

In order to better fulfill all the SVTP requirements taking advantage of the strengths of existing tools, the prototyping of the SpaceWire Validation Test Bench relies partly on the SCT with regard to the data link layer compliance tests (as it provides the best ECSS specifications coverage for this layer), complemented with a specific part to address the remaining SVTP requirements at physical and electrical level of the interface.

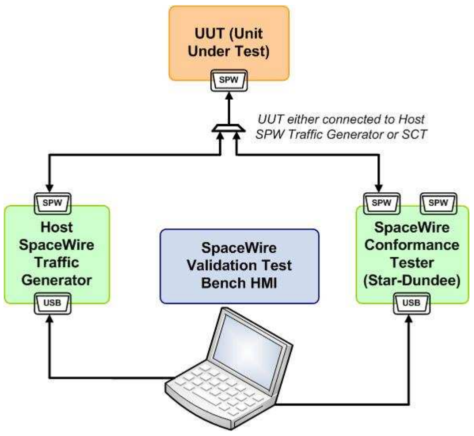

UUT (Unit Under Test)
SPW
UUT either connected to Host SPW Traffic Generator or SCT
SPW
Host SpaceWire Traffic Generator
USB
SpaceWire Validation Test Bench HMI
SPW
SPW
SpaceWire Conformance Tester (Star-Dundee)
USB

Fig. 6. SpaceWire Validation Test Bench Architecture

The overall architecture of the SpaceWire Validation Test Bench is therefore based on the following subsystems or components, as depicted in Fig. 6. :

- SCT from Star-Dundee for executing compliance tests from character level up to packet level
- Host SpaceWire Traffic Generator from DTSO for executing compliance tests at electrical and physical level
- An HMI interface implemented using LabWindows/CVI on Windows PC, to control.

According to the tests to be performed, a set of metrology equipments have to be added like Scope, Ohmmeter, Noise generator, any material for observed physical signals.

Special attention must be paid to the performances of the equipment, the required characteristics being specified in the SVTP. For instance, for a 200Mbit data rate, an adequate oscilloscope should have at least 500 MHz wide bandwidth with sampling frequency greater than 2 GHz, with wide bandwidth differential probe (1GHz).

The Host SpaceWire Traffic Generator is built around two main parts: a digital subsystem, and an analog subsystem, as depicted in Fig. 7. . The architecture and conception of the SpaceWire Validation Test Bench allows to apply (at input level) or characterize (at output level) the worst physical layer conditions the UUT SpaceWire link can tolerate for a given error rate, making possible to easily evaluate margins of the design.

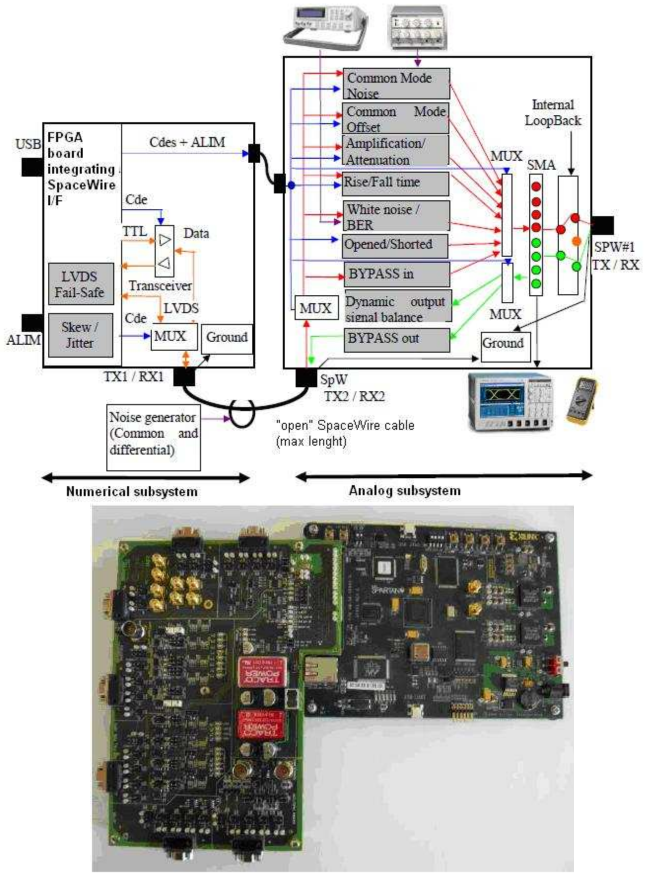


Fig. 7. Host SpaceWire Traffic Generator architecture & bread boarding

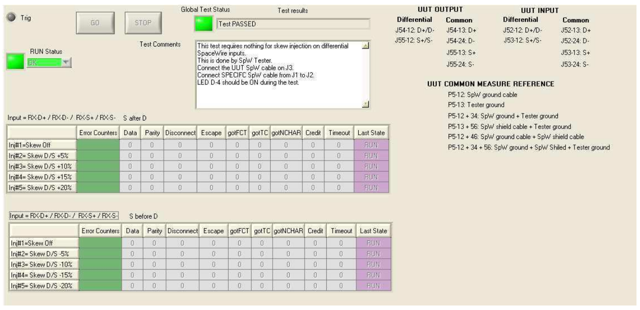
VII. TEST BENCH HOST-MACHINE INTERFACE

The HMI implementation, as depicted in Fig. 8. , is based on configurations file (conf, input, output), which makes the user test environment much more flexible and easy-to-use :

- « config.ini » constitutes the main configuration file, organized in different subcategories, for instance : [COMMUNICATION] for configuring the HMI interface with the test bench, [SPW_PARAM] for configuring the UUT SpaceWire Link, [TEST_PARAM] allowing the user to identify and select a particular subset of tests, depending on the validation perimeter aimed for the UUT.
- Input configuration files depending on the data rate range of the UUT, user has to select the right configuration file depending on the reception data rate of the UUT. These files contain all the predefined parameters for a specific data rate (according to the SVTP specifications), allowing the user to adapt these parameters to its particular needs (rising and falling time, jitter and skew criteria,).
- Output files which correspond to the log outputs of the SVTF and the detailed reports established during the tests execution.

The HMI main features are the following:

- Each test is independent and can be bypassed if needed or impossible to do (for example, if we cannot unplugged 100 Ohms resistor on UUT inputs).
- Each test is based on parameters which can be changed according to user requirements.
- Each test result is printed on HMI with colors code (grey: not passed / green : test passed / red: test failed)
- Each value of observed signal is filled through HMI panels
- The global sanction of SpaceWire conformance is printed in a file and screenshots are added to the final report.



Global Test Status

Test results: Test PASSED

Test Comments:

This test requires nothing for skew injection on differential SpaceWire inputs.
This is done by SpW Tester.
Connect the UUT SpW cable on J3.
Connect SPECIFC SpW cable from J1 to J2.
LED D-4 should be ON during the test.

Input = RX-D+ / RX-D- / RX-S+ / RX-S- S after D

	Error Counters	Data	Parity	Disconnect	Escape	gotFCT	gotTC	gotNCHAR	Credit	Timeout	Last State
Inj#1=Skew Off	🟩	0	0	0	0	0	0	0	0	0	RUN
Inj#2= Skew D/S +5%	🟩	0	0	0	0	0	0	0	0	0	RUN
Inj#3= Skew D/S +10%	🟩	0	0	0	0	0	0	0	0	0	RUN
Inj#4= Skew D/S +15%	🟩	0	0	0	0	0	0	0	0	0	RUN
Inj#5= Skew D/S +20%	🟩	0	0	0	0	0	0	0	0	0	RUN

Input = RX-D+ / RX-D- / RX-S+ / RX-S- S before D

	Error Counters	Data	Parity	Disconnect	Escape	gotFCT	gotTC	gotNCHAR	Credit	Timeout	Last State
Inj#1=Skew Off	🟩	0	0	0	0	0	0	0	0	0	RUN
Inj#2= Skew D/S -5%	🟩	0	0	0	0	0	0	0	0	0	RUN
Inj#3= Skew D/S -10%	🟩	0	0	0	0	0	0	0	0	0	RUN
Inj#4= Skew D/S -15%	🟩	0	0	0	0	0	0	0	0	0	RUN
Inj#5= Skew D/S -20%	🟩	0	0	0	0	0	0	0	0	0	RUN

UUT OUTPUT		UUT INPUT	
Differential	Common	Differential	Common
J54-12: D+/D-	J54-13: D+	J52-12: D+/D-	J52-13: D+
J55-12: S+/S-	J54-24: D-	J53-12: S+/S-	J52-24: D-
	J55-13: S+		J53-13: S+
	J55-24: S-		J53-24: S-

UUT COMMON MEASURE REFERENCE

P5-12: SpW ground cable
P5-13: Tester ground
P5-12 + 34: SpW ground + Tester ground
P5-13 + 56: SpW shield cable + Tester ground
P5-12 + 46: SpW ground cable + SpW shield cable
P5-12 + 34 + 56: SpW ground + SpW Shiled + Tester ground

Fig. 8. SpaceWire Validation Test Bench HMI

VIII. TEST BENCH CONSTRAINTS & LIMITATIONS

The limitations of the current test bench prototyping, mainly due to technical constraints or depending directly on the SVTP orientations, are listed below:

- Other hardware parameters like EMC, ground plan tests, environmental tests, high frequency noise shall be treated at mission/equipment level.
- Transmission & Reception data rate of the test bench are limited between 2Mbps up to 160Mbps.
- Skew and jitter test limitations, due to the limitation of the FPGA technology, which provide limited resources in term of internal delays (high-precision programmable delay element). The max data rate limitation results from these constraints which have led to the development of different test bench configuration files (in particular in term of FGPA bitstreams) depending on the data rate range of the UUT ([2Mbps, 10Mbps], [10Mbps, 20Mbps], ...) for optimized performances.
- Regarding the SVTP noise tolerance requirements, the method using a bulk current injection probe has revealed some limitations mainly due to the test equipment performances and capabilities: it didn't allow to precisely control the injected noise level as required, unlike a noise generator setup which has been used instead. For noise sensitivity measurements, [3] recommends the capacitive coupling (injection with an external function generator through a capacitor) or a bulk current injection method in which an external function generator is previously coupled with a RF amplifier before injection through a bulk current probe.
- Limitations intrinsically due to the black-box testing principles (in particular for fail-safe tests, or for tests that require having access to the internal hardware resources of the UUT Interface)

In the frame of the final task of the study, the test bench has also been validated and operated on CNES products (based on the CEA SpaceWire IP CODEC, licensed by CNES), and is foreseen to be also evaluated and valorized on Astrium product, like SCOC3 or OSCAR equipment.

REFERENCES

[1] Elena Suvurova "Toolset for Test and Verification of IP-Blocks with SpaceWire Interface" International SpaceWire Conference, Session SpaceWire Test & Verification, St. Petersburg State University of Aerospace Instrumentation, Russia

[2] Steve Parkes, Martin Dunstan, "Debugging SpaceWire Devices Using The Conformance Tester", International SpaceWire Conference, Session SpaceWire Test & Verification, University of Dundee

[3] Giorgio Magistrati, "Test on LVDS Components @ ESA", ESA TEC-ED, TEC-EDD, TEC-EDP, 2011 LVDS-Day

[4] Alex Kisin, Glenn Rakow, "SpaceWire Margins Tester", 2008 International SpaceWire Conference, Session SpaceWire Test & Verification, NASA GSFC.

[5] "Validation Test Plan for the digital Time Division Command/Response Multiplex Data Bus Remote Terminals" - SAE AS4111 October 1998

[6] "Electrical characteristics of low voltage differential signaling (LVDS) interface circuits", TIA PN4584 – Revision 1.2 May 2000

[7] Kevin Buchs, Pat Zabinski, and Jon Coker, "Basic Bit Error Rate Analysis for serial data links", Mayo-R-04-07-R0 June 2004

[8] John Goldie, Syed Huk, "LVDS Signal Quality : Jitter Measurement Using Eye Patterns Test Report #1", National Semiconductor Application Note 977, October 1994

[9] John Goldie, Syed Huk, "LVDS Performance : Bit Error Rate (BER) Testing Test Report #2", National Semiconductor Application Note 977, May 1996

[10] Shaune Allen, "SpaceWire Physical Layer Issues", NASA GSFC, 2006 MAPLD International Conference, September 2006

[11] John Goldie "Failsafe biasing of LVDS Interfaces", National Semiconductor Application Note 1194, December 2011

[12] Stephen Campainen, "Low Voltage Differential Signaling", Agilent Technologies Application Note 1382-6

Using SpaceWire In a Intellectualized Data Processor

Session: SpaceWire Missions and Applications,

Poster Paper

ZhouYuan
China Academy of Space Technology(XI'AN)
XI'AN, CHINA
mateyes@163.com

Li Li, Zhang Jian-hua, Cui Wan-zhao, Zhao Jun-yi
China Academy of Space Technology(XI'AN)
XI'AN, CHINA
zjh0901@gmail.com, zhaojy@cast504.com

Abstract*—The new generation meteorologic satellite of China selected SpaceWire as the best solution to satisfy the desire for standard and simple interfaces among instruments of spacecraft. Data generated by science instruments and are sent to intellectualized data processor for checking，multiplexing and formatting. Using SpaceWire as the interfaces between science instruments and intellectualized data processor, we can obtain many benefits, such as having flexible speeds on links, easing connection and control, simplifying the whole network architecture, etc. Thanks to SpaceWire router, several input links can route to a same out port, multiplexing them into a data stream, and we can save a lot of memory space. The intellectualized data processor can check the data from the out ports of SpaceWire router, and diagnose whether the SpaceWire packets is correct. If there has incorrect packets, the intellectualized data processor will deal with them intellectually and transmit the wrong status to Ground Station. This paper describes the SpaceWire network of payload instruments and how the intellectualized data processor works.

Index Terms—**satellite, SpaceWire, data processor**

I. Introduction

FY-4 is the second generation of Meteosat geostationary meteorological satellite, the main development objectives are: satellite attitude stabilization mode is the three-axis stabilized which improve the time resolution observations and regional mobile detection capability; improve imaging device performance, in order to strengthen the ability of monitoring weather systems; development and microwave detection Atmospheric Sounding solve three-dimensional high-orbit remote sensing; the development of extreme ultraviolet and X-ray solar observation, to enhance space weather monitoring and warning.

The payload equipments of the satellite are the two-dimensional scan of 10-channel imager, the atmospheric vertical interferometric detector, the lightning imaging device, and the CCD camera.

At present, many kinds of bus are applied to spacecraft for data transmission, telecommand and telemetry, such as MIL-1553B, CAN, RS485 and RS422. The outburst feature of the these bus is that the data transfer rate is very low, less than 1Mbps as usually. And there are more and more equipments on the spacecraft which the data speed is more than 1Mbps. Therefore we need a high-speed data bus to satisfy this requirement.

In terms of the interfaces of traditional equipments, once connecting the cables, the paths of transmitting data are fixed. And the information among the electronic equipments can't flow freely and share for each other. If there has a router on the spacecraft like that the Ethernet router in our office, we can solve this problem and make the information flow freely among the electronic equipments on satellites.

There is another problem should be solved is that many kinds of electronic equipments on satellite and the system functions are very complicate, so the test and verification for Electronic system is a great challenge, especially finding and locating the faults. To solve this problem, it is necessary to develop the intelligent aerospace electronic equipments which have the function for health monitoring and maintenance.

In order to solve the two problems mentioned above, we developed the intellectualized data processor base on SpaceWire successfully. Using the standard SpaceWire interface simplified and unified the interfaces of equipments, and the SpaceWire router breaks the limit between the traditional electronic equipment to provide information transmission path is fixed, make the device information sharing possible. And the router provide more SpaceWire links, this can realize the transmission link redundancy and fault isolation between the equipments. The data processor also has intelligent data judgment and processing functions, provide the functions of telemetry data monitoring system widely.

II. The Data Processor Routing Function and Advantage Base on SpaceWire

The intellectualized data processor has used the AT7910 which produced by ATMEL as the SpaceWire router. The AT7910 ASIC chip has eight bidirectional SpaceWire interfaces and Two External Interfaces, and data rate from 2 up to 200 Mbps in each direction of SpaceWire link. In the The intellectualized data processor, there are five SpaceWire interfaces are used, as showed in figure 1.

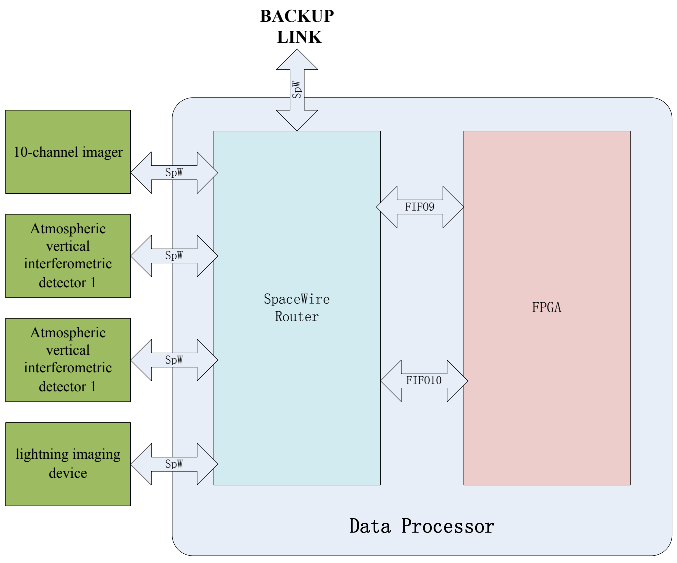


Figure 1 Data Processor Block Diagram

The port1 to port4 of SpaceWire router are connected to the payload equipments, and the local interfaces port9 and port10 are connected FPGA.

Using the SpaceWire router, compared with the traditional data processor, there are following advantages at least:

(1) Simplified the system interfaces. Using the standard SpaceWire interfaces, unified the connection within the electronic system, the SpaceWire cable can be interchangeable, providing convenience for system interconnection and the interface definition.

(2) The data transmission paths are more flexible, the performance of link redundancy is more outstanding. Four SpaceWire ports connected with the payload equipment scan fully interchangeable, the payload equipments only need specified by the PORT9 or PORT10 sends the data to the data processor, without attention from PORT1 to PORT4 which port access. Due to the routing function, payload equipments can be set the corresponding target address, communicate with each other through the SpaceWire router. In addition, the data processor, port 8 connected as a backup interface. When the fault of any link PORT1 to PORT4 occurs, you can easily transmit the link data to the backup equipment through PORT8, which can improve the reliability of data transmission.

(3) Providing the flexibility of Routing configuration, can provide the onboard network management function. The routing chip configuration can be completed by the local PORT9 or PORT10, or by SpaceWire port through the remote configuration commands. Based on the function of routing chip configuration, the link rate can be set to any SpaceWire interface, and the any SpaceWire port can be opened or closed according to the needs. In addition, the parameters of the SpaceWire router can be setted by the SpaceWire node through SpaceWire links.

(4) Providing the function of data multiplexing. Using the routing function, when the data packets come from several SpaceWire links, and routed to the one output port, the data packets are multiplexed in a data stream. Because the SpaceWire data packets are packet EOP or EEP, it can easily find the start and the end of the data packets. And it doesn't need extra storage memory to multiplex the data by user.

III. INTELLIGTENT PROCESSING OF THE DATA PROCESSOR

The intelligent processing functions of the data processor we developed are mainly manifested in the following aspects:

(1) the local configuration and route configuration for SpaceWire router.

(2) The intelligent judgment and processing of the instructions.

(3) The intelligent judgment and processing of the high-speed payload data.

(4) The function of health monitoring.

A. The variety configuration modes of SpaceWire router

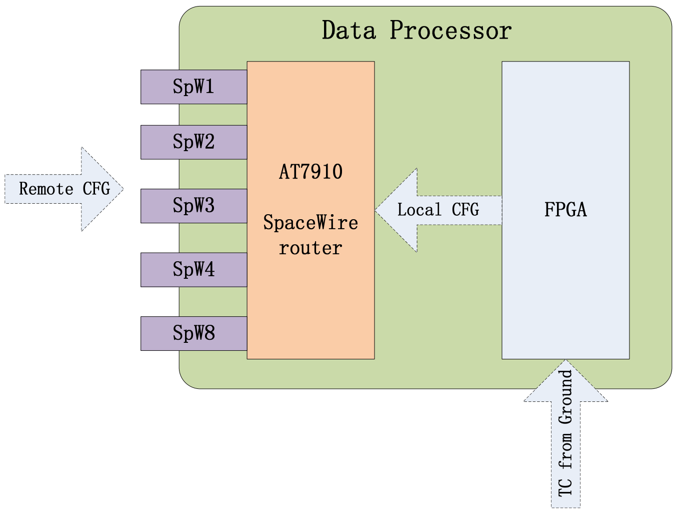


Figure 2 the variety configuration modes of SpaceWire router

The electronic system based on SpaceWire network, the correct configuration of the nodes and router is very important. And if we can modify the configuration parameters of the SpaceWire nodes and routers when it need to, it will be very convenient to operate the equipments onboard and can maximize the system functions. As showed in figure2, the SpaceWire router has at least three configuration modes:

(1) Local configuration mode. When the data processor is powered or reset, the FPGA can configure it as system default mode.

(2) Remote configuration mode by SpaceWire links. The SpaceWrie remote nodes can configure the SpaceWire Router by SpaceWire links if it needs to need change the work mode, such as link speed, link on/off, ect.

(3) Configure the router work mode by telecommand for the ground station. The data processor can receive the telecommands which retransmitted by others equipments, and change the router work mode if it is required.

B. The intelligent judgment and processing of the instructions.

The traditional data processor just passively accepts instructions and executes the instructions. Instruction sender cannot know whether these instructions are correctly received by data processor. This approach is not conducive to situation of the fault condition problems.

In order to solve this problem, the data processor receives the instructions, and checks the correctness of every instructions format and the check-sum, only the correct instruction will be executed. Whether instruction is correct or not, the data processor will return the check the results to the digital command sender. Thus, the sender can judge whether instruction is sent successfully by the return instruction packet content, if there has transmission error, the sender can determine what kind of error is. The intelligent judgment and processing of the instructions can effectively improve the reliability and safety of the instruction transmission.

C. The intelligent judgment and processing of the high-speed payload data

Before processing the high speed data from the SpaceWire links of the payload equipments, the data processor will check the data packets at first. The content which will be checked includes logic address checking, protocol type checking, the counter continuity of the data packets checking and the length of the data packets checking.

If the logic address or the protocol type of the data packet is error, the data processor will discard the incorrect data packet. If the counter of the data packets is not continuous, the data processor just gives a flag to show that the data packet received is not continuous. If the data length is not correct, the data processor will cut the longer data packet into the specified length, and fill the shorter data packet into the specified length. The longer data packet truncation is easy in implementation, but how to fill the shorter data packet into specified length is a little complex.

The following analysis will show the possibility of filling the shorter packets into specified data length in theory.

Hypothesis, there have N channels data will be routed to the same out port of the SpaceWire router, the data rate of the channel *i* is V_i, and $1 \leqq i \leqq N$, *i* is integer. The output rate of the router port is V, and The data length of packets from N channels is the same, L bits.

In order to ensure that the packets are processed and transmitted correctly, it must enquire the inequality 1. Otherwise, the data will be lost.

$$\sum_{i=1}^{N} V_i \le V \qquad (1)$$

Suppose that when the data processor had checking the length of the data packet, and finds the channel s has a shorter data packet, the data length is L_s, *and* $L_s < L$. the data rate of channel s is V_s. The short The data processor will fill the data length from L_s to L. Then the output data amount of the out port per second is:

$$\sum_{i=1}^{N} V_i + \left(\frac{V_s}{L_s} L_s - V_s\right) \qquad (2)$$

In order to ensure the data is not lost, it should meet the inequality 3 as showed below.

$$\sum_{i=1}^{N} V_i + \left(\frac{V_s}{L_s} L_s - V_s\right) \le V \qquad (3)$$

It can be deduced inequality 4:

$$L_s \ge \frac{V_s}{V + V_s - \sum_{i=1}^{N} V_i} L \qquad (4)$$

After testing, if the length of the short packet meets the inequality 4, the data process can fill the short packet into the specified length without data loss.

D. The function of health monitoring

The on-board intelligent data processor base on SpaceWire router has the function of collecting the health states extensively. This includes the following:

(1) Monitoring the state of the SpaceWire router. Such as link speed, link on/off state, link errors and some key parameters of router.

(2) Monitoring the execution of instructions. After checking the received instructions, the data processor return back the check results to the instructions sender, and after executing the correct instructions, the data processor also can send the execution results by telemetry.

(3) Monitoring the high speed payload data packets. The data processor will check the correction of the received high speed payload data packets, such as whether the logical address is correct, whether the protocol type is correct, etc.

IV. CONCLUSION

This paper introduced that using SpaceWire router on traditional data processor, what simplifies the aerospace electronic systems within the interfaces, provides more flexible interconnection method for the information transmission, and easy to realize redundant backup at link level. The data processor has some intelligent functions such as supporting a variety of ways to configure the parameters of SpaceWire router, and can realize on-orbit maintenance. Increasing the intelligent judgment of instructions receiving and processing, which enhance the reliability and safety of the data processor. The system health monitoring makes it more convenient for work and ground test.

The intelligent processing and maintenance of Aerospace electronic equipment will be an important direction for future development. Especially in deep space exploration and emergency task, intelligent processing and maintenance function is more important. We developed intelligent data processor based on SpaceWire routing has carried on the beneficial attempt and explore in this aspect.

References

[1] ECSS-E-ST-50-12C.SpaceWire-Links,nodes,routers and networks[S]. ESA-ESTEC ,2008.

[2] ECSS-E-ST-50-51C. SpaceWire protocol identification[s]. ESA-ESTEC ,2010.

[3] AT7910E.SpW-10 SpaceWire Router User Manual. ATMEL,2008.

[4] www.nsmc.cma.gov.cn

Real-time Performance Simulation of SpaceWire Router with Polling Arbitration Schemes

Session: Networks and Protocols, Poster Paper

Qiang ZHOU, Lan ZHANG
School of Electronics Information Engineering
BeiHang University
Beijing, China
zhouqiang_ee@buaa.edu.cn, zhang_lan_kuaile@126.com

Hengqing LIN
System Engineering Research Institute
China State Shipbuilding Corporation
Beijing, China
wu342@163.com

***Abstract*—SpaceWire is a standard for on-board satellite networks chosen by the ESA as the basis for future data-handling architectures. Because SpaceWire does not use the bus-shared arbitration, no collision would happen in the link. However, congestion may occur when simultaneous input port data attempt to share one output port of the router. Therefore, the arbitration scheme in SpaceWire router plays an important role on real-time performance. This paper researches on real-time performance simulation of SpaceWire router with three polling arbitration schemes. According to SpaceWire protocol, a SpaceWire simulation model based on OPNET Modeler is proposed, and with the functionality of OPNET Modeler, the network/node/process models are set up. And then three polling arbitration schemes such as FULL polling, EQUAL polling and WEIGHTED polling, are proposed in router node. After that, the validation of the simulation model is presented. And finally a simulation case on typical application is presented. The simulation focuses on the ETE delay of the packets, when aforementioned three polling schemes are implemented. The simulative results indicate that a suitable polling scheme can obtain better real-time performance.**

***Index Terms*—SpaceWire, real-time, simulation, OPNET.**

I. Introduction

SpaceWire is a standard for on-board satellite networks chosen by the ESA as the basis for future data-handling architectures [1]. Since SpaceWire does not use the bus-shared arbitration, no collision would happen in the link. However, congestion may occur when simultaneous input port data attempt to share one output port of the router. Therefore, the arbitration scheme in SpaceWire router plays an important role on real-time performance [2].

This paper researches on real-time performance simulation of SpaceWire router with three polling arbitration schemes such as FULL polling (FP), EQUAL polling (EP) and WEIGHTED polling (WP). According to SpaceWire protocol, a SpaceWire simulation model based on OPNET Modeler [3] is proposed. And then three polling arbitration schemes FP, EP and WP, are proposed in router node. To verify the effectiveness of the aforementioned simulation model, an analysis method is presented. And both simulation and analysis have the consistent results. Finally a simulation case on typical application is presented. The simulation focuses on the end-to-end (ETE) delay of the packets, when aforementioned three polling schemes are implemented. The simulative results indicate that a suitable polling scheme can obtain better real-time performance.

II. Overview of the SpaceWire

A. *SpaceWire Protocol*

SpaceWire is an emerging standard for on-board satellite networks, which uses serial, bi-directional, full-duplex links, with speeds for data high-speed transmission ranging from 2 to 400 Mbps. The newest SpaceWire Standard ECSS-E-50-12C is specified for physical connection to implement high-speed data transmission and data exchange. It comprises six levels as follows: Physical Level, Signal Level, Character Level, Exchange Level, Packet Level, and Network Level.

B. *SpaceWire Network*

SpaceWire network is composed of point-point links, nodes, and routers. SpaceWire node is the source or destination of a packet. SpaceWire router provides a means of routing packets from one node to other nodes. SpaceWire offers unprecedented flexibility in the choice of network topology to match the mission requirements [4]. It involves point to point, chains, rings and trees, even complicated topologies such as multi-dimensional chains and toroid. Figure 1 gives a specific example of SpaceWire network topology architecture [5].

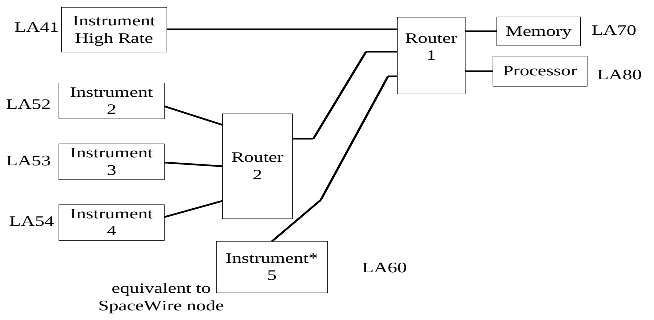


Fig. 1 An example of SpaceWire network topology architecture

C. *Polling Arbitration Schemes in Router*

SpaceWire routers enable packet arriving at input port to be transported to the corresponding output port according to its destination address. When simultaneous input port packets attempt to share one output port of a router, it may cause congestion. Therefore, a scheduling mechanism can be proposed to schedule these input queues.

Because SpaceWire has no specify definition on the scheduling mechanism, in this paper, we propose polling scheduling mechanism, which is simple, and can be used between interfaces of different data rate for the SpaceWire routers. And the arbitration scheme model in router can be seen in figure 2.

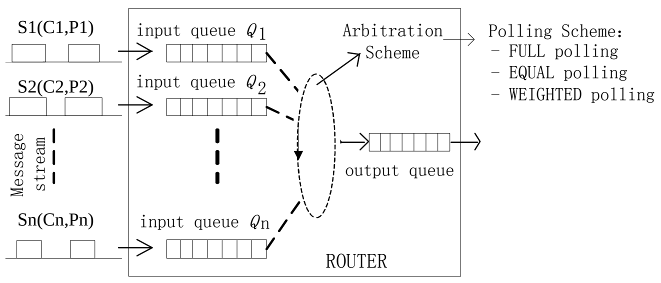


Fig. 2. Arbitration scheme model in router.

III. OPNET Simulation on SpaceWire

Based on the above-mentioned SpaceWire standards and polling arbitration scheme, a simulation model of SpaceWire network can be setup to evaluate the ETE delay of SpaceWire networks. The simulation model involves several aspects such as SpaceWire node model, router model and network model.

A. *SpaceWire Node Model*

SpaceWire node is the source or destination of a packet. It can be a processor, memory unit, sensor or some other units connected to a SpaceWire network. SpaceWire node can be designed in OPNET as shown in fig.3.

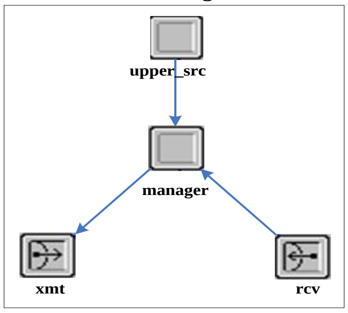


Fig. 3 SpaceWire node model

Where,

Upper_src is a generator module to create packets. It will produce packets at a certain period and certain length, and then send them to the *manager* module.

Manager is a message processing module. It can receive packets generated in the *upper_src* module, and use a user-defined process model to assign destination addresses to the packets or segment them. It can also retrieve packets arriving from the point-to-point receiver. Upon receiving a packet, it uses the same process to calculate the packet's end-to-end delay and write the value to a global statistic. Then, *manager* sends them to the point-to-point transmitter of the node.

XMT / RCV is a point-to-point transmitter/receiver pair for each node, allowing packets to be sent or received from other nodes via attached links. They work as a pair of communication ports of SpaceWire node.

B. *SpaceWire Router Model*

SpaceWire router is designed to connect many nodes together and provides a means of routing packets from one node to other nodes. In OPNET, SpaceWire router is designed with Polling scheduling mechanism and path addressing, as shown in fig. 4.

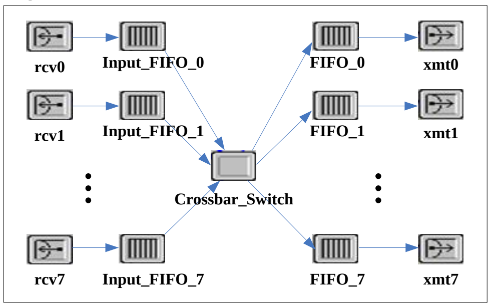


Fig.4 SpaceWire router model

Where,

Rcv0 ~ 7: receivers, which receive packets from other nodes via attached links;

Input_FIFO_0~7: input queues, which provide internal packet queuing facilities to store these packets, and then send them to the Crossbar_Switch.

FIFO_0~7: output queues, which store the packets waiting for being transmitted by the corresponding output port.

Xmt0 ~ 7: transmitters, which send the packets to the adjacent links.

Crossbar_Switch: Processing module, the core of a router, is used to determine the output port to route a packet to by checking its destination address. Additionally, it provides a means of queue scheduling to schedule the input queues, who request for the same output port, in order to solve the competition problem. We propose polling arbitration scheme for this module. And the process model represents the arbitration behavior of output queue in the SpaceWire router. Polling arbitration scheme can be implemented by finite state machine in OPNET.

C. *SpaceWire Network Model*

The SpaceWire Network Model is setup according to the topology described in fig. 1. The network topology consists of seven nodes, two routers and several SpaceWire links. Where, LA41, LA52~ LA54, LA60, LA70, and LA80 are working as nodes. Among them, LA70 is a mass storage, the destination of LA41, LA52 ~ LA54, LA60 and LA80. As a processor, LA80 is used to transmit the packets coming from LA54 to LA70 for storage. LA60 collects packets from sensors, and then sends them to LA70 mass storage.

We assume that the link can work properly, and the buffer capacity is set to infinite.

IV. Verification

In this section, an analysis method is proposed to verify the effectiveness of the aforementioned simulation model.

A. the Scenario Parameters

The scenario parameters are set as in Table I.

TABLE I
SETTING THE SCENARIO PARAMETERS

Symbol	*Quantity*	Value
*T*s	*Simulation time*	100s
S	*Simulation seed*	10
BER	*Link bit error rate*	0
C	*Link bandwidth*	200Mbps
Ti	*Packet transmission starting time*	Exponential distribution (mean value:0.01s)
d_{sw}	Scheduling delay	P(packet length)/C

B. Model Verification

Ref. [6] proposes a method to compute the end-to-end delay of a packet in a SpaceWire network. The method of computation is based on the idea that in a SpaceWire network, the delivery of a packet can be divided into two phases: being transmitted through the SpaceWire links and being routed across the routers. We take the maximum delay for each input link and add those values to the delay for the packet from itself. As the source and destination of the packet do not cause any delays and no collision would happen in the link, the worst case delay for packet occurs in the router.

The maximum ETE delay: the worst case delay occurs when the output port is already in use by another packet, and there is one or more packets coming from other ports may already be waiting for the same output port to become free. The delay is denoted as $d_{\max}$.

$$d_{\max} = \sum_{f_{in}\in S_F} d(f_{in},\mathrm{first(f)}) + 2\sum_{j=1}^{n}\left[\sum_{f_{in}\in U^{j}_{l_{in}}}\max\left(\frac{T_{f_{in}}}{C}\right)+\max\left\{\frac{\mathrm{T_f}}{\mathrm{C}},\max_{f_{in},l_{in}=prev(f,l_j)}\left(\frac{T_{f_{in}}}{C}\right)+\sum_{\substack{f_{in}\in U^{j}_{l_{in}},\\ f_{in}\neq\max(f_{in})}}\left(\frac{T_{f_{in}}}{C}\right)\right\}\right]+\frac{T_f}{C}; \quad (1)$$

Where,

n is the number of routers in the SpaceWire network.

l is the number of the links that the packet follows.

T_f/C is the delay of each link, which is calculated by packet length/link transmission speed.

S_F is the set of packet flows that have the same source as *f*.

fin is the packet flow that use the link l_{in}.

$U^{j}_{l_{in}}$ is the set of packet flows that use the links (except l_j) attached to the router *j*.

l_j is the ordered list of the links the packet flow *f* follows.

Now, we mainly consider about the packet ETE delay impacted by the packet length. For simplicity, it is supposed that there are only LA52 and LA53 sending packets. Then we analyze the impact of the parameter on the packet ETE delay.

TABLE II
COMPARISON ON MAXIMUM ETE DELAY

LA53 packet	LA52 Maximum delay			LA53 Maximum delay		
length (Kbits)	exp.(1) (us)	opnet (us)	deviation (%)	exp.(1) (us)	opnet (us)	deviation (%)
0.1	100.5	100.056	0.442	81	80.682	0.393
4	120	119.906	0.078	120	119.742	0.215
8	200	196.991	1.505	220	217.802	0.999
16	360	359.49	0.142	420	419.283	0.171
30	640	639.468	0.083	770	768.999	0.13
36	760	758.027	0.260	920	918.506	0.162

Table II shows that the deviation between the simulation results and exp.(1) is not more than 1.505%, which indicates that the OPNET simulation results are consistent with the analysis results. This verifies the validity of the simulation model.

V. Simulation And Analysis

In this section we discuss the relationship between the ETE delay and the packet length for specific scenario and specific message flows, when the arbitration scheme in router is set to FP, EP and WP, respectively.

A. Simulation Scenario & Message Parameters

The scenario parameters are setup as the same as in Table I, and the network topology is setup as in section III.C. And the message parameters are shown in Table III. The period of LA53 and LA52 are both set to be fixed on 200us. The packet length of LA52 is 4Kbits. And the packet length of LA53 is set 10K, 15K, 20K, 25K, 30K, 36K(unit: bits) respectively. According to three polling schemes, the corresponding polling factors are defined.

TABLE III
SETTING THE MESSAGE PARAMETERS

Polling arbitration scheme	LA52			LA53		
	Packet Length	Period	Polling factor	Packet Length	Period	Polling factor
FP	4Kbits	200us	4Kbits	variable	200us	= Packet Length
WP	4Kbits	200us	400bits	10 Kbits, 15 Kbits, 20 Kbits, 25 Kbits, 30 Kbits, 36 Kbits	200us	800bits
EP	4Kbits	200us	400bits		200us	400bits

B. Simulation

Figure 5 presents the relationship between the ETE delay of LA52 and the packet length of LA53 when the arbitration scheme in router is set to FP, EP and WP, respectively.

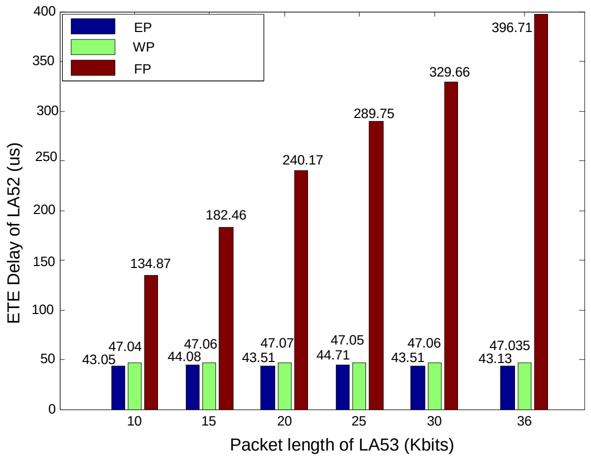


Fig. 5 The ETE delay of LA52 versus the packet length of LA53

Figure 5 shows that: 1) For FP, the ETE delay of LA52 increases sharply from 134.87, 182.46, 240.17, 289.75, 329.66 to 396.71 (unit: us) when the packet length of LA53 increases from 10, 15, 20, 25, 30 to 36 (unit: Kbits). 2) for both EP and WP, the ETE delays of LA52 almost do not change, and the delays of two schemes are all less than 50us. Both of them are significantly less than the ETE delay of FP. 3) FP has the highest ETE delay among the three polling arbitration schemes, while the delay of EP is slightly smaller than that of WP.

Figure 6 presents the relationship between the ETE delay of LA53 and its packet length when the arbitration scheme in router is set to FP, EP and WP, respectively.

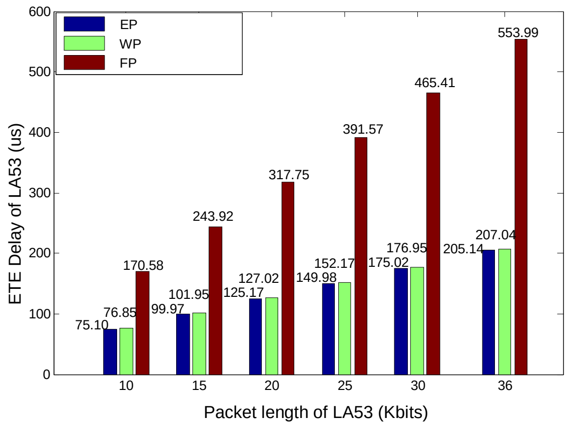


Fig. 6 the ETE delay of LA53 versus the packet length of LA53

Figure 6 shows that: 1) For FP, the ETE delay of LA53 increases sharply from 170.58, 243.92, 317.75, 319.57, 465.41to 553.99 (unit: us) when the packet length of itself increases from 10, 15, 20, 25, 30 to 36 (unit: Kbits). 2) for both EP and WP, the ETE delays of LA53 both increase slowly, while the delays of this two schemes are significantly smaller than that of FP. 3) FP has the highest delay among the three polling arbitration schemes, while the ETE delay of EP is slightly smaller than that of WP.

C. Result

Based on the specific scenario and specific message flows defined in Section V, we can infer the following results. First, the FP has the highest ETE delay among the three polling arbitration schemes, and the delay strongly depends on the packet length. That is to say, the FP has the worst delay performance among the three polling arbitration schemes. Second, the ETE delay performances of the other two schemes are almost similar, although the ETE delay performance of EP is slightly better than that of WP.

VI. Conclusion

In this paper, a real-time performance simulation of SpaceWire router with three polling arbitration schemes such as FULL polling (FP), EQUAL polling (EP) and WEIGHTED polling (WP) is researched. According to SpaceWire protocol, a SpaceWire simulation model based on OPNET Modeler is proposed, and three polling arbitration schemes FP, EP and WP, are proposed in router node. After that, an analysis method is presented to verify the effectiveness of the aforementioned simulation model. Finally a simulation case on typical application is presented, and the simulative results indicate that a suitable polling scheme can obtain better real-time performance.

Acknowledgment

This work was supported by Beijing Natural Science Foundation (4133089).

References

[1] ECSS-E-50-12-C. SpaceWire Engineering: SpaceWire-Links, node, routers and networks ESA-ESTEC. November 2008.

[2] Q. ZHOU, H. XIN, Y. SHI. Realtime performance of arbitration scheme for SpaceWire router, Proceedings of the 4th International SpaceWire Conference, SpaceWire 2011, University of Dundee, 2011,pp:23-26.

[3] Huawei Shi, Nianjun Zhang, Tongguang lv. Analysis of OPNET simulation technology and its application[J].Computer Engineering & Design,2006, (17):3309~3310.

[4] S. Parkes, P. Armbruster. SpaceWire: Spacecraft Onboard Data-handling Network. Acta Astronautica, 2010,66(1-2): 88-95

[5] SpaceNet-SpaceWire-RT Initial Protocol Definition. Space Technology Centre School of Computing University of Dundee, DD1 4HN Scotland, UK. October 2008.

[6] T. FERRANDIZ, F. FRANCES, C. FRABOUL. A method of computation for worst-case delay analysis on SpaceWire networks [J]. Institute of Electrical and Electronics Engineers (IEEE), 2009: SIES '09 ; 8 - 10 July 2009, Ecole Polytechnique Federale de Lausanne, Switzerland. IEEE, Piscataway, pp. 19-27. ISBN 978-1-4244-4109-9.

The General Situation of SpaceWire Research in China

Missions and Applications, Poster Paper

Chen Xiaomin , Guo Lin, Sun Huixian
National Space Science Center, Chinese Academy of Sciences
Beijing, China
E-mail: *chenxm@cssar.ac.cn*, *guolincug@yahoo.com.cn*, *shxian@cssar.ac.cn*

***Abstract*—SpaceWire, an on-orbit high-speed network, has lots of useful characteristics which are high speed, full-duplex, convenience of setting up, flexible topology and open-protocol. Now SpaceWire has become the new generation of on-orbit data bus recommended by ESA and NASA, and was successfully applied**

At present, the space industry in China is developing at high-speed, and a number of satellites are planning or preparing to launch. SpaceWire bus with its excellent performance has been paid more and more attention and studied by Chinese space scientists. This paper reviews the current research situation of SpaceWire Technology in China, introduces the possible application scene of SpaceWire in China spacecraft, and analyses the requirement of the new generation high-speed bus in Chinese space missions and the applicable situation of the SpaceWire. In the end, this paper summarizes the problems which were found in Chinese researchers work on the SpaceWire, and concludes their advice and wishes.

***key word*: SpaceWire, China, Spacecraft Electronics.**

I. SpaceWire State of the Art in China

SpaceWire as an on-orbit high speed network, provides a unified high speed data-handling infrastructure for connecting together sensors, processing elements, mass-memory units, downlink telemetry subsystems and electronic ground support equipment (EGSE), and has lots of useful characteristics, such as high speed, reliability, low power consumption, structure simple etc.[1] SpaceWire protocol is open and flexible to be compliant with possible future higher demanding needs, has broad prospects for development.

At present, more than 10 research institutes are carrying out study on SpaceWire technology, which principally includes *National Space Science Center , Chinese Academy of Sciences(NSSC,CAS), Beijing Institute of Control Engineering(BICE), Beijing Institute of Space Mechanics and Electricity(BISME), Harbin Institute of Technology(HIT), Capital Normal University(CNU), Xi'an Microelectronics Technology Institute(XMTI) etc*. Through statistics on Chinese engineering and technology literatures, as recently as five years, the topics with regard to SpaceWire are more than 40 articles, which involve the field of study as shown in Table 1.

TABLE I. STATISTICS ON CHINESE SPACEWIRE ARTICLES

Fields	**Quantities**
Summarizes	4
Components	13
Test & Verification	3
Upper layer protocols	1
Onboard Equipments	2

Is obvious from the previous table, the research on SpaceWire is still at primary scenario in China, which is mainly reflected in the following 2 aspects.

First, a lot of work are still focused on summarizes and introduction, development on elementary components (such as Codec, Router), prototype design on point-to-point transmission application, there are wide gaps compared to advanced research and application, such as network application which interconnects with routers, hardware and software design on SOC.

Second, SpaceWire hardware structure is relatively simple to Switched Ethernet and IEEE-1394, the current FPGA design method has been very mature. Many research institutes choose the Codec IP and Router IP for research objectives, their production levels are different, but have the widespread problem is lack of succession and full verification, are far away from on-orbit space application.

According to the situation which we grasp, currently, outstanding SpaceWire productions in China are as shown in Table 2.

TABLE II. REPRESENTATIVE SPACEWIRE PRODUCTS IN CHINA

SpaceWire productions	**research institutions**	**Features Description**
SpaceWire Codec IP	*NSSC,CAS*	FPGA: A3P1000,Std; Function: Compliance with ECSS-E-ST-50-12C, configurable low-power consumption mode; Performance: up to 200Mbits/s
	BISME	FPGA: Virtex-5 LX110T; Function: Compliance with ECSS-E-ST-50-12C;

		Performance: up to 300Mbits/s[2]
SpaceWire Router IP	*NSSC,CAS*	FPGA: A3P1000, Std; Function: 8-ports Router, Compliance with ECSS-E-ST-50-12C, Priority based, round-robin arbitration ， Group adaptive routing; Performance: up to 200Mbits/s, switching latency is 100~125ns
PCI-SpaceWire interface card	*NSSC,CAS*	FPGA: APA600; Function: transmission of data and time-codes, monitor and record status, link fault Injection； Performance: up to 100Mbits/s
	HIT	Function: transmission data; Performance: up to 100Mbits/s[3]
SpaceWire communication prototype	*NSSC,CAS*	ASIC: AT7911e; Function: Half-duplex communication, transmission of scientific data and control commands; Performance: 140Mbits/s

II. Main Scrape of SpaceWire Application in China

SpaceWire has been used in more than 30 space missions. With the development of comprehensive national strength, a number of satellites are planning or preparing to launch in China. But so far, SpaceWire is not yet used in any spacecraft in China. Toward this situation, we will attempt to investigate and analyze its reason.

In the present limited demand, the advantage of SpaceWire is not obvious. In Chinese spacecraft, with the widespread use of the network structure is high reliable 1553B or dual-CAN as control bus. 1553B is very mature and almost all satellites are widely used in China. So it is the first choice of the on-orbit data network with higher prestige. The CAN often gets the favour of small satellites. China is also often used RS-422 at high data rate transmission application scene. When network needs to further improve the data rate, First choice would be LVDS connection to constitute several point-to-point links, but not be SpaceWire. Investigate its reason, currently in Chinese spacecraft, the demand of high data rate is only limited to a few imaging science instruments, the data rate of whole network is not high. Therefore, individual instruments adopt special “point-to-point” LVDS connection already can meet the mission, and the need of standard upper layer protocol is greatly weakened.

Costly and complex design on circuit board. According to SpaceWire user guide as an example, Each SpaceWire node is composed of one AT7911e, two dual-port RAMs and one processor. Although SpaceWire has higher reliability than LVDS, it has higher cost and complexity. The design fees of SpaceWire interface might occupy a very high proportion in whole funds of instruments, which let to the designer difficult to choose SpaceWire. In addition to expensive besides, SpaceWire interface also can lead to volume, weight and power consumption is higher, also can take up the whole instrument more resources.

The standard maturity is relatively low, especially for control bus in spacecraft. Compared with 1553B, IEEE-1394 and switched Ethernet, ECSS-E-ST-50-12C only supports the data link layer of OSI reference model. Despite the RMAP transport layer protocol has been standardized, continuously put forward some ideas and design of upper layer protocol, such as SpaceWire-RT, SpaceWire-D, SpaceWire-PnP, SpaceFibre etc, but in addition to RMAP, these protocols are still in draft stage, lack of documents and supported chips. In addition, the chips integrated RMAP are rarely, therefore, Although Chinese engineers want to solve practical engineering problems of reliability and real-time of transmission, the protocol status greatly restricted the solution to the problem, leading engineers reluctant to use SpaceWire.

III. The Future of SpaceWire in China

Currently in china, for promoting the SpaceWire application, we believe that there are two ways. On the one hand, through the integration of protocol control IP and interface IP, to develop design of SOC, which can save cost and power, size and weight of circuit board, improve the availability of SpaceWire. The other hand, SpaceWire as an open protocol stack, designer and researcher should is not only as technology trackers and imitators, but also actively involve in the development of SpaceWire upper layer protocols, accelerate the standardization of various upper layer protocols in draft stage.

SpaceWire has been successfully applied in many spacecrafts from ESA, NASA and JAXA, which indicates its performance and functionality obtained international recognition of the major space research institutions, represents the development trends of on-orbit data network. We have abundant reason to believe that SpaceWire application in Chinese spacecraft will be achieved in the near future.

References

[1] ECSS, “SpaceWire-links, nodes, routers and networks”, ECSS-E-ST-50-12C, Junly 2008.

[2] Liu Tao, Huang Wei, Pan Weijun, “Design and Verification of Spacewire IP Core”, Spacecraft Recovery and Remote Sensing, Vol.1 NO.32, pp.51-58, 2011.

[3] Qiao Liyan, Chen Libin, Peng Xiyuan, “Design of spacewire-PCI correspondence card based on IP core”, JOURNAL OF ELECTRONIC MEASUREMENT AND INSTRUMENT, Vol.24 No.10, pp.918-923, 2010.

A Low-power SpaceWire Codec IP Core

SpaceWire Components, Poster Paper

Guo Lin, Sun Huixian , Chen Xiaomin
National Space Science Center, Chinese Academy of Sciences
Beijing, China
E-mail: guolincug@yahoo.com.cn, shxian@cssar.ac.cn, chenxm@cssar.ac.cn

***Abstract*—SpaceWire is an onboard data-handling network for spacecraft which offers high- speed, low power, simplicity, low cost, and architectural flexibility. SpaceWire coder/decoder (Codec) which uses the DS encoding method to serialize SpaceWire characters for communication over the SpaceWire link. Standard SpaceWire Codec can alter data signalling rate (DSR) in the Run state, but the data signalling rate is not adjusted according to the type of transmission character, which will cause unnecessary Power loss.**

We designed and implemented a low-power SpaceWire Codec IP core, it satisfies the ECSS-E-ST-50-12C protocol, and can work through the configuration registers in the standard and low-power modes. In low-power mode, When the transmit interface does not have N-Chars and Time-Codes to send, the Codec will automatically adjust the data signalling rate to 10Mbps, thereby reducing the operating power consumption.

The IP core is implemented in an Actel A3P1000 approximately 4% of the logic resources and is up to 200Mbits/s in Std grade of FPGA, the DSR conversion delay is less than 100ns. By Star-Dundee's SpaceWire PCI-2 and SpaceWire Link analyser to validate, the IP core has good compatibility with standard SpaceWire equipment, and can reduce consumption by about 56%.

***key word*: SpaceWire, Low-power, Codec, IP.**

I. INTRODUCTION

Since 2003, CSSAR, CAS has carried out the widespread and in-depth research on ESA standard ECSS-E-ST-50-12C. Currently who has been independently developed products including multi-port SpaceWire communication terminal based on AT7911e, Codec IP core and Router IP core, SpaceWire-PCI interface card, all have obtained successful application.

This paper introduces a configurable low-power consumption Codec IP core improved from current IP, it provides and achieves a low-power consumption design in run state of codec. The final experimental results show the power consumption has obvious reduced, meanwhile, the IP has good compatibility with standard protocol terminal and up to 200Mbits/s data signalling rate.

II. THE ANALYSIS OF EXISTING CODEC POWER CONSUMPTION

According as ECSS-E-ST-50-12C, the standard Codec IP core or ASIC can operate at any data signalling rate between the minimum data signalling rate and the maximum possible data signalling rate. [1] The delivered character is separated into two types: link-characters (L-Char) and normal-characters (N-Char). The recover clock from Data-strobe signal through XOR circuit. Data transfer process shown in Figure 1.

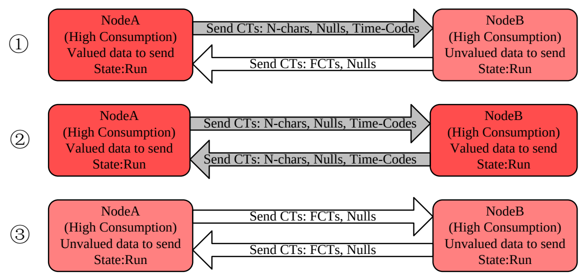

①
NodeA (High Consumption) Valued data to send State:Run
Send CTs: N-chars, Nulls, Time-Codes
Send CTs: FCTs, Nulls
NodeB (High Consumption) Unvalued data to send State:Run
②
NodeA (High Consumption) Valued data to send State:Run
Send CTs: N-chars, Nulls, Time-Codes
Send CTs: N-chars, Nulls, Time-Codes
NodeB (High Consumption) Valued data to send State:Run
③
NodeA (High Consumption) Unvalued data to send State:Run
Send CTs: FCTs, Nulls
Send CTs: FCTs, Nulls
NodeB (High Consumption) Unvalued data to send State:Run

Fig. 1. Data transfer scenes between standard SpaceWire nodes

Figure1 shows three data transfer scenes were simplex, full-duplex and idle (do not send N-Chars and Time-Codes). Since the data signalling rate of node A and node B are pre-configured, therefore in idle scene, nodes still maintained the same data signalling rate, which will cause unnecessary Power loss.

In order to reduce the power loss on idle scene, SpaceWire router AT7910e of Atmel Corporation provides two special modes to save power, they are request mode and silence mode. [2] It can automatically start on request mode if the source port attempts to send data, and disable on silence mode when it no longer has any data to transfer. The work scene as shown in figure 2, but through further analyzes shows that this approach has some drawbacks:

- When the source port has data to be sent, it needs at least 20us for link initialization, is not suitable for real-time data and time-codes transmission applications;

- When no N-chars have been transmitted after timeout period, router can disable the link to save power, but it is only suitable for simplex communication, has significant limitations.

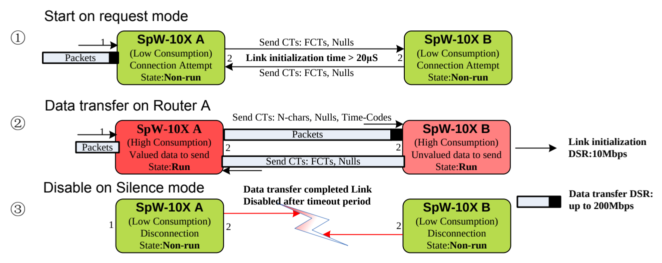


Fig. 2. AT7910e's low-power loss strategy

III. A Design Method of Low-power Consumption Codec

The power consumption and operating frequency have proportional relationship. According to experimental results indicated that data signalling rate for 10Mbits/s (idle) and 200Mbits/s (full-duplex), the power consumption of single Codec was higher than about 4mW and 70mW, compared with standby power consumption. Thus, if no N-chars or time-codes to send, the codec maintain links through slow data signalling rate(i.e.10Mbits/s) to transmit Null and FCT, which can also save power obviously, meanwhile, solve two drawbacks of AT7910e on request mode and silence mode, one is high link initialization latency expenses, the other is not suitable for the duplex communication. The data transfer scenes shown in Figure3.

By the scene 2 of figure 3, when node A sends data in simplex communication mode, node B is slow data signalling rate to send FCT or Null. The FCTs offered by node B need to ensure non-stop data transmission from node A, thus, data signalling rate on link both sides must satisfy the formula 1. Since by the formula, when data signalling rate of node B is 10Mbits/s, Maximum data signalling rate of node A is 200Mbits/s.

$$(8/DSR_{high})\times 10 \geq 4/DSR_{low} \qquad (1)$$

In low-power consumption Codec IP core, the output DS signals generated by three modules, Inc, invalid data generation module (Unvalued data, Ud), data signalling rate conversion module (DSR conversion, Dc) and valid data generation module (valued data, Vd). In Ud module, the input clock is 10MHz, DS signals generated by single data rate (SDR) encoding method, it only produces Nulls and FCTs characters, may realize the link initialization. In Dc module, maximum input clock is 100MHz and has same phase with Ud's, DS signals encoding method also is SDR, it can complete remaining bit-stream from Ud's to reduce data signalling rate switching delay. In Vd module, the input clock is the same as Dc's, DS signals encoding method is dual data rate (DDR), it can produce all the SpaceWire characters. Figure 4 is DS

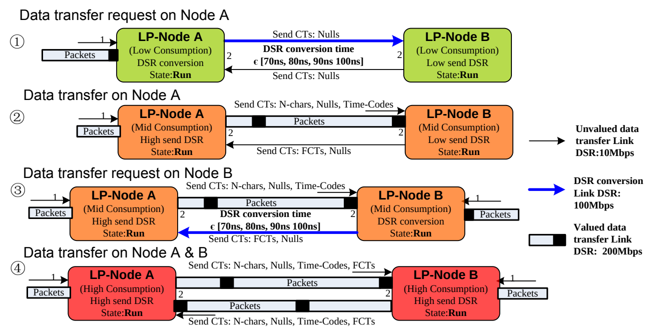


Fig. 3. Data transfer scenes of low-power Codec

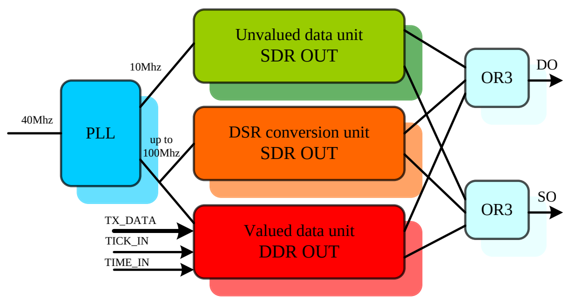


Fig. 4. DS signals generator block diagram

IV. Implementation and Result

The IP core is implemented for Actel A3P1000 with std grade. The main features and performance are as follows:

- The logical resource approximately is 4%, communication data rates up to 200Mbits / s;
- The IP core can be configured on standard mode and low-power consumption mode, all have good compatibility with standard SpaceWire terminal;
- When input clock of Vd and Dc is 100MHz, the data signalling rate conversion delay is not higher than 100ns, and through testing all 120 kinds of switching process, the result is correct, Figure 5 is simulation waveform of a switching process;
- Compared with the existing Codec chips, the power consumption improve can amount to 56%.

V. Conclusion

The low-power codec can adjust data signalling rate according to the type of transmission characters, when no valid data to be sent, it can reduce data signalling rate to save power. Compared with AT7910e, it has better flexibility and performance, can be used for full-duplex communication especially. Through experiments, its compatibility and performance parameters are good. Is foreseeable that, if SpaceWire router IP core uses it, will lead to more substantial power improvement.

REFERENCES

[1] ECSS, “SpaceWire-links, nodes, routers and networks”, ECSS-E-ST-50-12C, Junly 2008.

[2] Atmel, “AT7910E SpW-10X SpaceWire Router”, April 2008

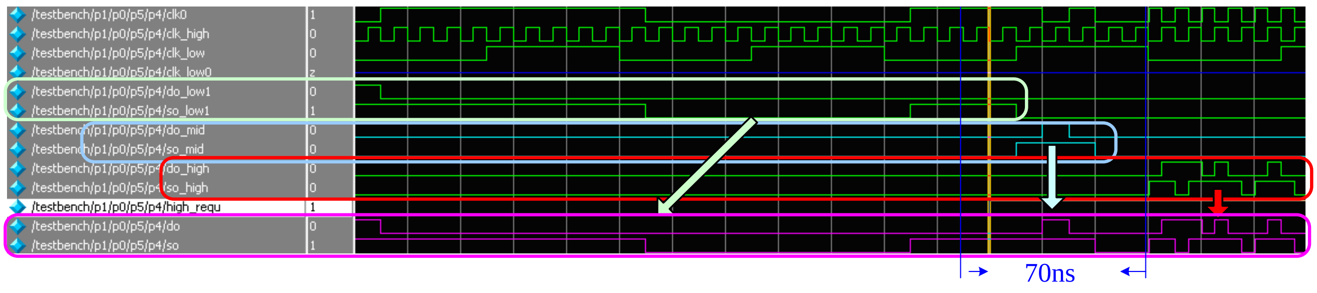
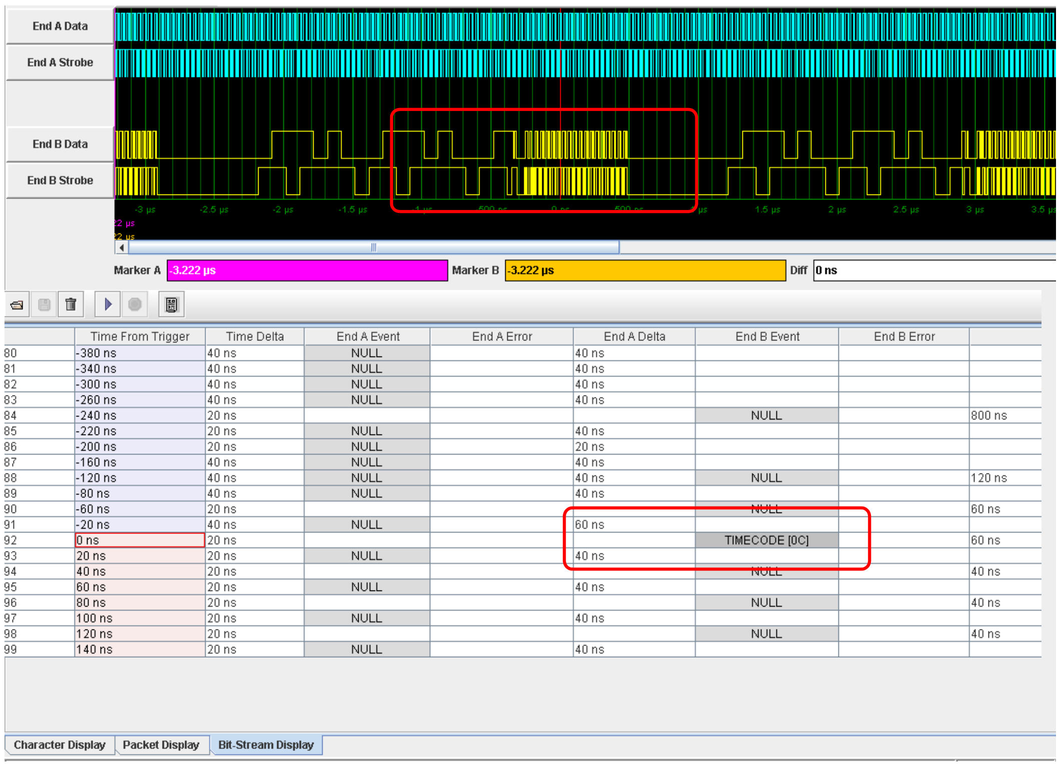


Fig. 5. Simulation waveform of a switching process



	Time From Trigger	Time Delta	End A Event	End A Error	End A Delta	End B Event	End B Error	
80	-380 ns	40 ns	NULL		40 ns			
81	-340 ns	40 ns	NULL		40 ns			
82	-300 ns	40 ns	NULL		40 ns			
83	-260 ns	40 ns	NULL		40 ns			
84	-240 ns	20 ns				NULL		800 ns
85	-220 ns	20 ns	NULL		40 ns			
86	-200 ns	20 ns	NULL		20 ns			
87	-160 ns	40 ns	NULL		40 ns			
88	-120 ns	40 ns	NULL		40 ns	NULL		120 ns
89	-80 ns	40 ns	NULL		40 ns			
90	-60 ns	20 ns				NULL		60 ns
91	-20 ns	40 ns	NULL		60 ns			
92	0 ns	20 ns				TIMECODE [0C]		60 ns
93	20 ns	20 ns	NULL		40 ns			
94	40 ns	20 ns				NULL		40 ns
95	60 ns	20 ns	NULL		40 ns			
96	80 ns	20 ns				NULL		40 ns
97	100 ns	20 ns	NULL		40 ns			
98	120 ns	20 ns				NULL		40 ns
99	140 ns	20 ns	NULL		40 ns			

Fig. 6. Time-codes transmission on low-power consumption mode (End A:PCI-2, End B: Low-power -Codec)

FORMAL VERIFICATION FOR SPACEWIRE DECODING BY APPLING THEOREM PROVING

Session: SpaceWire test and verification

Poster Paper

Yupeng Zhang, Zhiping Shi, Yong Guan, Xiaojuan Li

Beijing Key Laboratory of Electronic System Reliability Technology, Capital Normal University, Beijing, 100048, China

namoweiguang@126.com, shizhiping@gmail.com, gxy169@sina.com

Jie Zhang

College of Information Science & Technology, Beijing University of Chemical Technology, Beijing, 100029, China

jzhang@mail.buct.edu.com

ABSTRACT:

In SpaceWire standard , decoding circuit belongs to the receiver and it corresponds to the Data-Strobe encoding circuit. Based on higher-order logic theorem prover HOL4, this paper applies theorem proving, one of the formal methods, to verify the SpaceWire decoding circuit. And the paper focuses on the properties of DataValid in the decoding circuit. Firstly, this paper extracts the relevant properties of DataValid on the basis of SpaceWire standards. These properties are described in higher-order logic. Then analyze the VHDL design codes of the circuit and model it logically according to the realized function of the codes. Finally with the aid of HOL4 it is validated that the model of the circuit design can satisfy the properties faithfully.

KEY WORDS:

SpaceWire standard; formal verification; theorem proving; HOL4

I. INTRODUCTION

In recent years, SpaceWire protocol has got wide attention and rapid development in theory and technology applications. It can be extensively found in aerospace domain，open field, mines, nuclear power station and other harsh or dangerous environment. Because of SpaceWire's vital importance, any tiny errors in system design are likely to produce huge economic losses and casualties. Therefore it is very necessary to verify the design of SpaceWire. However, the traditional methods, simulation and testing are based on test cases which are impossible to cover all the cases for huge and complex systems [1] [2]. Formal verification which is based on some specifications or attributes uses mathematical methods to prove the correctness or incorrectness of system. And the verification is complete for the properties to be verified. This paper applies theorem proving, one method of formal verification, to analyze the behavior of system , model the system logically and then prove the properties of model mathematically with the aid of higher-order logic proof tool HOL4.

Decoding circuit is the receiver's key circuit in SpaceWire standard. It is related to the Data-Strobe (DS) encoding circuit in transmitter [3]. The key point of the verification is formal modeling.

The paper is organized as follows: Chapter Ⅱ introduces the method of formal verification in details which includes decoding specification and implementation. Then the result is shown in Chapter Ⅲ. The paper comes to an end with conclusions and future work in Chapter Ⅳ.

II. METHOD

In order to ensure the design of *DataValid* meets the requirement of SpaceWire standard, the paper converts the properties of *DataValid* into logical description by using higher-order logic language. The process is normally called decoding specification. Then VHDL design codes are abstractly modeled by corresponding logical predicates which can describe the realized functions of the codes [4]. And the model can explicitly shows the characteristic of clock, the order of code execution and the behavior. Then the paper draws logical diagrams which can indicate the relationship among predicates clearly. This model is called decoding implementation.

The paper mainly takes advantage of goal-guiding method which is one obvious characteristics of HOL4. This approach is known as strictly logical reasoning [5]. In addition, relevant tactics and axioms obtained from HOL4 are used to verify whether the implementation satisfied the requirement of specification.

A. DECODING SPECIFICATION

Specifications indicate the temporal properties or functions of the system to be verified. The receiver firstly needs to decode the data and the decoding process is finished in decoding circuit. This paper focuses on the specifications of *DataValid* in decoding circuit.

DataValid is an important signal. It can check whether the received link signal is effective and many operations must be based on its signal value to continue downward [2]. Decoding circuit obtains the signal *DataValid* through logical operations of the signal *DataIn* and *StrobeIn*. When the value of *DataValid* is high, the received data is valid. On the contrary, the received data is invalid.

Property 1:

As long as the *reset* signal is *T*, the output signal *DataValid* is *F*. “*T*” means high level and “*F* ” means low level.

$\forall$t. reset t ==>（datavalid t = F） (1)

“$\forall$t ” means for all t ; “==> ”means implication.

Property 2:

There are five clock delay in the output. If *reset* signal is *T* at time *t+5* or *t+4* or *t+3*, the output signal *DataValid* is *F. If reset* signal is *F* at time *t+5* or *t+4* or *t+3 and is T* at time *t+2,* the output signal *DataValid* is equal to the logic XOR gate of the signal *DataIn* and *StrobeIn* at time *t+2*. At other time *DataValid* is always equal to the logic XOR gate of *DataIn* and *StrobeIn* each *at* at time *t+2* and at time *t+1.*

$\forall$t. datavalid(t+5) =

If reset(t+5) $\vee$ reset(t+4) $\vee$ reset(t+3) then F

else if reset(t+2) then

XOR (d(t+2)) (s (t+2))

else

XOR4 (d(t+1)) (d(t+2)) (s(t+1)) (s(t+2)) (2)

The definitions of predicates “XOR” and “XOR4” in property 2 is shown as follows.

Definition 1:

|- $\forall$a b. XOR a b = ¬a $\wedge$ b $\vee$ a $\wedge$¬b (3)

The predicate “XOR*”* is used to formalize the logic XOR gate of two signals.

Definition 2:

|- $\forall$a b c d. XOR4 a b c d =
(¬a$\wedge$ b$\wedge$c$\wedge$d) $\vee$ (a $\wedge$¬b$\wedge$c$\wedge$d) $\vee$ (a$\wedge$b$\wedge$ ¬c$\wedge$d) $\vee$ (a$\wedge$b$\wedge$c$\wedge$¬d) $\vee$
(¬a$\wedge$ ¬b$\wedge$ ¬c$\wedge$d) $\vee$ (¬a$\wedge$¬b$\wedge$c$\wedge$¬d) $\vee$ (¬a$\wedge$b$\wedge$ ¬c$\wedge$ ¬d) $\vee$
(a$\wedge$¬b$\wedge$¬c$\wedge$¬d) (4)

The predicate “XOR4” formalizes the logic XOR gate of four signals. “¬” means negation.

According to the combination of property 1 and property 2 the paper gets the formal descriptions of decoding specifications.

B. DECODING IMPLEMENTATION

According to the design codes of decoding circuit in SpaceWire, this paper draws a related structure diagram and then simplifies it which the output only has signal *DataValid.* The structure diagram is shown in Fig 1:

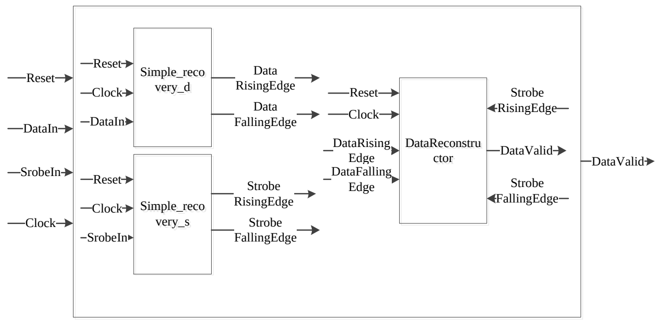


Fig 1: The structure diagram

Definition 3:

|- $\forall$out. ONE out = $\forall$t. out t = T (5)

The predicate “ONE ” means its output is always true. It is used to formalize the operation in which the output is 1 at any time.

Definition 4:

|- $\forall$inp out. NOT (inp,out) = $\forall$t. out t = ¬inp (6)

The predicate “NOT” means its output is the negation of input.

Definition 5:

|- $\forall$sw a b out. MUX (sw,a,b,out) = $\forall$t. out t =

(if sw t then a t else b t) (7)

The predicate “MUX” formalizes the case statement “IF…THEN… ELSE…”.

Definition 6:

|-$\forall$inp out. REG (inp,out) = $\forall$t. out t =

(if t = 0 then F else inp (t – 1)) (8)

The predicate “REG” has the function of register which has one clock delay.

Definition 7:

|- $\forall$rst a out. DFF (rst,c,out) = $\exists$x a b. REG (c,x) $\wedge$ MUX (rst,b,x,d) $\wedge$ ONE a $\wedge$ NOT(a,b) (9)

The predicate “DFF” has the function of asynchronous D flip-flop. It is used to formalize the component “FDC” which is called in design code of *simple_recovery* in fig 1. The definition of “DFF” is obtained from the combination of definition3, 6 and 7. The main code of “FDC” is shown in Fig2.

```
ARCHITECTURE behavior OF FDC IS
BEGIN
    PROCESS (C,CLR)
    BEGIN
      IF  (CLR='1') THEN
         Q<='0';
      ELSIF (C'EVENT AND C='1') THEN
         Q<=D;
      END IF;
    END PROCESS;
END behavior:
```

Fig 2: The main code of FDC

Definition 8:

|- ∀a b out. AND2B (a,b,out) = ∀t. out t = ¬a t /\ b t (10)

The predicate “AND2B” formalizes the component “AND2B1” which is also called in design code of *simple_recovery* in fig 1. The main code of “AND2B1” is shown in Fig3.

```
ARCHITECTURE behavior OF AND2B1 IS
BEGIN
O <= (NOT I0) AND I1;
END behavior;
```

Fig3: the main code of AND2B1

This paper contacts the definitions of predicates above with logical operation “AND”. The signals in design codes are represented by existential quantifiers. Then the formal model of *simple_recovery_d* in Fig1 can be built. For the purpose of simplicity *d*, *reset*, *d1*, *d2* are treated as replacement for *DataIn, Reset,DataRisingEdge,DataFallingEdge* respectively. And the paper uses *e0, d0, x0, x1, x2, x3, x4, x5, a1, a2* as substitutes for the signals in design codes. Based on the formal model of *simple_recovery_d* the paper uses line instead of logical operation “AND”, then draws a logical diagram which can clearly reflect the relationship among the predicates. The model of *simple_recovery_s* in Fig1 can also be got by the same way. The logical diagram of *simple_recovery_d* is shown in Fig 4.

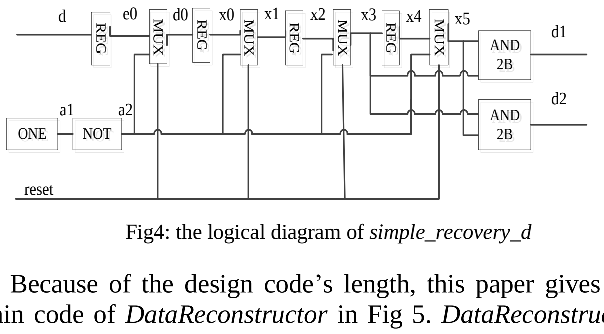


Fig4: the logical diagram of *simple_recovery_d*

Because of the design code’s length, this paper gives the main code of *DataReconstructor* in Fig 5. *DataReconstructor* is one part of decoding circuit as Fig 1 shows. Its output is the signal *DataValid* .

```
PROCESS (Reset, Clock)
      VARIABLE NoX : STD_LOGIC;
   BEGIN
      IF  Reset = '1' THEN
         DataValid               <= '0';
         DataRisingEdge_delayed    <= '0';
         DataFallingEdge_delayed   <= '0';
         StrobeRisingEdge_delayed  <= '0';
         StrobeFallingEdge_delayed <= '0';
      ELSIF rising_edge(Clock) THEN
         DataRisingEdge_delayed    <=
                          DataRisingEdge;
         DataFallingEdge_delayed   <=
                          DataFallingEdge;
         StrobeRisingEdge_delayed  <=
                          StrobeRisingEdge;
         StrobeFallingEdge_delayed <=
                          StrobeFallingEdge;
         NoX :=
         (DataRisingEdge    AND    NOT
         DataRisingEdge_delayed)    XOR
         (DataFallingEdge    AND   NOT
         DataFallingEdge_delayed)    XOR
         (StrobeRisingEdge   AND   NOT
         StrobeRisingEdge_delayed)  XOR
         (StrobeFallingEdge  AND   NOT
         StrobeFallingEdge_delayed);
         DataValid <= NoX;
       END IF;
```

Definition 9:

|-∀a b out. AND (a,b,out) = ∀t. out t = a t /\ b t (11)

The predicate “ AND” indicates the logic AND gate.

Definition 10:

|- ∀a b out. XORING (a,b,out) =∀t. out t = XOR (a t) (b t) (12)

The predicate “XORING” is equal to the function of logic XOR gate and corresponds to Definition 1.

Definition 11:

|- ∀a b c d out. XOR4ING(a,b,c,d,out) = ∀t. out t =

XOR4 (a t) (b t) (c t) (d t) (13)

The predicate “XOR4ING” corresponds to the Definition 2.

After finishing the definition 9, 10 and 11, the formal model of *DataReconstructor* can be establised with the similar method as Fig 4 shows. At the same time *s1*, *s2*, *d3*, *d4*, *s3*, *s4*, *datavalid* are used to replace *StrobeRisingEdge,StrobeFalling-Edge, DataRisingEdge_delayed, DataFallingEdge_delayed*， *StrobeRisingEdge_delayed,StrobeFallingEdge_delayed,DataValid* respectively. And the paper substitute *a3, a4, a5, a6, m1, m2, m3, m4, n1, n2, n3, n4, n5* for the signals in the design codes. The logical diagram of *DataReconstructor* is shown in Fig 5.

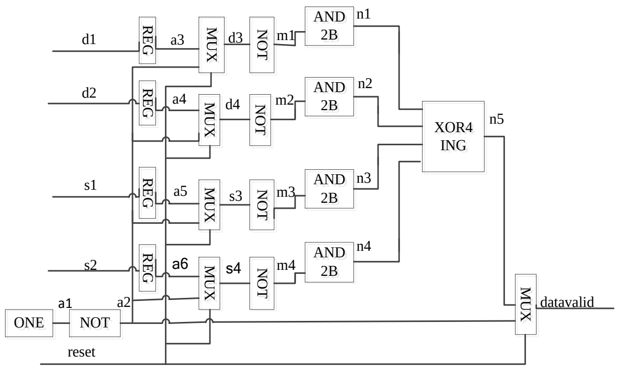


Fig5: the logical diagram of *DataReconstructor*

Now the formal model of decoding implementation can be obtained by combining the model of *simple_recovery_d , simple_recovery_s and DataReconstructor*.

III. RESULT

After having finished the formalization of specification and implementation, the paper applies goal-oriented reasoning in HOL4 and the proof is interactive. The initial goal which implementation can imply specification is set.

According to the definitions, the initial goal is divided into several subgoals and then the subgoals are proved in turn. If every subgoal has been proved, the initial goal is proved. In the meantime, the related tactics and tacticals in HOL4 are also used. In this paper, the result shows the initial goal is proved.

IV. CONCLUSIONS AND FUTURE WORK

The paper applies theorem proving to verify that the design codes of decoding circuit can satisfy the specification of SpaceWire standards. Besides, the formal model based on the function of VHDL design codes is effective and it also can be used in other hardware design verification. Furthermore, this method can help the designer to find errors in the early design stage and has important practical significance.

Theorem proving emphasizes man-machine interaction which is different from model checking and may have the problem of heavy workload. In the future, the research will use model checking to prove sub-module. Then gather and process the results with the aid of theorem proving. That is to say the research will focus on combing theorem proving with model checking to verify other key circuits of SpaceWire standard.

ACKNOWLEDGMENT

We appreciate Prof. Jin Shengzhen for his helpful suggestions. Thanks are also due to all members of our group. This work was supported by the International Cooperation Program on Science and Technology (2010DFB10930, 2011DFG13000), the National Natural Science Foundation of China (60873006, 61070049, 61170304, 61104035); the Natural Science Foundation of the City of Beijing (4122017), the S&R Key Program of the Beijing Municipal Education Commission (KZ201210028036).

REFERENCE

[1] Han Jun-gang, Du Hui-min. Digital Hardware Formal Verification [M]. Beijing: Peking University press, 2001.

[2] William K. Lam, Sun Microsystems. Hardware design verification: simulation and formal method-based approaches [M]. Prentice Hall Professional Technical Reference, 2005.

[3] ECSS Standard ECSS-E-ST-50-12C, SpaceWire-Links, Nodes, Routers and Networks [S].

[4] Li Li-ming, Liu Li-ya, Guan Yong, et al. A formal method for verification the implementation of SpW data-strobe encoding by applying theorem proving [C]. SpaceWire-2010 Proceedings of the 3rd International SpaceWire Conference .St. Petersburg, 2010 -6-22.

[5] The HOL system TUTORIAL (For HOL Kananaskis-7), 18th. March. 2012.

Latency Jitter Estimation and Control in SpaceWire

Session: SpaceWire Networks and Protocols, poster paper

Nadezhda Matveeva, Elena Suvorova
Saint-Petersburg State University of Aerospace Instrumentation
SUAI
Saint-Petersburg, Russian Federation

n.matveeva88@gmail.com, wildcat15@yandex.ru

Abstract — The SpaceWire networks are broadly used in embedded and onboard systems, where low latency and jitter are critical. These characteristics are also important for networks with SpaceFibre channels. Jitter increases buffer sizes, which memory volume increases energy consumption, system cost and weight. It is very important in space applications.

High jitter presence complicates systems, where a synchronization of different blocks or packets ordering is required. Examples of such systems are systems of image processing, video stream from cameras processing and information from groups of sensors collecting.

Jitter reduction in systems of a stream broadcasting or image processing allows to reduce receiver buffer size and simplify process of frame restoring. Many factors influence at jitter: network structure, data processing algorithms of switches, data packet length, presence and characteristics of other data streams and etc.

In this paper we show dependencies between network parameters and jitter value. We present a set of rules that allows to minimize jitter value by adjustable network parameters changing. Jitter minimization allows to calculate buffer sizes more accurately in network and network components development.

Index Terms — SpaceWire, jitter, delay, jitter estimation.

I. Introduction

Delay and jitter are very important in the transmission of data of different types. For example, it is video and audio information. These parameters are critical for synchronization of various system components. Information about jitter and delay ranges allows to optimize communication system design, to calculate size of buffers and do not use extra buffer space. Buffer is required to restore packets order arriving from the source.

Standard SpaceWire[1] does not define the quality of service (QoS). Therefore it does not contain methods to control the parameters defining the QoS. However, if several constraints in the network are specified and additional terminal nodes/switches functionality is implemented, the value of the delay and jitter will be within the prescribed limits.

In this paper, we propose a method that will help to reduce the value of jitter in data transmission with different throughput requirements.

II. Terms

In this paper we will use the following terminology.

Guaranteed class of service – packets for which throughput on each switch port is guaranteed.

Non-guaranteed class of service - packets for which throughput is not guaranteed. System allocates throughput for packets with guaranteed class of service first. The remaining throughput can be allocated for packets with non-guaranteed class of service. If remaining throughput is 0, then packets from non-guaranteed class of service are not processed by a switch.

Guaranteed period (T) - the time during which the throughput is guaranteed. It is defined by the network administrator.

Ci – throughput for class of service with identifier *i*.

Transmission time of packets with class of service *i (Tsci)* – this is part of time, during which packets with class of service *i* can be transferred in guaranteed period. This time can be calculated using the following formula:

$$T_{sci} = \frac{C_i * T}{100}$$

III. Description of the method

Changes to the standard are not required for use of the proposed mechanism. It's necessary to determine the limit size of packet and implement additional functionality in switches. Let us put that class of service is uniquely determined by logical address of the packet. On the switch we need to store additional information about the required throughput.

The total value of the throughput for guaranteed classes of service on one port of switch must not exceed 100%. If user plan to employ control codes in system, the required throughput must be taken into account.

All packets are placed into a single buffer space because SpaceWire credit scheme does not provide individual credits for different data streams. The speed of packet (with specified class of service) moving along the network is not taken into account here.

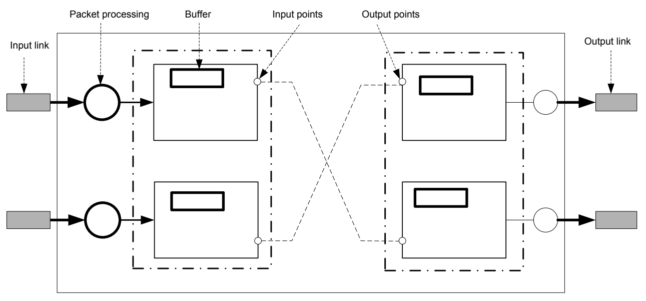


Fig. 1 Packet processing scheme

Network administrator should also set transmission delay parameters for each class of service in the ports of switch. Depending on the required throughput the interval between transmission of two adjacent packets of a given class of service is set. If the interval is equal to 0, then the packets are sent one after another without additional delays in the switch. This will lead to the packets with one class of service accumulation and the flow uniformity property will be lost.

t_b - transmission time of one Nchar. It depends on the system frequency and speed of the channels.

$SizePacket$ - packet size in bytes.

$NumPacket = \left\lfloor \frac{C \cdot T}{SizePacket \cdot t_b} \right\rfloor$ - maximum number of packets to be transmitted in the guaranteed period.

$Tout = \frac{T}{NumPacket}$ - period of time, which is allocated for one packet transmission.

$deltaTime$ - the time between arrivals of two adjacent packets with same class of service at the destination.

IV. Results

Researches of the proposed method were performed using the model DCNSimulator [2]. Several networks were built in the simulation environment. They are presented in Fig. 2 - Fig. 3.

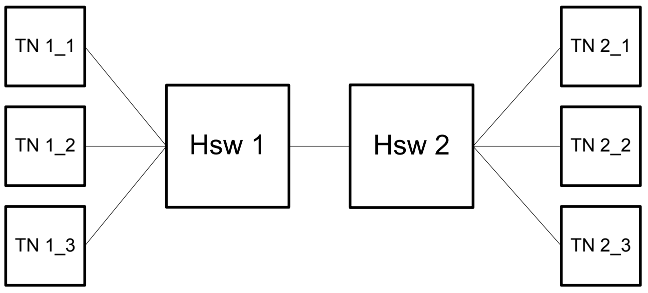


Fig. 2 Network № 1

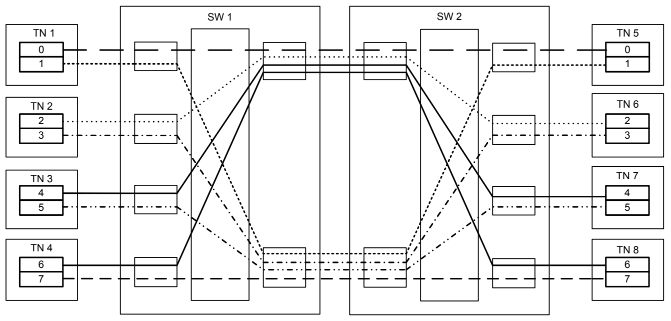


Fig. 3 Network № 2

Packets are transmitted via a single output port of the switch in the network № 1. It allows to explore packet streams distortion characteristics.

The structure of the network № 2 allows to research distortion characteristics of packet flows in the input port, which occurs due to the fact that the preceding packet from the input port can't move to the output port immediately because this port is processing another packet.

These two types of distortions are the main types of data flow distortions that can occur in SpaceWire network. It's possible to calculate packet transmission delay and jitter value if the information about data flow characteristics change after passing through sectors of first and second type and number of such sectors on the path from source and destination is available. On the other hand, the constraints on these parameters (transmission delay and jitter value) can be taken into account during network development process and data transmission routes determination.

Two types of researches were made using network structures show above: using mechanism of traffic smoothing and keeping its characteristic of uniform arriving at the receiver side and using mechanism of throughput guaranteeing only.

Research №1. For the structure of the network № 1 (Fig. 2), each of the data sources synchronously generates one packet every 21 microsecond. Sources identified as TN 1_1,1_2,1_3. Sources send data to the destinations. Destinations are identified as TN 2_1,2_2,2_3. Data transmission channels speed is 100Mb/s, devices frequency is 25MHz. Packet size is fixed and its size is 64 bytes. Simulation time is 8 ms. During this time 1139 packets are transferred. Packets are generated uniformly every 21 microsecond, therefore the ideal time between neighboring packets arrival to the destination is 21 microsecond. In the figures Fig. 4 - Fig. 6. you can see how the time between arrivals of two adjacent packets at the destination for packets with different classes of service changes when mechanism of traffic smoothing is used and when not. Jitter value when using traffic smoothing mechanism is approximately 6.6 times less then without it.

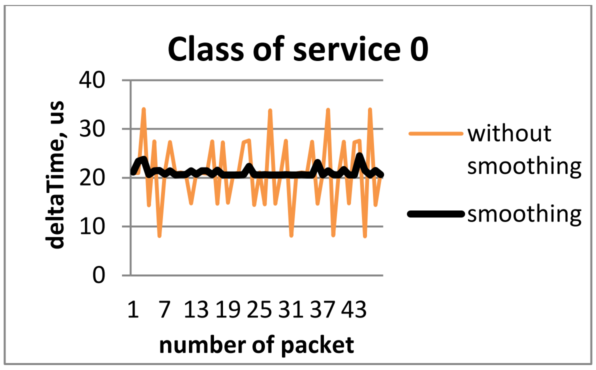


Fig. 4 Research №1. Packets with class of service 0

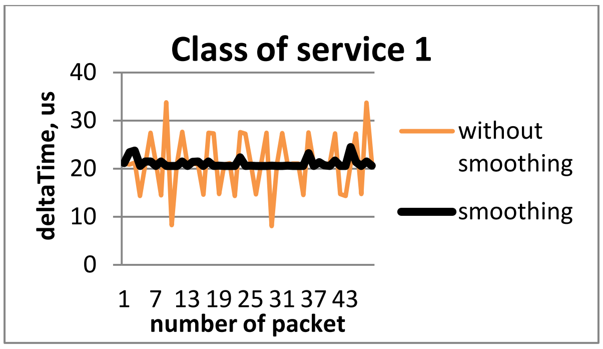


Fig. 5 Research №1. Packets with class of service 1

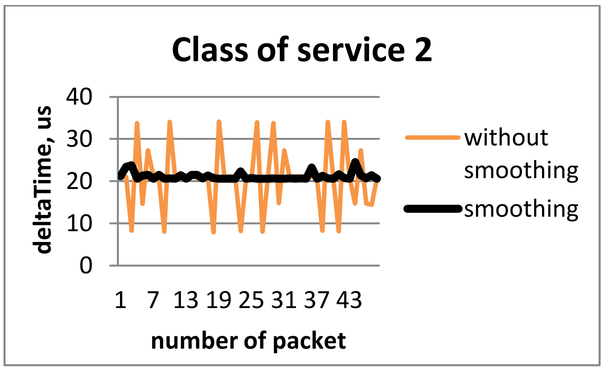


Fig. 6 Research №1. Packets with class of service 2

Research №2. For the network structure № 2 (Fig. 3), each of the sources of the data synchronously generates one packet every 26 microsecond. Data is transmitted along the routes of the corresponding color on Fig. 3. The speed of data transmission channels is 100Mb/s, devices frequency is 25MHz. Packet size is fixed and its size is 64 bytes. Simulation time is 4 ms. During this time 1232 packets are transferred. Packets are generated uniformly every 26 microsecond, therefore the ideal time between neighboring packets arrival to the destination is 26 microsecond.

In the figures Fig. 4 - Fig. 6. you can see how the time between arrivals of two adjacent packets at the destination for packets with different classes of service changes when mechanism of traffic smoothing is used and when not. Jitter value when using traffic smoothing mechanism is approximately 3 times less then without it.

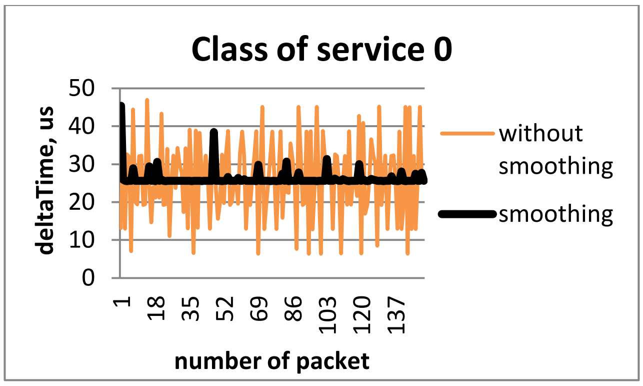


Fig. 7 Research №2. Packets with class of service 0

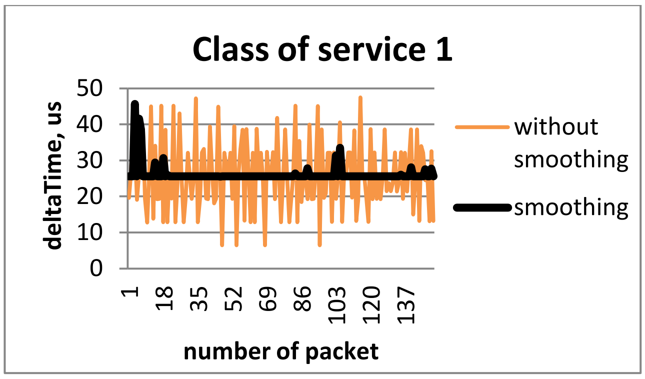


Fig. 8 Research №2. Packets with class of service 1

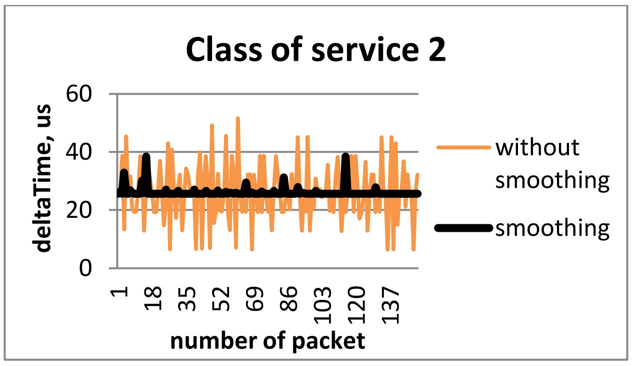


Fig. 9 Research №2. Packets with class of service 2

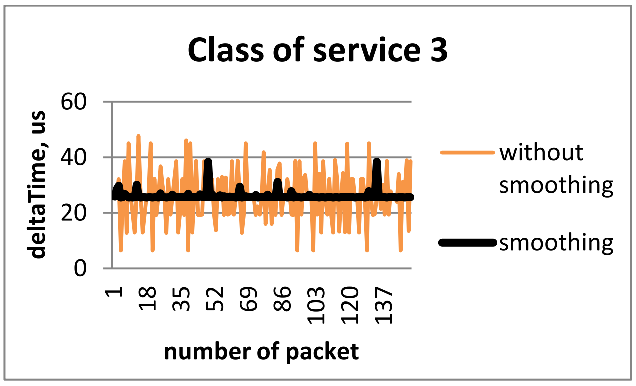


Fig. 10 Research №2. Packets with class of service 3

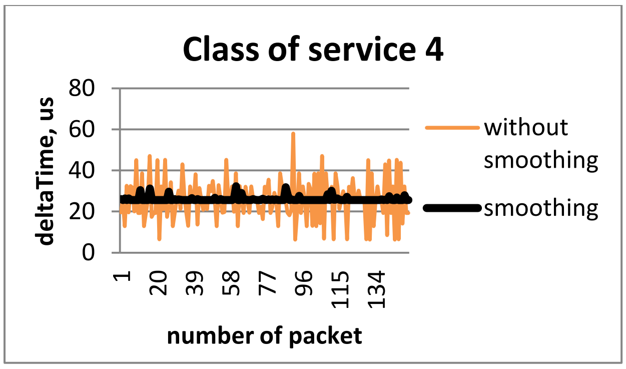


Fig. 11 Research №2. Packets with class of service 4

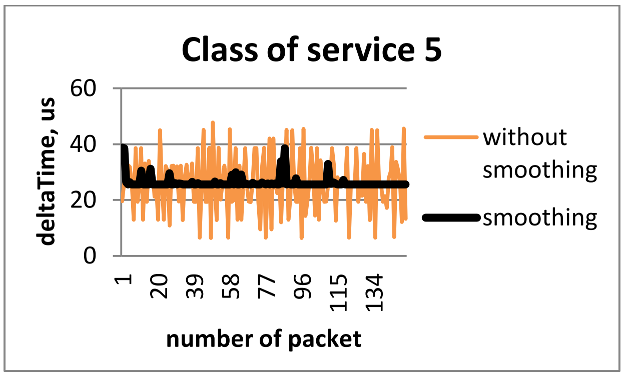


Fig. 12 Research №2. Packets with class of service 5

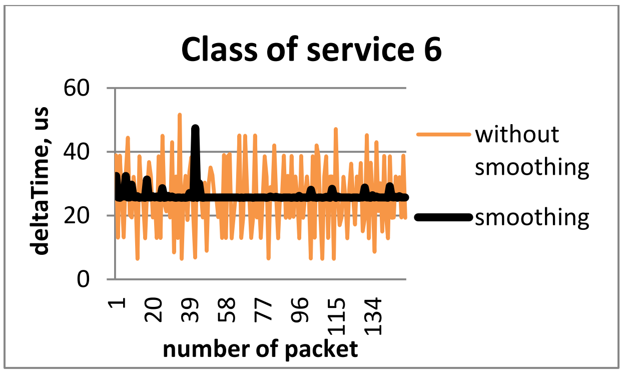


Fig. 13 Research №2. Packets with class of service 6

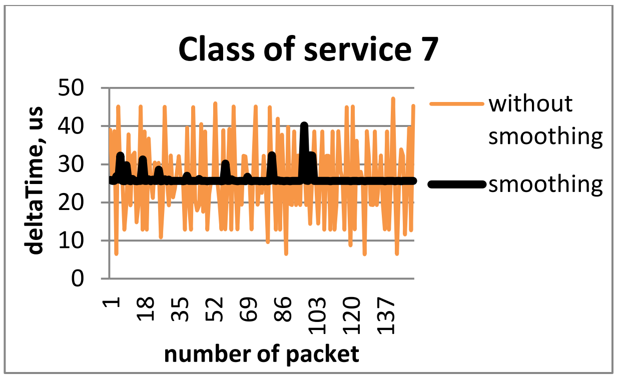


Fig. 14 Research №2. Packets with class of service 7

V. Conclusion

According to the Research 1 and 2, it can be concluded that the proposed mechanism can reduce jitter, to keep the data flow characteristics. Thanks to this mechanism jitter in SpaceWire network can be made predictable and customizable.

Acknowledgment

The investigations and results presented in the current paper are performed under the financial support of the Ministry of Education and Science of the Russian Federation.

References

[1] *Space engineering. SpaceWire – Links, nodes, routers and networks ECSS-E-ST-50-12C, 31 July 2008.*

[2] *A. Eganyan, L. Koblyakova, E. Suvorova, "SpaceWire network simulator. SpaceWire-2010" in Proceedings of international SpaceWire conference St.Petersburg 2010, pp.403-406, 2010.*

PHY Components for perspective SpaceWire-2 interface prototyping and evaluating

Components (Poster), Short Paper

Dmitri Skok, Sergey Kondratenko, Aleksey Zaicev, Alexander Glushkov, Tatiana Solokhina, Vladimir Gusev, Jaroslav Petrichkovich
«ELVEES» R&D CENTERof Microelectronics
Zelenograd, Moscow 124460, Russia
E-mail: tanya@elvees.com

Abstract—The results of simulation and measurement of the physical implementation of the SpaceWire-RT compatible transceiver units are discussed. These transceivers work at rates ranging from 5 Mbps up to 1.25 Gbps. The issues of electrical isolation, implementation, impedance matching and the selection of the physical transmission medium are considered. Different variants of the transmitter implementation in terms of power consumption to meet the requirements of impedance matching are compared. Transceivers are manufactured in CMOS 180 nm technology at JSC Mikron (Russia), employing topological approaches used for radiation-tolerant circuits. The variants of the implementation of transceivers with data rates up to 20 Gbps are proposed, including the use of 130-90 nm technologies.

***IndexTerms*—SpaceWire-RT, Transceivers, Physical Layer, Technology Scaling, Impedance Matching.**

I. Introduction

Currently «ELVEES» R&D CENTER works hard on the next-generation SpaceWire-RT standard specification and implementation.

The first thing that attracts attention in the analysis of new requirements for transceiver of SpFi-CML subsystem [1] – is an exceptionally large range of rates - from 0.1 to 20 Gbps in the medium term and up to 50 Gbps in the longer term at the data transmission over twisted-pair cable up to 5 m. Apparently, building transceivers for various sub-ranges of rate may differ. The draft protocol SpaceWire-RT actually fixes this situation, sharing LVDS transceivers (optional SpFi-LVDS) with rates up to 600 Mbps and CML transceivers (optional SpFi-CML) with the above rates. Some implementations of transceivers[2], [3] in CMOS technology with standards130 nm –250 nm suggests that the required data rates of up to 1.25 Gbps and up to 5 Gbps (depending on space and power constraints) are feasible.

An important objective is to limit the power consumption of the transceivers at no more than 200 mW. This is a strict requirement, because given the transmitter output voltage of 2 V just the power output to the line (100 Ohm) would be 40 mW, not including the losses in the matching circuit. Analysis of characteristics of the transceivers, available from Texas Instruments with the rate of up to 3.125 Gbps [4], showed, that their power consumption is in the range of 400 ... 700 mW.

Another goal is to extend the communication range up to 100 m, at least for selected configurations. While SpW itself is a relatively lightweight protocol, it is strongly preferable to keep SpaceWire-RT also such. It suggests keeping up with two-level signaling and avoiding power and space hungry complicated digital signal processing. In this case, the protocol would be suitable for inter-chip communication on-PCB as well.

This implies the following activities:

– selection of the transmitter circuit solutions (line driver namely) that provide greater efficiency;
– analysis of the possibility of decreasing the output signal level of the transmitter, which can simultaneously improve the electromagnetic compatibility of the system;
– seeking for solutions to further simplify cabling.

II. Cable Characterization and System Simulation

In order to estimate the practical limits of the communication range and available data rates, different samples of common cables have been S-parameters measured and their linear models developed. Samples include:

- 8 m CAT5 STP;
- 6 m RG-58;
- 1 m 50 Ohm thin coax.

For all samples S11, S12, S21, S21 were recorded in the frequency range of 300 kHz – 13 GHz. For STP also cross-talk at near and far end was measured.

With the media models at hand, a number of analyses were carried out:

- eye diagram based estimate of the maximum communication range for different data rates and media types;

- TX side pre-emphasis and RX side equalization in order to increase range and rate;
- feasibility of duplex communication over the single coax cable.

Simulation results of the ideal TX and RX blocks then used to verify their respective implementations.

Simulation results of the transmitter is shown on fig. 1. The simulation involves also the 8 m UTP transmission line and the matched receiver.

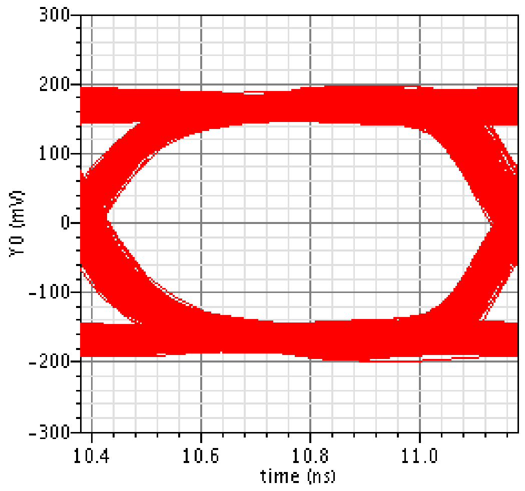


Fig.1. Eye diagram at the output of the transmitter, PEX model, 1250 Mbps

Eye diagram at the receiver input is shown on fig. 2. One can see that the eye is open at 1250 Mbps. Other simulations show that the practical limit of the UTP line length at this speed, without equalization and pre-emphasis, is about 20 m. These results are in good agreement with the experimental ones.

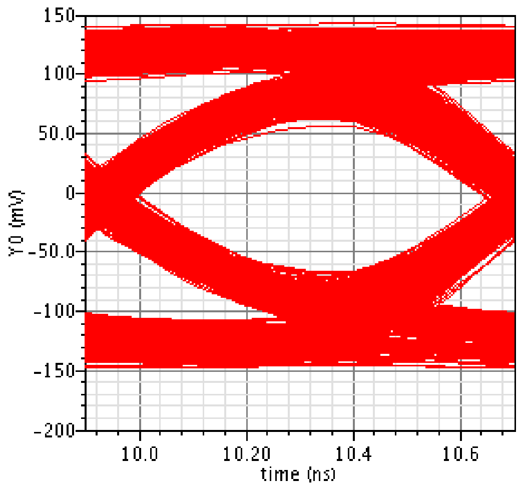


Fig. 2. Simulated eye diagram at the input of the receiver. Media: 8 m CAT5 UTP

In order to justify the requirements to the input characteristics of the receiver in the absence of decoupling transformer and to estimate the symmetry of the transmitter output, common mode signal was also simulated (fig.3). The common mode may also be critical to electromagnetic compatibility depending on application.

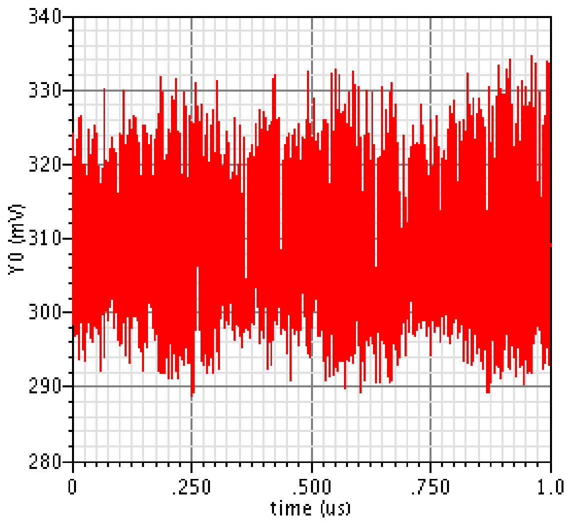


Fig. 3. Common mode signal at RX input

III. Possible implementations of CML-compatible driver composed of with REDUCED POWER CONSUMPTION

In the transmitter-receiver pair the driver in the transmitter has the most significant power consumption.

Common SpaceWire-RT transmitter physical layer implementation is based on CML logic. Despite the obvious advantages, this circuit has relatively high power consumption, as the considerable part of the supply current passes through the terminating resistors. At the same time, the performance requirements for transceivers SpFi-CML at the physical layer can be fully satisfied by using CML-logic as well as VML-logic. Equivalent output circuit of CML- and VML-drivers are identical and both exhibit a voltage source with the output impedance of 100 Ohms.

Another VML-style driver advantage, compared to CML drivers, is the possibility to decrease the supply voltage down to 1 V and below while maintaining sufficient output voltage swing. This facilitates utilization of deeper sub-micron technologies (90-130 nm) in order to further increase the communication rate.

IV. Structures and characteristics of the developed Transceivers

The transceivers were designed for the SpFi-CML subsystem of SpaceWire-RT protocol, which contain digital parts with a supply voltage of 1.8 V. All designed devices manufactured with CMOS 180 nm technology.

A. The analog part of the transmitter and receiver

The adjustable driver was designed for research purposes (fig.4). Output signal level can be changed by varying supply voltage AVDD of VML-cascade in the 1.8-3.3 V range. The designed VML-driver has the Data Rate (DR) of up to 1.25 Gbps. Line impedance matching achieved with the external circuitry.

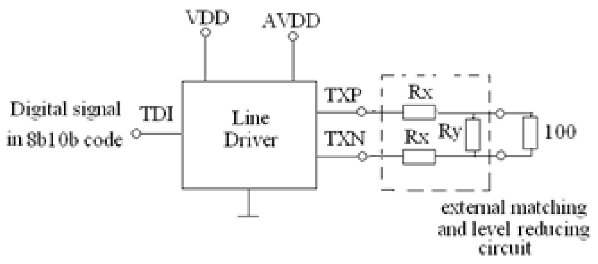

VDD
AVDD
Digital signal in 8b10b code
TDI
Line Driver
TXP
TXN
Rx
Rx
Ry
100
external matching and level reducing circuit

Fig.4.The adjustable driver

Although is not optimal for production, this approach allows independent variation of both output swing and line impedance matching for testing and investigation purposes.

The receiver is capable of 1.25 Gbps data rate with the sensitivity of ±30 mV.

Total estimated dissipating power (TDP) of both transmitter and receiver analog blocks is from 85 mW to 170 mW, depending on operation conditions. Power consumed by these units at the lower boundary supported transceivers speed range is less than 100 mW.

B. The structure and characteristics of the transmitter

Block diagram and interface of transmitter IP block shown in fig.5. The composition and functions of the main blocks of the transmitter:

- BR – parallel 10-bit buffer register. Takes TXD [9:0] code group at the rising edge of CLK –provided EN=1. If EN = 0 BR contents does not change.
- PLL – frequency synthesizer of bits F_{bit}.
- P2S –converts the content of BR into a sequence of DATA bits, starting from bit 9 (MSB).
- C – switch. In normal mode (LB_EN =0) Data = DATA, LB_OUT = 0. In LoopBack mode (LB_EN=1) Data=LB_IN, LB_OUT=DATA.
- TX – output driver. Converts the sequence of digital bit Data to differential pair of analog signals TXP, TXN. If PWDn =0 or EN = 0 outputs TXP, TXN are set to high -Z.

Data rate is selected by the code at on the control inputs SPEED:
DR =5, 10,...(step5) ...125 Mbps- the lower rate range; DR = 312.5,625, 1250Mbps-upper rate range.

The frequency of the characters (parallel 10-bitcode groups) - DR/10. Reference frequency- CLK =125 MHz.

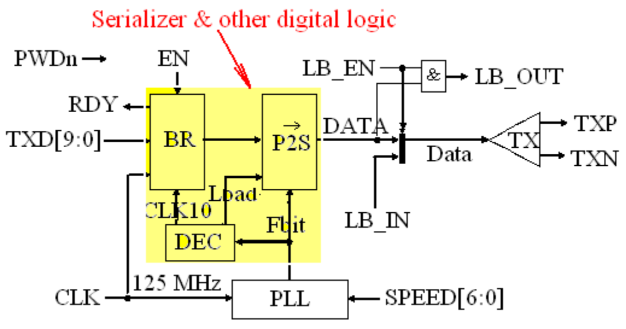


Fig.5. The structural circuit and interface of the transmitter

C. The structure and characteristics of the receiver

Block diagram and interface of receiver IP block shown in fig. 6.The composition andfunctions of the mainblocksof the receiver:

- RX–input differential amplifier. Converts the differential signal RXP, RXN to digital DATA. While PWDn =0 or when the input signal is absent (RXP ≈ RXN) is set DATA to 1.
- C –switch. In normal mode (LB_EN = 0) Data0 = DATA, LB_OUT = 0. While LoopBack (LB_EN = 1) Data0 = LB_IN, LB_OUT = DATA.
- CDR –restores CLK = Fbit and synchronized bit sequence Data from the input bit sequence Data0 ("Clock and Data Recovery"). Controls the synchronization mode (using REFCLK = 125 MHz): CDR_MODE [1:0] - management, LOCK [1:0] –the indicator of capture frequency and phase. Restores CLK = Fbit and synchronized bit sequence Data from the input bit sequence Data0 ("Clock and Data Recovery").
- ALG – contains a 10-bit shift register that stores the current sequence of bits Data (basic function block). Synchronizes (CLK/10) output data parallel code. With permission (ALIGN_MODE), detects COMMA sequence, aligns the boundaries of 10-bitcode groups (cyclic shift numbers of digits) that has a flag COMMA_DET =1 and the error signal ALIGN_ERROR =1 (before alignment boundaries are not aligned).
- BR –buffer 10-bit register of output data RXD [9:0].

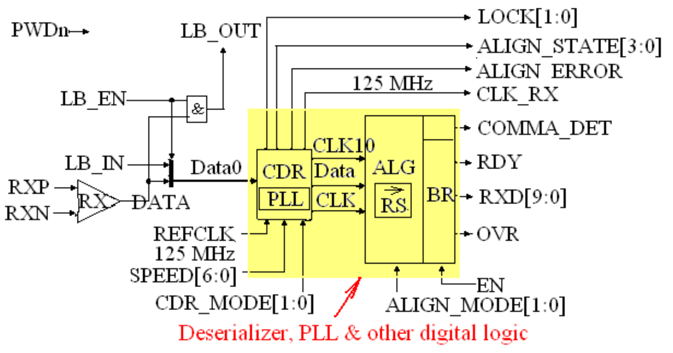


Fig.6.The structural circuit and interface of the receiver

The total estimated power consumption of the digital part of the transmitter and the receiver is less than 30mW, that fits well the total SpFi-CML power budget.

D. Topology and the placement of the transceivers on chip

Dimensions of IP blocks of the transmitter and receiver on the chip are the same - 470 × 395 µm ², square - 0.186 mm^2 (including 2 elements of ESD protection). Block sizes and pin locations allow for direct attach each of them standing side by side with two analog pin elements (fig.7).

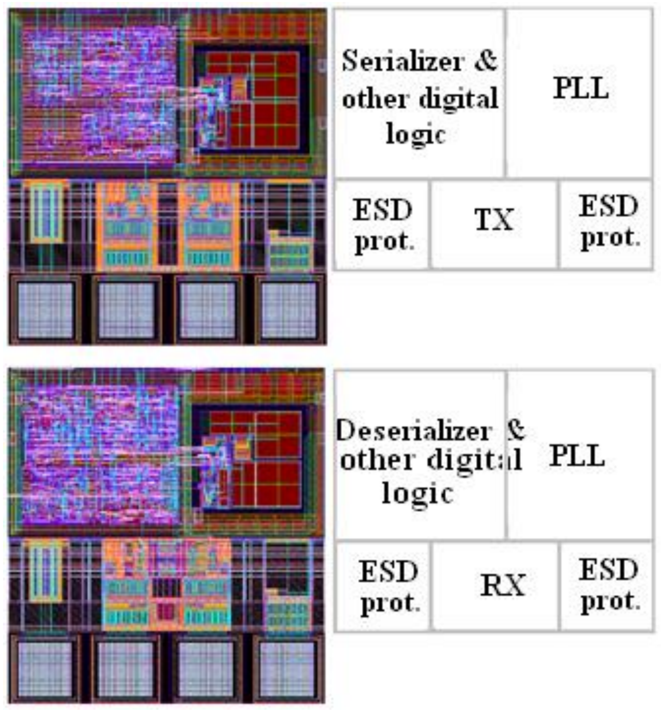

Serializer & other digital logic
PLL
ESD prot.
TX
ESD prot.
Deserializer & other digital logic
PLL
ESD prot.
RX
ESD prot.

Fig.7. Topology of the transceivers on chip

V. PHYSICAL LAYER PROFILES SPECIFICATION PROPOSAL

The ongoing SpaceWire-RT standard specification combines, from the one hand, the wide range of transmission rates, and from the other, the requirement of galvanic isolation between transmitter and receiver.

Readily available transformers don't provide sufficient bandwidth (~ 1 MHz — 5 GHz) to cover the entire rate range. On the other hand, different data rates suggest different media types and application conditions. It seems natural to identify and denote the set of parameters, nominating those conditions, and specify them as a separate profiles withing the same standard.

Profile A. Most suited for standard CAT5 UTP/STP media and communication range of up to 60 m @ 125 Mbps data rate.

Profile B. Similar to Profile A, but with increased data rate and reduced range. As the transformer is a limiting factor to data rate range, the minimum rate is also increased.

Profile C. Is targeted the to data transmission within the same PCB or between PCB's of the same device. As the galvanic isolation requirements mostly eliminated in this configuration, simple capacitor decoupling may be used. Hence the maximum data rate and wide rate range, if needed.

Profile D. This profile is distinct from the others in that it uses a single coaxial cable (RG-58 or like) for full duplex data communication. Unlike UTP/STP, this cable is specified over a much wider frequency range, has lower insertion loss and dispersion. Together these provide for higher data rates at longer distances. Depending on the cable model, its weight per meter may also be substantially lower than that of UTP. As a drawback, this profile requires echo cancellation circuitry at each side. Also, voltage difference between the ends should be limited for human safety reasons.

Profile E. Implies data transmission over the optic fiber. Galvanic isolation is intrinsic and the rate is virtually unlimited.

VI. CONCLUSION

This article presents analysis of the SpaceWire-RT protocol requirements for SpFi-CML subsystem on twisted pair. The possibility and feasibility of VML-transceivers compliant physical layer transceivers with CML-transceivers and have compared them with several advantages, including the power consumption. The results of development of transceivers with data rates in the range of 5 Mbps ...1.25 Gbps, overlapping needs of SpFi-CML subsystems of second generation. The directions of further development of the standard are offered.

REFERENCES

[1] "D2.1 - SpaceWire-RT Outline Specification", SPACEWIRE-RT Consortium, 06.09.2012.

[2] S. Habinc, J. Gaisler, "GR712RC – A multiI-processor device with SpaceWire Interfaces", International SpaceWire Conference 2010, June 2010, pp 153-157.

[3] S. Kondratenko, V. Baikov, Yu. Gerasimov, T. Solokhina, "LVDS IP-blocks for high speed data transmission in SpaceWire systems", International SpaceWireConference 2010, June 2010, pp 153-157.

[4] www.ti.com,site of Texas Insruments company.

SpaceWire Traffic Generator: a highly-scalable packet generation device

SpaceWire test and verification, Poster

Takayuki Yuasa, Tadayuki Takahashi
Institute of Space and Astronautical Science, JAXA,
3-1-1 Yoshinodai, Sagamihara, Kanagawa 252-5210, Japan

Masaharu Nomachi
Osaka University,
1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan

Iwao Fujishiro, Fumio Hodoshima
Shimafuji Electric,
8-1-15 Nishi-kamata, Ohta, Tokyo 144-0051, Japan

I. BACKGROUND: SPACEWIRE BACKPLANE FOR GROUND TEST EQUIPMENTS

JAXA, Osaka University, and Japan Space Systems have been developing a SpaceWire backplane system for ground test equipments based on Micro Telecommunications Computing Architecture (hereafter, uTCA) as explained in Nomachi et al. (this conference). Development involved not only a uTCA shelf but also a generic SpaceWire FPGA card and 28-port SpaceWire router card which can be inserted to the ordinary AMC (advanced mezzanine card) slot the dedicated controller slot (i.e. MicroTCA Carrier Hub slot), respectively. Typical outlook of this backplane system is presented in Figure 1.

Each AMC slot provides 4 SpaceWire links connected to the 28-port router via the backplane tracks, and therefore, inserted SpaceWire FPGA cards can communicate each other. The backplane link topology is shown in Figure 2. Our generic SpaceWire FPGA card also provides 4 external SpaceWire connectors on the front panel. Since the number of cards can be easily increased just by inserting a new card to available slots, this system is highly modular and scalable. The maximum operational link rate of all the SpaceWire links in the SpaceWire Backplane system is 200 MHz.

Combined with the generic SpaceWire FPGA card, this backplane system can be used as a development platform for SpaceWire-based ground support electronics and/or a simulator for a large-scale SpaceWire network.

II. SPACEWIRE TRAFFIC GENERATOR

As an application of the SpaceWire uTCA Backplane system, aiming at a packet injection test for a multi-port SpaceWire router or a large-scale network, we implemented a traffic generation logic on the generic SpaceWire FPGA card. One or more Traffic Generator modules inserted to a uTCA shelf compose a SpaceWire Traffic Generator system together with control software on a computer. The system block diagram of SpaceWire Traffic Generator is presented in Figures 3. Development of SpaceWire Traffic Generator is supported by JAXA and the product is available from Shimafuji Electric.

Commercially available uTCA shelves typically offer 6 or 12 AMC slots, and therefore, the number of external SpaceWire ports can be extended, with a step of 4, up to 24 or 48 depending on the shelf. These numbers allow SpaceWire Traffic Generator to be used for high-traffic tests of a large SpaceWire router or even a network. These external SpaceWire ports support Tx link frequencies of 200/n MHz where n is a natural number. SpaceWire links over backplane operate at 200 MHz by default for providing small latency and high data transfer speed for control purposes.

A typical operation flow of Traffic Generator is presented in Figure 4. In the following sections, we describe details of

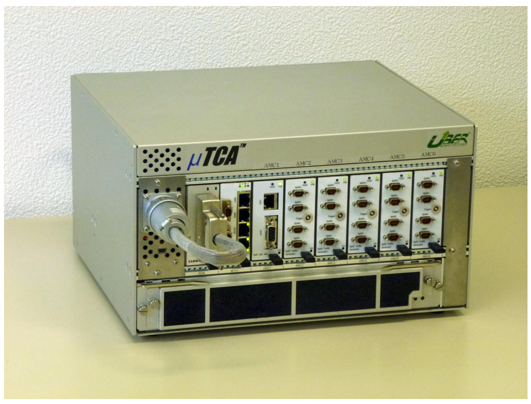


Fig.1 An example outlook of SpaceWire Traffic Generator enclosed in a 6-slot uTCA shelf. In this picture, 5 SpaceWire Traffic Generator modules are inserted, i.e. 20 external test ports are available. Two modules connected by a thick cable are power supply and distribution modules.

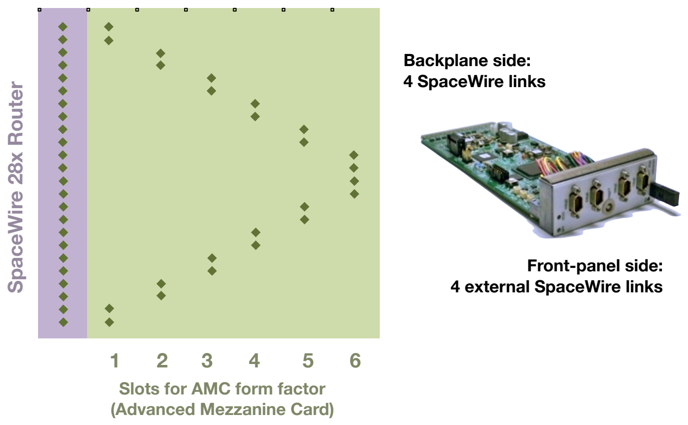


Fig.2 Topology of the backplane SpaceWire connection for a typical 6-slot backplane rack. 4x6=24 links from the backplane and 4 from the front panel of the router module are interconnected by the 28-port router shown in purple. A picture of the generic SpaceWire FPGA card is also shown.

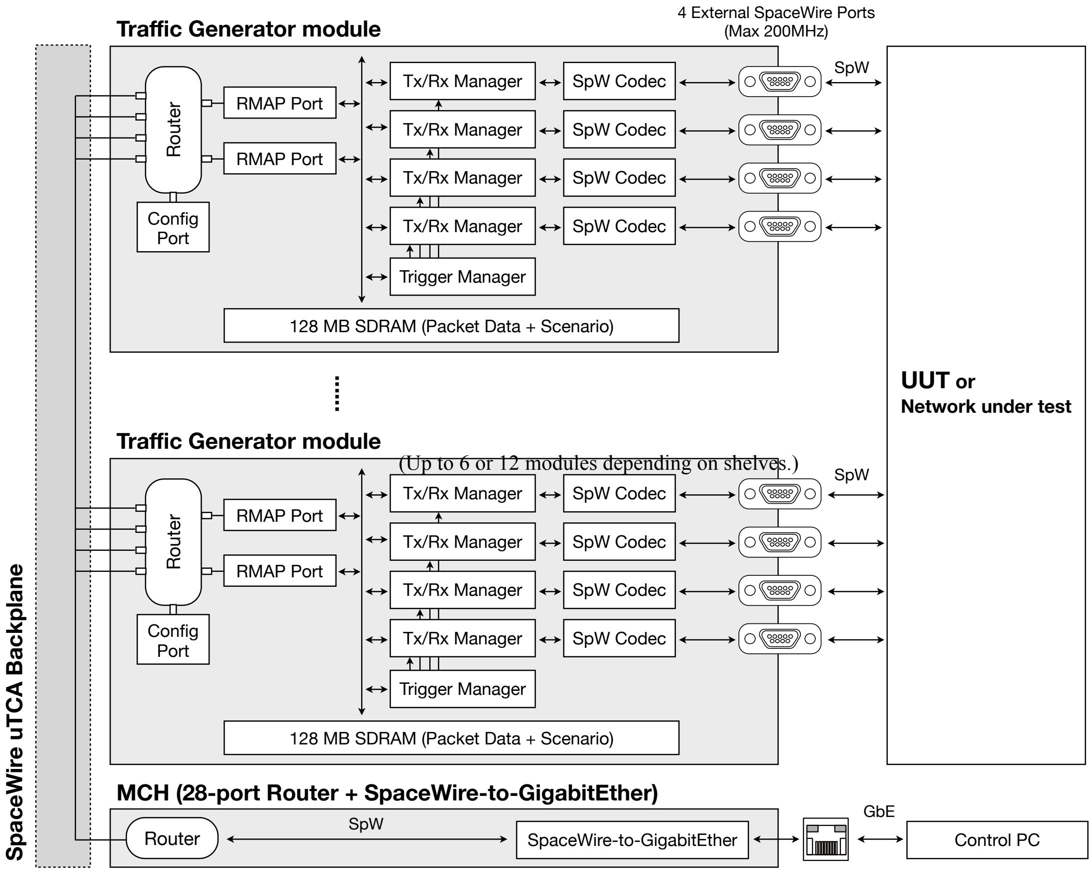

Traffic Generator module
4 External SpaceWire Ports (Max 200MHz)
Router
RMAP Port
RMAP Port
Config Port
Tx/Rx Manager
SpW Codec
Trigger Manager
128 MB SDRAM (Packet Data + Scenario)
SpW
UUT or Network under test
(Up to 6 or 12 modules depending on shelves.)
SpaceWire uTCA Backplane
MCH (28-port Router + SpaceWire-to-GigabitEther)
SpaceWire-to-GigabitEther
GbE
Control PC

Fig.3 A block diagram of SpaceWire Traffic Generator. Each Traffic Generator card is attached to the backplane, and has 4 SpaceWire links connected to the router in the MCH module.

packet transmission, reception, triggering functions, and dedicated control software.

III. Packet Transmission

Packet transmission from the Traffic Generator external SpaceWire port is controled by Tx packet descriptors whose structure is shown in Figure 5. Figure 6 illustrates an overview of packet transmission procedure. The control software on the computer writes Tx descriptors to the Tx Descriptor FIFO of the Traffic Generator, and the Tx logic will send packets by consuming and interpreting the descriptors stored in the FIFO. The Tx Descriptor FIFO has a depth of 1024 descriptors, and can be further updated during packet transmission.

Packet content should be written to the onboard SDRAM before starting packet transmission. The Traffic Generator does not manipulate packet content, and therefore, users should construct valid packets on the control computer when any upper layer protocol, sucha as RMAP or GRDDP, is necessary. In addition to ordinary SpaceWire packets, Timecode characters can be emitted by setting the "Mode flag" field properly.

The "Tx Wait" paramter in the descriptor which defines wait duration between the completion of packet transmission for the current descriptor and start of the next descriptor. In addition to this descriptor-to-descriptor interval, there are several types of parameters can be modified via registers to globally control transmission speed, i.e. configurable NULL

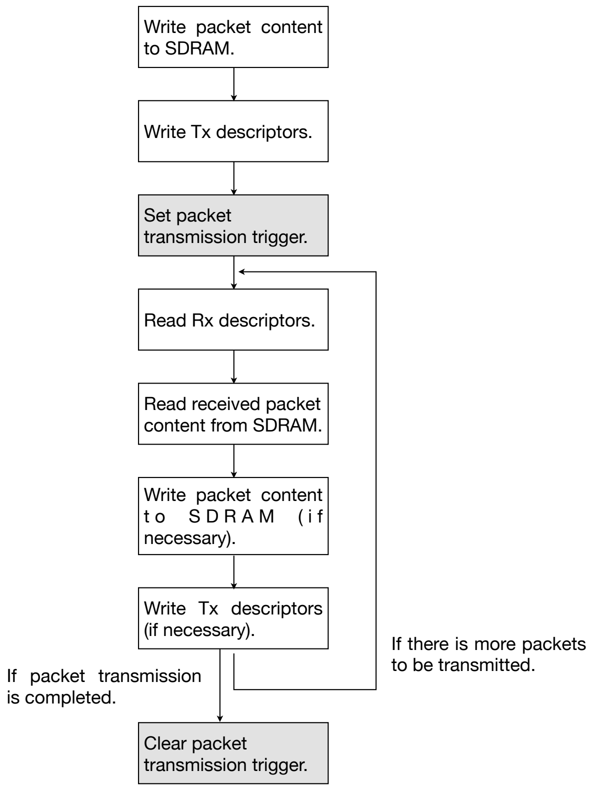

Write packet content to SDRAM.
Write Tx descriptors.
Set packet transmission trigger.
Read Rx descriptors.
Read received packet content from SDRAM.
Write packet content to SDRAM (if necessary).
Write Tx descriptors (if necessary).
If there is more packets to be transmitted.
If packet transmission is completed.
Clear packet transmission trigger.

Fig.4 A flow chart of a typical operation of Traffic Generator.

interleaving and fixed wait interval for Tx descriptor consumption.

Definitions of individual fields of the Tx descriptor are as follow:

- **Tx Descriptor ID**: 7-bit arbitrary number which can be used to identify Tx descriptor. This ID is recorded, combined with time information, in the Tx Log FIFO when packet transmission defiend in this descriptor is completed.
- **Tx Memory Address**: A pointer to packet content stored in the SDRAM Tx packet area.
- **Mode Flag**: A 2-bit flag that determines end-of-packet marker type or Timecode. 00 = no EOP/EEP, 01 = terminated with EOP, 10 = terminated with EEP, 11 = Timecode.
- **Timecode value**: Timecode with this value will be emitted instead of ordinary packet when Mode flag is 11.
- **Tx Length**: Length of packet content.
- **Tx Wait**: A 2-bit flag and a 14-bit counter. The flag determines resolution of the counter. 00 = 5ns, 01 = 1us, 10 = 1ms, 11 = 1s. Extraction of next descriptor will be delayed

Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	TxDescriptor ID (7bit)							Tx Memory Address Tx_MA[26:2]																								
2	Mode Flag		Time Code Value (6bit)						Tx Length Tx_LEN[23:0]																							
3	Reserved (0x0)																															
4	Wait duration until next packet Tx_Wait[15:0]																Tx Repeat Count Tx_RPT_CNT[15:0]															

Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	LINK STATUS							Rx Memory Address Rx_MA[26:2]																								
2	Mode Flag		Time Code Value (6bit)						RSV (8bit)								Rx Length Rx_LEN[15:0]															
3	Local Time (Arrival of the first byte)																															
4	Local Time (Arrival of the last byte)																															

Fig.5 Structures of Tx and Rx packet descriptors. Tx descriptors are generated by the control software and written to the Tx descriptor FIFO on the Traffic Generator module. Rx descriptors are created by the receiver logic when a packet (or timecode) is reived, and read out by the control software. Data contained in a packet is stored in SDRAM, and addresses are pointed by the "Tx Memory Address" or the "Rx Memory Address" fields.

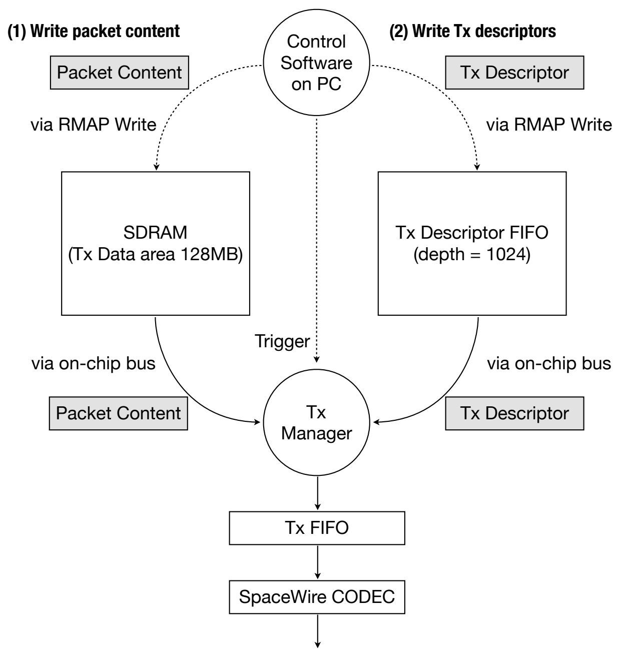


Fig.6 A flow chart of the packet transmission procedure.

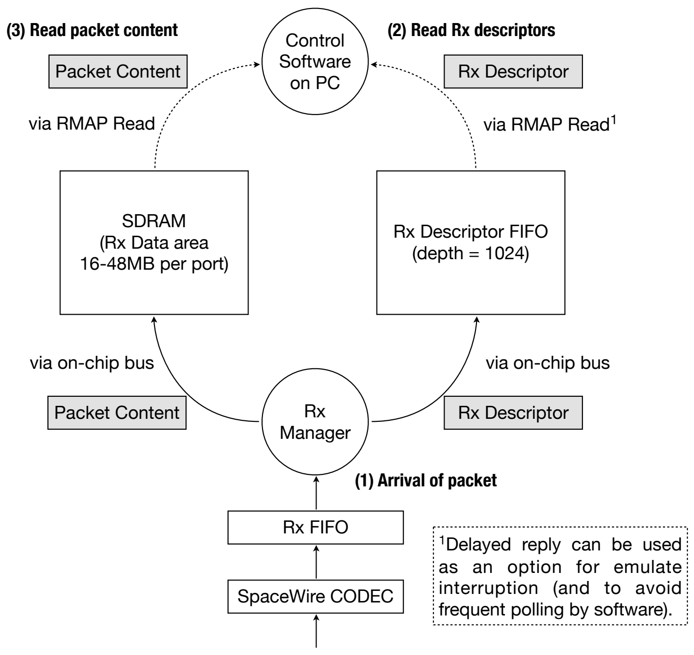


Fig.7 A flow chart of the packet reception procedure.

by 14-bit counter * resolution (2-bit flag).

- **Tx Repeat Count**: A 16-bit counter that specifies the number of repeated execution of the current descriptor.

Every outgoing packets will be time-tagged (times of emission of first and last bytes), and recorded as Tx Log. Local time has a 5ns resolution and 32-bit width (0-21.4s).

Note that the Traffic Generator is designed to operate over the Packet layer of the SpaceWire standard, and therefore, it is not capable of Character/Signal level manipulation such as bit-error injection or DS-signalling error injection.

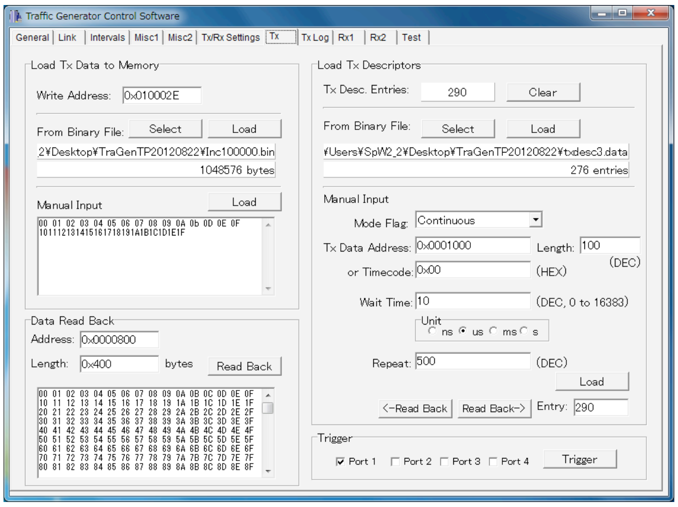
IV. PACKET RECEPTION

Packet receive process is described in Figure 7. When a packet is received at Traffic Generator, an Rx descriptor which has the structure presented in Figure 5 will be created, and stored in the Rx Descriptor FIFO which has 1024 descriptor depth. Received packet content will be stored in the SDRAM which is used as a 16-MB ring buffer by default (the size can be modified from 0-48MB). Reception of a timecode character also results an Rx descriptor that has Mode flag = 11 and timecode value in the "Timecode value" field. No data will be written to the SDRAM.

To read the received packet content, control software should read the Rx descriptors from the FIFO, and then read the data from the SDRAM. When the Rx Descritptor FIFO is full, the either of the following operations are performed by the receive logic depending on a configuration (changeable via register) ; (1) received packets are simply discarded, and number of discarded packets are recorded (i.e. sink mode), or (2) the SpaceWire link is blocked by not replying FCTs.

V. CONTROL SOFTWARE

Shimafuji Electric developed control software for Traffic Generator whose screenshot is presented in Figure 8. The software provides GUI interfaces for each functions of Traffic Generator, such as filling Tx Descriptor FIFO, filling Tx packet content to the SDRAM, triggering packet transmission, reading Rx Descriptor FIFO, and reading received packet content from the SDRAM.



Load Tx Data to Memory

Write Address: 0x010002E

From Binary File: 2¥Desktop¥TraGenTP20120822¥Inc100000.bin

1048576 bytes

Manual Input

```
00 01 02 03 04 05 06 07 08 09 0A 0b 0D 0E 0F
101112131415161718191A1B1C1D1E1F
```

Data Read Back

Address: 0x0000800

Length: 0x400 bytes

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
```

Load Tx Descriptors

Tx Desc. Entries: 290

From Binary File: ¥Users¥SpW2_2¥Desktop¥TraGenTP20120822¥txdesc3.data

276 entries

Manual Input

Mode Flag: Continuous

Tx Data Address: 0x0001000 Length: 100 (DEC)

or Timecode: 0x00 (HEX)

Wait Time: 10 (DEC, 0 to 16383)

Unit: ns [✓ us] ms s

Repeat: 500 (DEC)

Entry: 290

Trigger: [✓ Port 1] Port 2 Port 3 Port 4

Fig.8 A screenshot of the Traffic Generator control software.

The software also provides an easy scripting language to generate Tx Descriptors for fast test case generation, and an example script is shown in Figure 9.

The software has been deployed together with Traffic Generator for testing the realtime performance of SpaceWire Middleware implemented on the SpaceCard flight computer (Mitsubishi Heavy Industry) in Feb. 2013. The flexible packet transmission functions which is also fairy accurate in time (5-ns resolution) was very effectively used in the test.

VI. CONTROL SOFTWARE

JAXA, Osaka University, and Shimafuji Electric have been developing a SpaceWire Traffic Generator system which is highly-scalable in terms of SpaceWire port number (e.g. 4-24 in 6-slot uTCA rack).

We consider that SpaceWire uTCA Backplane and SpaceWire Traffic Generator are potentially useful test beds for all SpaceWire-related developers, and will make these available internationally through our collaborators.

```
TxDescriptor Mode=EOP TxAddress=0x00000800 Length=100 Repeat=1 Wait=0ms
TxDescriptor Mode=Continuous TxAddress=0x00001000 Length=100 Repeat=3 Wait=10us
TxDescriptor Mode=EEP TxAddress=0x00002000 Length=10 Repeat=1 Wait=0ms
TxDescriptor Mode=Timecode Timecode=0x03 Wait=0ms
```

Fig.9 An example script which generates 4 Tx descriptors. Three of them are for a 100-byte packet terminated with an EOP, a 100-byte packet with no end-of-packet marker (repeated 3 times, wait 10us after single transmission), a 10-byte packet terminated with an EEP. The last line generates a timecode character with a time-code value of 0x03.

A Scheduling Method of RMAP Packets for SpaceWire-D

Poster Paper

Yang Chen, Mitsutaka Takada, Ryo Kurachi, Hiroaki Takada
Center for Embedded Computing Systems,
Graduate School of Information Science,
Nagoya University, Furo-cho, Chikusa-ku, Nagoya, Japan
chenyang@ertl.jp, mtakada@nces.is.nagoya-u.ac.jp, kurachi@nces.is.nagoya-u.ac.jp, hiro@ertl.jp

***Abstract*— SpaceWire-D is a time-triggered protocol developed for SpaceWire to provide deterministic transmission and real-time constraint for RAMP packets. According to SpaceWire-D, any RMAP packet has to be assigned into a time slot; and transmission of the packet is required to meet the real-time constraint—the transmission has to be completed before the end of the time slot.**

The assignments of the packets constitute a schedule table that should be predetermined and held by the nodes in a SpaceWire system. However, because of the complexity of the network communications and the massive packets in a real SpaceWire system, a schedule table that ensures all the packets meeting the real-time constraint is hard to be determined. To take this issue, we propose a Simulated Annealing based scheduling method of RMAP packets for SpaceWire-D. The method choses paths for the packets, assigns packets to the time slot with respect to the real-time constraint. The proposed method is evaluated by the simulations and the effectiveness is shown in the experimental results.

***Index Terms*— Relevant indexing terms: SpaceWire, Networking, Spacecraft Electronics, Real-time scheduling.**

I. Introduction

SpaceWire has been developed as a network standard for spacecraft, and widely used by aeronautical organizations and companies. Within a SpaceWire network, the nodes are connected through low-cost, low-latency, full-duplex, point-to-point serial links and packet switching wormhole routing router. It utilizes asynchronous communication and allows speeds between 2 Mbit/s and 400 Mbit/s [1].

SpaceWire provides a means of sending packets of information from a source node to a target node. Remote Memory Access Protocol (RMAP), a standard communication protocol for SpaceWire network, is used to specify the packets[2]. However, transmission of the RMAP packets on a spacecraft always has high real-time requirement. Hence the need for deterministic transmission of information arises, and providing this determinism for SpaceWire networks is essential. For this reason, SpaceWire-D, a time-triggered protocol developed for SpaceWire, has been proposed to provide the deterministic real-time transmission [3].

Based on SpaceWire-D, a schedule table is constructed, in which the system time is divided into time slots, and any packet in the system is assigned to a deterministic time slot with the real-time constraint—transmission of the packet must be completed before the end of the time slot.

However, because of the complexity of the network communications and the massive packets in a real SpaceWire system, such a schedule table that ensures all the packets meeting the real-time constraint is obviously hard to be determined without an efficient scheduling method.

To take this issue, in this paper we propose a scheduling method of RMAP packets for SpaceWire-D. In detail, first, we provide an algorithm to calculate the worst-case latency (WCL) of packets in a time slot to guarantee the real-time constraint. Second, we propose a Simulated Annealing (SA) based scheduling method to obtain a schedule table with respect to the real-time constraint. Particularly, considering the redundancy requirements, both the WCL calculation algorithm and the scheduling method allow the system to hold redundant path for packets transmission. The proposed method is evaluated by the simulations and the effectiveness is shown in the experimental results.

The remainder of this paper is organized as follows. Chapter II introduces the system model. Chapter III presents an algorithm for calculating the WCL of packets in a time slot. Chapter IV proposes the scheduling algorithm. Evaluation of the proposed scheduling algorithm is shown in Chapter V, and followed by Chapter VI that concludes the paper and discusses future works.

II. System Model

We assume an object system is composed of nodes, routers and links, as the format specified in [1], in which each node is able to send and receive packets. Based on SpaceWire-D, time code is used to distribute system time over a SpaceWire network. The interval between two adjacent time codes is defined as a time slot. A packet has to be transmitted during a pre-determined time slot. The deterministic transmitting information of all packets is included in a scheduling table, held by each node that may send packets in the system.

```
Algorithm 1  Algorithm of searching all paths from
             source to target
pathSearch(paths, passedLinks,
           passedRouters, source, target):
1:  for all links starts from source do
2:     passedLinksNew ← passedLinks appends link
3:     if link connects to target then
4:        paths ← paths appends passedLinks
5:     else if link connects to router then
6:        if router is not in passedRouters then
7:           passedRoutersNew ← passedRouters
                appends router
8:           paths ← pathSearch(paths, passedLinksNew,
                passedRoutersNew, router, target)
9:        end if
10:  end for
11: return paths
```

In general, a time slot will be used by different applications to send their packets. To classify their transmissions, we define a channel as a logic group that holds the information used by an application to transmit its packets [4]. A channel has a unique source node that sends packets of the application, whereas it may have multiple targets node for receiving the packets. Packets of a channel can be sent from the source to any one of the targets. The path from a source to target is the sequence of the links between the source and target. It is assumed that a path must be pre-determined.

In order to avoid that the fault of one link breaks all the transmission, redundant paths are usually configured between a source and a target, which are the paths between the source and target without sharing any links. The redundancy is the number of the redundant paths, which is determined by the application and limited by the links. For example, if there are 3 paths $\theta_1, \theta_2, \theta_3$ between a source and a target without sharing any links, the maximum redundancy from the source to target is 3. While if the application configures paths with redundancy 2, then there are 3 candidates can be chosen—$\{\theta_1, \theta_2\}$, $\{\theta_1, \theta_3\}$ and $\{\theta_2, \theta_3\}$.

The following steps give a method to obtain all the redundant paths candidates with redundancy *n* from a source to a target (it is assumed that n is less than the maximum redundancy):

1) Search for all the paths from the source to the target according to Algorithm 1.(The parameters, paths, passedLinks and passedRouters, are initialized to empty)

2) Group every n paths of the result of setp 1. For each group, if its paths do not have sharing links, add the group of paths into the candidates.

III. Calculation of the Worst-case Latency

In a time slot, a packet is sent to any target of the channel stochastically. Hence the latency spent on transmitting all packets of a time slot is not unique. In order to guarantee the real-time constraint for a time slot, it has to obtain the worst-case latency (WCL) spent on transmitting all packets of the time slot.

Calculation method of the WCL is illustrated in this chapter. We will first explain calculation of latency of transmitting a single packet with a deterministic path. Then it is extended to show calculation of WCL of all packets in a time slot.

A. Latency of Transmitting a Single Packet

Assume a packet τ_i is transmitted with a deterministic path. The latency of τ_i is the sum of following elements:

- T^i_{smdt} : The delay occurred at the source before transmission of τ_i.
- T^i_{srt} : The time spent on sending τ_i completely.
- T^i_{rmdt} : The delay occurred at the target, which is the interval between τ_i is received and the reply of τ_i is started to send.
- T^i_{rrt} : The time spent on sending the reply of τ_i completely.

T^i_{smdt} and T^i_{rmdt} are determined by the source and target. T^i_{srt} and T^i_{rrt} are calculated by following formulas according to the type of τ_i, referring to the packet format specified in [2].

1) when τ_i is a RMAP write packet

$$T^i_{srt} = 10 \times \frac{(R_i + P_i + D_i + 17)}{S} + T_{pd} R_i$$
$$T^i_{rrt} = 10 \times \frac{(R_i + 8)}{S} + T_{pd} R_i \qquad (1)$$

2) when τ_i is a RMAP write packet

$$T^i_{srt} = 10 \times \frac{(R_i + P_i + 16)}{S} + T_{pd} R_i$$
$$T^i_{rrt} = 10 \times \frac{(R_i + D_i + 13)}{S} + T_{pd} R_i \qquad (2)$$

3) when τ_i is a RMAP write packet

$$T^i_{srt} = 10 \times \frac{(R_i + P_i + 25)}{S} + T_{pd} R_i$$
$$T^i_{rrt} = 10 \times \frac{(R_i + 17)}{S} + T_{pd} R_i \qquad (3)$$

In the formulas, D_i is the transmission data length. *S* is the minimum line speed (M bit/s) in the transfer section. R_i, P_i and T_{pd} are obtained according to the deterministic path of τ_i. R_i is the number of routers between the initiator and the target. P_i is the number of reply addresses (greater than R_i, and a multiple of 4). T_{pd} is network transmission delay time when packets are passing through the router.

B. WCL of Packets in a Time slot

When more than one packet exist in a time slot, according to the definition of channel, each packet may be transmitted according to different paths stochastically. Therefore, calculation of the WCL has to consider the combination of all the paths. For example, assume that a time slot has *N* channels $C_1, C_2, \ldots C_N$; each channel C_I has a set of packets $\{\tau_i\}_I$, and paths $\{\theta_i\}_I$. The combination thus is the Cartesian product of $\{\theta_i\}_1, \{\theta_i\}_2, \ldots, \{\theta_i\}_I$.

In each member of the combination, the packets are transmitted to a unique target according to a deterministic path. To calculate the WCL, first, we define segment as a group of packets, in which any packet has shared links with at least one anther packet. Therefore, the time spent on sending all the packets, defined as WCL of the segment, is the accumulation of their single latency, which is calculated by following formula.

$$WCL = \max_{0\le i\le n} T^i_{smdt} + \sum_{i=0}^{n} T^i_{srt} + \max_{0\le i\le m} T^i_{drmt} + \sum_{i=0}^{m} T^i_{rrt} \qquad (4)$$

In the formula, n is number of the packets; m is number of their replies.

Second, because the packets of different segment can be transmitted simultaneously without any interference on each other, the WCL of the time slot is calculated by obtaining the maximum of the WCL of the all segments.

Summarize above, we propose the calculation of WCL as follows:

1) *Calculate the Cartesian product of* $\{\theta_i\}_1, \{\theta_i\}_2, \ldots, \{\theta_i\}_l$.

2) Divide segments for each member of the Cartesian product.

3) For each segment, calculate its WCL based on formula (4).

4) Maximum of these WCL of segments thus is the WCL of the time slot.

IV. Proposed Scheduling Method

In this chapter, we propose the Simulated Annealing (SA) based scheduling algorithm. The scheduling algorithm choses redundant paths for the packets, assigns them to the time slot with respect to the purposes that it guarantees the packets in each time slot meeting the constraint, and keeps the number occupied slot number and the utilization as small as possible.

First, we give a brief introduction of the SA algorithm. It shows that the neighbor searching and evaluating are the two main parts of the SA for achieving the scheduling. The neighbor searching explores new solutions of the paths and time slot assignment, whereas the evaluating utilizes a function to judge the new solutions and decide which one is the best. Therefore, in the second and third part of this chapter, we will explain how the neighbor searching is conducted, and what is the function selected for the evaluating, respectively.

A. Simulated Annealing Algorithm

The SA algorithm is a generic probabilistic metaheuristic algorithm for the global optimization problem. The advantage of SA is that it can locate a good approximation to the global optimum from a given evaluating function in a large search space[5].

The pseudo code of the SA algorithm is given in Algorithm 2. It starts with an initial solution S_0 and performs a maximum of k_{max} steps. At each step, it utilizes *neighbor(S)* to generate a new solution S_{new}, and S_{new} will be evaluated by *EF(S)*. If the result of $EF(S_{new})$ is smaller than that of $EF(S_{cur})$ ($E_{new} < E_{cur}$), current solution S_{cur} is replaced by S_{new}. Otherwise, it decides whether or not to replace S_{cur} with S_{new} probabilistically. The probability is usually according to the Metropolis principle [6]

Algorithm 2 Simulated Annealing Algorithm

```
1:  $T \leftarrow T_0,\ S_{cur} \leftarrow S_0,\ E_{cur} \leftarrow EF(S_0)$
2:  $S_{best} \leftarrow S_{cur},\ E_{best} \leftarrow E_{cur}$
3:  $k \leftarrow k_{max}$
4:  while $k < k_{max}$ do
5:      $S_{new} \leftarrow neighbor(S_{cur}),\ E_{new} \leftarrow EF(S_{new})$
6:      $\Delta \leftarrow E_{new} - E_{cur}$
7:      if random[0, 1) < exp($\Delta$/T) then
8:          $S_{cur} \leftarrow S_{new},\ E_{cur} \leftarrow E_{new}$
9:      end if
10:     if $E_{new} < E_{best}$ then
11:         $E_{best} \leftarrow E_{new}$
12:     end if
13:     $k \leftarrow k + 1,\ T \leftarrow \alpha T$
14: end while
15: return $S_{best}$
```

that accepts S_{new} depending on the difference between E_{new} and E_{cur}, as well as on a global parameter T (line 7 of Algorithm 2). T denotes the temperature, and it is gradually decreased with α ($0 < \alpha < 1$) during the algorithm.

In general, the probability of selecting S_{new} will be decreased as the temperature T is reduced. This feature of SA algorithm is effective to move away from a local optimal solution and improve the probability of finding the global optimal solution. Finally, S_{best} corresponding to the E_{best} is generated as the best solution.

B. Neighbor Searching

Although there are two searching objects of the scheduling—redundant paths and time slots assignment, it is possible to conduct them in the different round of the *neighbor(S)*. For example, *neighbor(S)* searches for a new solution of redundant paths for a packet when the counter parameter k of SA is odd, whereas it searches for a new solution of time slot assignment when k is even.

To search for new redundant paths of a packet, it chose a packet randomly, then randomly changes its redundant paths from its redundant path candidates obtained by the method of Chapter II. On the other side, when searching for a new solution of time slot assignment, it choses a packet randomly, then move this packet to another randomly selected time slot.

C. Evaluating Function

Evaluating function is responsible for determining how "good" a solution is with respect to the optimization purposes— guarantee the packets in each time slot meeting the constraint, and keep the number of occupied time slot and the utilization as small as possible. Let us start from analyzing the effects that a solution may affect on the results. According to the neighbor searching approach, a new solution can be classified to positive and negative. A positive solution may has one of the following 4 effects:

1) Number of overflow time slots decreases.

2) Utilization of overflow time slots decreases.

3) Number of normal using time slots decreases.

4) Utilization of normal using time slots decreases.

In contrast, a negative solution may has one of the following 4 effects:

1) Number of overflow time slots increases.

2) Utilization of overflow time slots increases.

3) Number of normal using time slots increases.

4) Utilization of normal using time slots increases.

In the above, the normal using time slots are the time slots their utilization ≤ 1; and overflow time slots are the time slots their utilization > 1. The utilization of time slot *i* is ratio of the WCL to the length of time slot *i*.

The evaluating function should be able to precisely reflect the solution effects according to its results. Therefore, we chose it as follows.

$$EF(S) = (N_l + 1)\sum_{i \in l_o} U_i^S + N_{l_u} + \prod_{i \in l_u} U_i^S \quad (5)$$

In the formula, U_i^S is the utilization of time slot *i* in solution *S*. l, l_u, l_o denote all time slots, normal using time slots, and overflow time slots, respectively. N_l, N_{lu} denote the number of all time slots and normal using time slots, respectively.

According to formula (5), positive 1) is corresponding to the best result—the smallest value of the formula among the 8 situations, and then is the positive 2). The worst is the negative 4).

V. Experiments and Results

To evaluate the effectiveness of the proposed scheduling method, we utilize the method to schedule packets of a SpaceWire system and show the results in this chapter. The evaluated SpaceWire system is provided by a Japanese aeronautical organization, which is composed of 4 routers and 4 nodes. There are 4 channels in the system, which transmit total 86 packets. Each of the channels has one source and 2 targets. For the source to each target, redundant paths with redundancy 2 are provided.

The results are shown in TABLE.I. It is assumed that the system time is divided into 8 time slots and repeats infinity. Length of each time slot is 5ms. The left two columns are the results of randomly scheduling—paths and time slots of all the packets are assigned randomly. The right two columns are the results of the proposed scheduling method. The results show that two time slots, time slot 3 and 7, do not meet the real-time constraint under randomly scheduling. Whereas the proposed scheduling method guarantees the constraint for all the time slots, and the occupied time slots number is decreased— time slot 4 has not been used.

TABLE I. Results of the Proposed Scheduling Method

Randomly Scheduling		**Proposed Scheduling**	
Time slot	Utilization	Time slot	Utilization
0	0.724	0	0.9156
1	0.913	1	0.7728
2	0.532	2	0.9488
3	1.139	3	0.87
4	0.471	4	0
5	0.631	5	0.959
6	0.588	6	0.991
7	1.265	7	0.657

VI. Conclusion

In this paper, we proposed a Simulated Annealing (SA) based scheduling method for RMAP packets. The method choses redundant paths for the packets, assigns them to the time slot with respect to the real-time constraint. The method is evaluated by the simulations. The results show that the proposed scheduling method guarantees the constraint for all the time slots, and decreases the number of occupied time slots as well as the utilization.

In future work, we plan to examine the scheduling algorithm in more complex systems. Moreover, the investigation of configuring the SA parameters in order to obtain better performance is also considered.

VII. References

[1] ECSS, "SpaceWire – Links, nodes, routers and networks" ECSS-E-ST-50-12C, 31 July 2008.

[2] ECSS, "SpaceWire – Remote memory access protocol", ECSS-E-ST-50-52C, 5 February 2010.

[3] S. Parkes, A Ferrer, S. Mills, A. Mason, "SpaceWire-D:Deterministic Data Delivery with SpaceWire", International SpaceWire Conference, St Pertersburg, Russia, June 2010.

[4] M.Takada, H.Takada, Y.Chen, T.Yuasa, T.Takahashi, M.Nomachi, "Development of Software Platform Supporting a Protocol for Guaranteeing the Real-Time Property of SpaceWire", International SpaceWire Conferenece, Gothenburg, Sweden, June 2013.

[5] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi: Optimization by simulated annealing,Sci., vol.220, no.4598, pp.671-680(1983).

[6] Metropolis, Nicholas; Rosenbluth, Arianna W.; Rosenbluth, Marshall N.; Teller, AugustaH.; Teller, Edward.: Equation of State Calculations by Fast Computing Machines.The Journal of Chemical Physics 21 (6): 1087. doi:10.1063/1.1699114.

Real-time Data Recording System with SpaceWire for Asteroid Sample Return Mission HAYABUSA2

SpaceWire onboard equipment and software, Poster Paper

Satoko Kawakami, Yasuhiro Takeda, Hiroki Hihara
NEC TOSHIBA Space Systems. Ltd.
10, Nisshin-cho 1-chome, Fuchu, Tokyo, Japan
s-kawakami@bk.jp.nec.com, y-takeda@ei.jp.nec.com, h-hihara@bk.jp.nec.com

Tetsuya Masuda, Masatoshi Ebara
Space Systems Division
NEC Corporation
10, Nisshin-cho 1-chome, Fuchu, Tokyo, Japan
masuda@jd.jp.nec.com, m-ebara@cw.jp.nec.com

Ryu Funase
Department of Aeronautics and Astronautics
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
funase@space.t.u-tokyo.ac.jp

Takahiro Yamada
Institute of Space and Astronautical Science (ISAS)
Japan Aerospace Exploration Agency (JAXA)
Sagamihara, Chuo-ku, Kanagawa 229-8510, Japan
tyamada@pub.isas.jaxa.jp

***Abstract*—A real-time recording system which adopted SpaceWire and RMAP (Remote Memory Access Protocol) has been developed for an asteroid probe HAYABUSA2. It exploits the deterministic scheduling scheme formalized for SpaceWire by SpaceWire-D draft standard. Since the captured precise image of the surface of a target asteroid is to be recorded during autonomous touch-down sequence, the recording system synchronized with the attitude and orbit control subsystem (AOCS) with the deterministic scheduling scheme. The scheme has also enabled the data recorder to inherit the priority based packet recording function developed for prior HAYABUSA without any change.**

***Index Terms*— SpaceWire, Data Recorder, Networking, Spacecraft Electronics.**

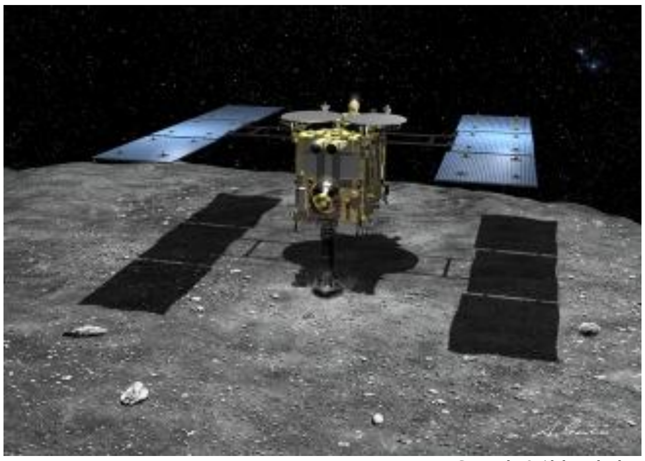
I. Introduction

HAYABUSA2 is an asteroid probe planned to be launched in 2014, and aims at sample-return from a C-type asteroid considered to contain organic or hydrated materials. Figure 1 shows the image drawing of HAYABUSA2 on an asteroid.

The precise captured image of the surface of a target asteroid is to be recorded in the onboard storage during the autonomous touch-down and take-off sequence in parallel. Therefore the real-time recording capability synchronized with the attitude and orbit control subsystem (AOCS) is required.

The transmission system between the data recorder and AOCS are based on the network design criteria developed for JAXA/ISAS scientific satellites [1], [2], [3], [4]. The deterministic implementation scheme of SpaceWire, which has been established by SpaceWire-D draft standard, enabled synchronous operation between satellite bus system and mission payload.

Real-time synchronized image capturing system with optical sensors developed for scientific purposes and a data recorder is realized by adopting deterministic implementation scheme.



Copyright © Akihiro Ikeshita

Fig. 1. **HAYABUSA2 asteroid probe (image drawing)**

II. Data Recorder Architecture

Data Recorder (DR) design is closely tied to the mass memory modules, which are high-density memory modules with unique stacking technology and optimized for SpaceWire/RMAP interface devices. The stack structure of DR is fabricated with bus connectors, which resulted in eliminating back planes.

Figure 2 shows the outlook of DR. Its A6 size has been realized by exploiting following technologies;

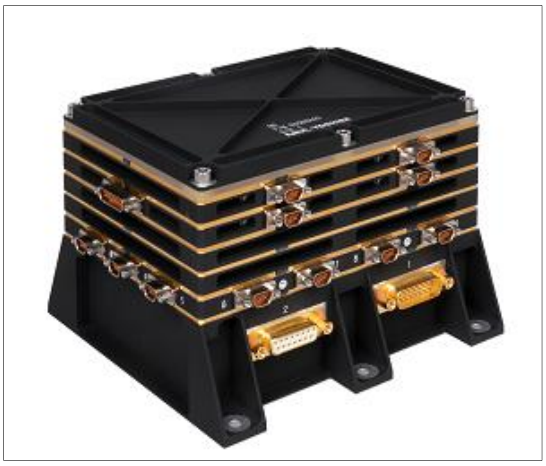


Fig. 3. The outlook of DR

- Stack memory module was developed and qualified through JAXA programs. The memory module comprises eight 512Mbit SDRAM chips manufactured by ELPIDA, and stacked through NEC's low profile JAXA-authorized stacking technology.
- A6 size memory board provides 16Gbits on a single board with additional memory modules for Reed-Solomon CODEC equipped on the same board. Single symbol (8 bits) error in SDRAM induced by radiation effects is corrected automatically and transparently by scrubbing the memory (i.e., reading out values and rewriting correct ones).
- Priority based file access function through simultaneous record and reproduce operation are provided.
- Each module has built-in current monitoring circuitries against single event latch up.
- Fully SpaceWire compatible interface with RMAP function has been realized with an embedded SpaceWire router, which enables congestion-free data collection both for pull and push scheme.

The data recorder has an original protocol stack which is complied with the SpaceWire-D draft standard protocol stack [5], [6]. Figure 3 shows the data recorder protocol stack and the SpaceWire-D draft standard protocol stack. The figure shows that the data recorder handles RMAP in deterministic way. The data recorder has no implementation for SpaceWire-D and SpaceWire-R in itself, and these two protocol layers are accommodated in the optical navigation camera electronics (ONC-E) and the sensor digital electronics (DE). ONC-E and DE compile mission data collection plan, and the plan is delegated to the DR for its initiator mode operation.

In order to exploit the deterministic implementation scheme with SpaceWire and RMAP protocol, the priority based file system is provided. The data structure consists of a volume group (or record), a memory manager, logical volumes (or files), physical extents (or partitions) and blocks (or sectors). These are configurable by commands prior to the onboard operation of the DR through ground station operation. The architecture is shown in figure 4. Volume group (or record) is an entire region to compose a system. This supports subordinate concepts of logical volumes and is implicitly given. Memory manager configures its recording memory area. It also arranges volume groups. Logical volumes (or files) are variable length storage area which consists of physical extents. Physical extents (or partitions) are continuous region of variable length area, which are composition elements of logical volumes. Physical extent consists of blocks. Blocks (or sectors) are fixed length size memory are, and they are

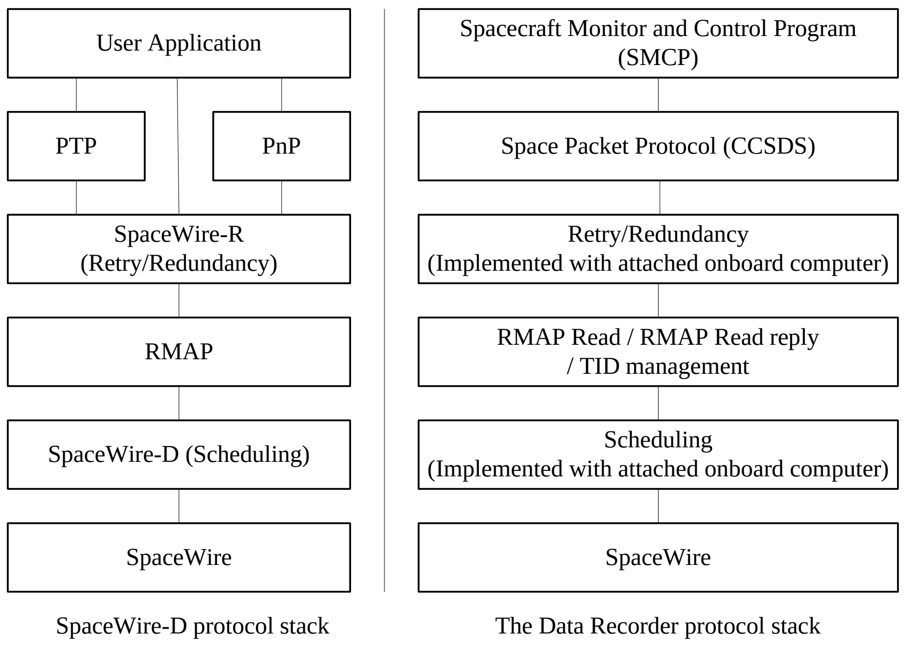

User Application
PTP
PnP
SpaceWire-R
(Retry/Redundancy)
RMAP
SpaceWire-D (Scheduling)
SpaceWire
SpaceWire-D protocol stack
Spacecraft Monitor and Control Program
(SMCP)
Space Packet Protocol (CCSDS)
Retry/Redundancy
(Implemented with attached onboard computer)
RMAP Read / RMAP Read reply
/ TID management
Scheduling
(Implemented with attached onboard computer)
SpaceWire
The Data Recorder protocol stack

Fig. 2. The communication protocol layer of DR

minimum recording units and composition element of physical extents.

Since the priority based packet recording function are inherited from prior HAYABUSA asteroid probe, the file system is configured as fixed length partitions. Each partition corresponds to a category number, which is shown in the secondary header of Space Packets. The category number is associated with the priority for recording and reproducing. The priority mechanism is developed for prior HAYABUSA asteroid probe with its original communication protocol. Whereas thanks to the deterministic scheme of the data recorder, its inherited priority based recording and reproducing function of prior HAYABUSA are applied without any change with SpaceWire/RMAP protocol..

Technical features of DR are shown in table 1.

TABLE I. DATA RECORDER TECHNICAL FEATURES

Parameter	Value
Memory Size	Synchronous DRAM: 16Gbits at BOL 12Gbits at EOL
Recording data rate	39Mbps [Effective value] 15Mbps [Requirement]
Reproducing data rate	25Mbps [Effective value] 15Mbps [Requirement]
SpaceWire port	Telemetry/Command: 2ch Recording: 7ch Reproducing: 2ch
Size	142(W) x 150(D) x 107(H) (mm)
Mass	< 2.32kg
Power consumption	< 11.5W

III. REAL-TIME RECORDING SYSTEM

Figure 5 shows the diagram of the real-time recording system of HAYABUSA2. The data handling subsystem of HAYABUSA2 inherits the prior HAYABUSA asteroid probe. HAYABUSA employed original onboard communication protocol PIM (Peripheral Interface Module transmission protocol). So we developed protocol bridges for the translation between PIM and SpaceWire. The protocol for accessing the interface of each components and the scheduling scheme is close to RMAP (Remote Memory Access Protocol) and SpaceWire-D, so the development of those protocol bridges were straight-forward.

Although the AOCS of HAYABUSA2 accommodates SpaceWire interfaces and connected to legacy onboard devices with PIM interfaces through protocol bridges, the operation scheme is the same as its predecessor. As a result the ground station operation is the same as that for prior JAXA/ISAS scientific satellites.

Sensors are controlled by the DE and ONC-E. DE and ONC-E have communication lines with the attitude and orbit control processors (AOCPs) and the DR. These lines features the deterministic implementation specified in SpaceWire-D draft standard. DE and ONC-E are working synchronously with AOCPs and which enables synchronous operation of sensor control functions and data recording. The scheme has been established through the activity of the SpaceWire user's group, Japan [7].

Thanks to SpaceWire backplane implementation fabricated

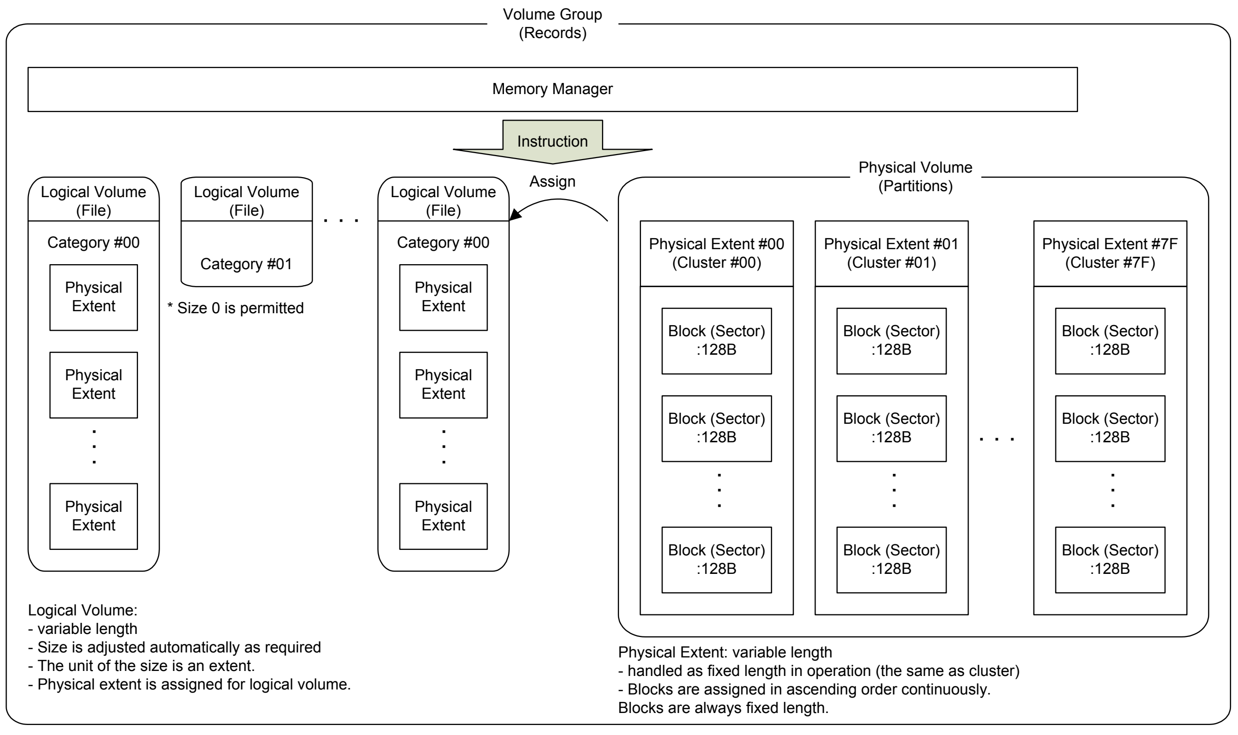


Fig. 4. The data structure configuration of DR

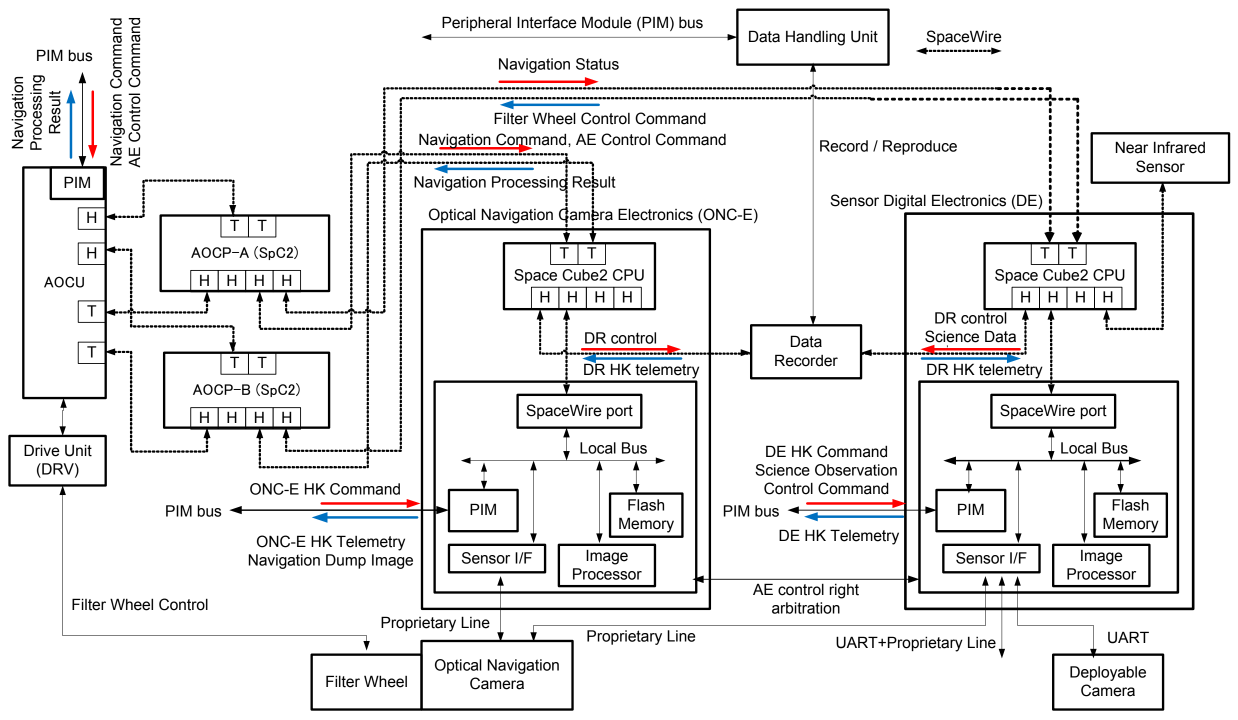


Fig. 5. Synchronous communication block diagram of AOCS and DR

in the DE and the ONC-E, image buffer memory located inside the sensor signal processor is connected with mass memory modules in the data recorder through SpaceWire and RMAP protocol. Even though the size of the scientific mission image captured by optical sensors are changed, the synchronization scheme between image memory buffer inside the sensor signal processor and the data recorder is maintained through the deterministic communication implementation, and no image is to be lost with the priority based flexible file system of the data recorder

ACKNOWLEDGMENT

Authors thank ASTRO-H project people for their valuable suggestions for implementing real-time system for asteroid probe mission.

REFERENCES

[1] Tadayuki Takahashi, et al., "The ASTRO-H Mission", SPIE, 7732, 77320Z, 30 July 2010.

[2] Takahiro Yamada, and Tadayuki Takahashi, "Standard Onboard Data Handling Architecture Based on SpaceWire", International SpaceWire Conference 2008, 4-6 November 2008, p.253-256.

[3] Takayuki Yuasa, Tadayuki Takahashi, Masanobu Ozaki and Motohide Kokubun, "A Deterministic SpaceWire Network Onboard the ASTRO-H Space X-Ray Observatory", International SpaceWire Conference 2011, 8-10 November 2011, p.348-351.

[4] Hiroki Hihara, Toshiaki Ogawa and Kenji Kitade, "NEXTAR: Small Satellite Bus Based on SpaceWire Deterministic Implementation", International SpaceWire Conference 2011, 8-10 November 2011, p.344-347.

[5] Satoko Kawakami, Kazuyuki Yamada, Hiroki Hihara, Masaharu Nomachi, Takahiro Yamada, Motohide Kokubun and Tadayuki Takahashi, "Deterministic Implementation of SpaceWire on Data Recorder and Payload Interface Units", International SpaceWire Conference 2011, 8-10 November 2011, p.182-185.

[6] Takahiro Yamada, "Proposal for Defining Standard Services Over SpaceWire –Revision A -", The sixteenth SpaceWire working group meeting ESTEC, Netherlands, 22 March 2011.

[7] SpaceWire User's Group, Japan, "SpaceWire Network Design Guideline", Version 1.0, 13 May 2010.

Radiation Tolerant SpaceWire Remote Terminal Controller ASIC (RMR-02P)

SpaceWire Components, Poster Paper

Aleksey Sakharov, Dmitri Skok, Vladimir Gusev,
Tatiana Solokhina, Jaroslav Petrichkovich
"ELVEES" RnD Center
Zelenograd, Moscow, Russian Federation
asaharov@elvees.com, dskok@elvees.com,
vgoussev@elvees.com, tanya@elvees.com,
slava@elvees.com

Yury Sheynin, Elena Suvorova
St. Petersburg University of Aerospace Instrumentation
St. Petersburg, Russian Federation
sheynin@aanet.ru, suvorova@aanet.ru

***Abstract*—The paper considers radiation tolerant remote terminal controller with serial SpaceWire communication channel, that used for connection of peripheral devices to SpaceWire network. Its architecture, main features and various system applications are presented. The results of tests are introduced.**

***Index Terms*— Remote terminal controller, SpaceWire, peripheral interfaces.**

I. Introduction

The, SpaceWire technology [1] is getting the most used and intensively developed technology for spacecraft computer networks as it enables to build a communication infrastructure for all onboard equipment on basis of single network.

Design of on-board systems on the SpaceWire technology developers has to solve the problem of connecting different peripheral devices to a SpaceWire network. Usually modern (all the more the old ones) electronic devices do not have opportunity to connect into SpaceWire network directly. So, the first question that engineers ask is "What should I use to integrate my nice device into a SpaceWire network?" After some investigation, in many cases the second question rises itself: "Why should I use this big bridge to integrate my very nice device into a SpaceWire network? Ok, this bridge is very intellectual and even has a microprocessor inside, but it's n-th times bigger and heavier than my very very nice device…"

In this article a simple Remote Terminal Controller ASIC RMR-02P is suggested to solve the problem of integrating peripheral devices in SpaceWire network. The controller provides very efficient, low power opportunity to provide connection of a wide range of peripheral devices, such as ADC, DAC, FLASH memory, different controllers and sensors, to high throughout, noise-immune SpaceWire network without using extra components (microcontrollers, memory circuits, FPGA, receiver-transmitters etc).

II. Application

The SpaceWire remote terminal controller ASIC RMR-02P provides light-weight SpaceWire connectivity for simple slave devices: ADC, DAC, peripheral controllers, sensors, actuators, etc. Due to hardware implementation of the protocols and flexible slave device interfaces the RMR-02P does not require external memory and glue logic chips. No special software programming is needed as well.

The RMR-02P has the following features:

- Dual-port ECSS-E-50-12C SpaceWire controller
- SpaceWire data rate from 2 to 300 Mb/s
- Built-in ANSI/TIA/EIA-644 LVDS transceivers with 100 Ohm impedance matching resistors
- Built-in hardware implemented RMAP (ECSS-E-ST-50-52C) and Distributed Interrupt Protocol
- Two programmable universal 16-bit parallel ports, supported different modes: master, slave (mailbox), GPIO/SPI-master
- Maximum data rate of the parallel port 32 MB/s
- Maximum data rate of the SPI-master port 25 Mb/s
- 16-bit Intel/Motorola microcontroller interface
- Maximum data rate of the microcontroller interface 32 MB/s
- Ceramic 112-pin package 26.7x26.7 mm

III. Description

The simplified functional diagram of the RMR-02P is shown on the figure 1.

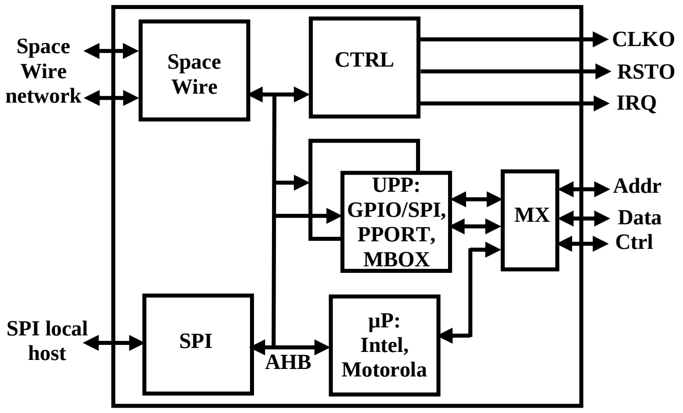


Figure 1: RMR-02P functional diagram

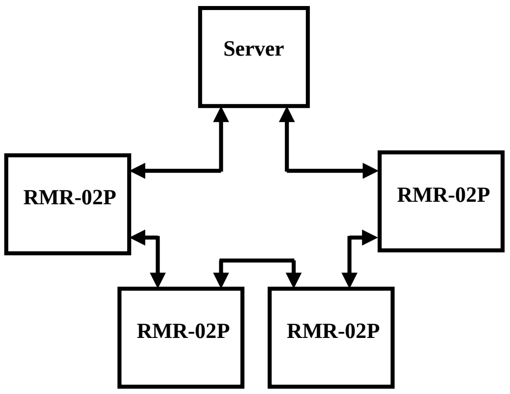


Figure 3: Ring connection

The RMR-02P contains the followingfunctional blocks:

- SpaceWire – dual-port SpaceWire controller
- UPP – universal parallel ports
- µP – microcontroller interface
- MX – IO multiplexer
- CTRL – control circuitry
- SPI – debugging slave serial port interface

The dual-port SpaceWire controller provides connectivity to the SpaceWire network. Built-in ANSI/TIA/EIA-644 LVDS transceivers with 100 Ohm impedance matching resistors simplify system integration. The controller supports two protocol for control and data transfer implemented in hardware: Remote Memory Access Protocol (RMAP) [2] and Distributed Interrupt Protocol (DIP) [3].

Two SpaceWire ports are designed to be used in high reliable systems with redundant SpaceWire connections. Both ports are equivalent. An internal router transfers data between the SpaceWire ports and chip’s internal bus as well as between the SpaceWire ports themselves. This allows to build different redundant connection structures, e.g. Star (see figure 2) and Ring (see figure 3). In the both cases, single hop failure does not fail system connectivity. The main advantage of the Ring connection structure is much smaller cable-ware than for the Star one.

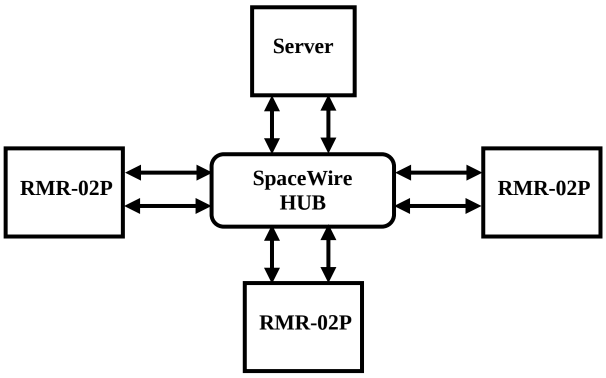


Figure 2: Star connection

Two 16-bits universal programmable parallel ports (UPP) provide connectivity to a wide range on external passive (i.e. without microprocessor or microcontroller) devices. Each of them can be independently programmed in the following modes:

- Master mode
- Slave mode (mailbox)
- GPIO/SPI mode

In the Master mode, the UPP operates as a 16-bit bidirectional parallel master ports with Intel-style control signals, ready signal and interrupts request signal. The waveform of the port is programmable: setup and hold times, CS signal duration, idle time between accesses, access timeout can be configured via the SpaceWire ports. A 128-entry internal FIFO allows RMAP packets of up to 256 byte payload. Another FIFO related feature is early read by request or FIFO readiness. The distributed interrupts can be generated by the number of events: external interrupt signal, FIFO readiness, FIFO overflow/underflow error, access timeout expired.

In the Mailbox mode, the UPP operates as a 16-bit bidirectional parallel slave ports with Intel-style control signals, ready signal and interrupts request signal. Many features of the Master mode - programmable waveform, internal FIFO, external interrupts request are similar to the Master mode. The difference is that in Mailbox mode the external device writes to and reads the RMR-02P port.

In the GPIO/SPI-master mode, the UPP operates as SPI master and 16 general purpose IO signal. Each of the 16 GPIOs can be independently configured as an input, output or SPI chip select signal (i.e. up to 16 chip select signals can be used for SPI). The distributed interrupts can be generated on state changing of any GPIO signal configured as an input as well as SPI operation finishing.

The microcontroller interface of the RMR-02P has 16 bit data bus, 16-bit address bus, Intel or Motorola-style control signals with a READY/ACK signal, two chip select signals and two external interrupts signals. The interface operated as a master such that the RMR-02P can simply replace microcontroller. The interface shares the same pins as the parallel ports, so either the microcontroller interface or the parallel ports can be used. The waveform of the interface is programmable like in done in the parallel master port: setup and hold times, CS signal duration, idle time between accesses and access timeout are configurable.

The microcontroller interface of the RMR-02P supports RMAP packets of up to 256 byte payload with address increment and without address increment.

The RMR-02P has two special signals controlled via the SpaceWire ports which facilitate system integration. The RSTO output signal is pulse generating signal under SpaceWire control. It is designed as a reset signal for external devices. The duration and the polarity of the RSTO signal are programmable.

The CLKO output is continues clock generating signal. It is designed as a clock signal for external devices. The period, on and off state are controlled via the SpaceWire port. The maximum and minimum period of the CLKO is 25 MHz and 0.4 MHz.

The ancillary SPI-slave port is designed as a debugging interface. It can be used to control the RMR-02P instead of the SpaceWire ports.

IV. Results

The RMR-02P was designed on Radiation Tolerant standard cell and memory libraries of R&D Center ELVEES. It was manufactured on 180 nm CMOS process and assembled in 112-pin ceramic package (see figure 4).

The chip has a small area and low power consumption (see table 1). Power consumption was measured at maximum SpaceWire data rate and at supplies voltage of peripheral buffers and digital core equal 1.8 and 3.3V consequently.

Testing of the chips demonstrated full functionality and good performance results (see table 2 and table 3). The TID and SEU/SEL test are to be done as the next step.

The RMR-02P chips are available for evaluation.

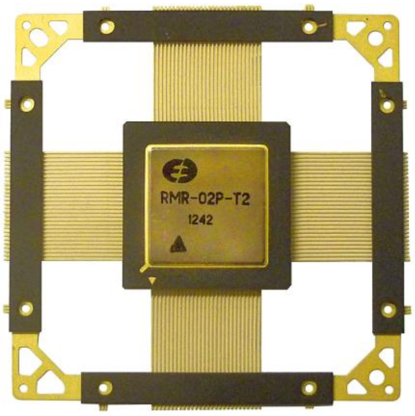


Figure 4: RMR-02P in CQFP-112 package

TABLE I. Area and Power Consumption

Instance	***Area, mm***	***Power, mW***
SpaceWire RMAP IP CORE	2.1×2.0	95
RMR-02P	4.0×4.0	230

TABLE II. SpaceWire Link Data Rate

Cable Length, m	**Maximum SpaceWire rate, Mbit/s**	
	Receive	***Transmit***
1	300	300
10	300	300
20	200	100

TABLE III. Peripheral Interfaces Data Rate

Peripheral interface	**Maximum data rate, Mbit/s**
SPI-master	25
Parallel port	256
Mailbox	200
μP	256

V. Conclusion

The article presents the RMR-02P ASIC which has been designed for connecting a wide range of peripheral devices into SpaceWire networks. The RMR-02P ASIC has been manufactured on 180 nm CMOS process and successfully tested.

Currently, the RnD Center ELVEES works on the next generation of a remote terminal ASIC. In particular, it will include support of the Streaming Transport Protocol [4] for high efficient integration devices which generate or consume continuous data flow, e.g. fast ADC and DAC.

References

[1] ECSS_E_50_12C. SpaceWire – Links, nodes, routers and networks. – European Cooperation for Space Standardization (ECSS), 2008

[2] ECSS-E-ST-50-52C. SpaceWire – Remote memory access protocol. – European Cooperation for Space Standardization (ECSS), 2010.

[3] Sheynin Y., “Distributed Interrupts in SpaceWire Interconnections”, 8th SpW WG meeting, Noordwijk, January 2007

[4] Yuriy Sheynin, Elena Suvorova, Felix Schutenko, Vladimir Goussev, “Streaming Transport Protocols for SpaceWire Networks”, Proceedings of the 3rd International SpaceWire Conference, June 2010.

DCNSimulator – Software Tool for SpaceWire Networks Simulation

Session: SpaceWire networks and protocols

Poster Paper

Artur Eganyan, Elena Suvorova*, Yuriy Sheynin*, Alexey Khakhulin**, Igor Orlovsky***

**) Saint-Petersburg State University of Aerospace Instrumentation*
Saint-Petersburg, Russian Federation
artfla@rambler.ru, suvorova@aanet.ru, sheynin@aanet.ru

***) Rocket and Space Corporation Energia after S.P. Korolev*
4A Lenin Street, Korolev, Moscow area, 141070, Russia
Alexey.Hahulin@rsce.ru, Igor.Orlovsky@rsce.ru

***Abstract* — In the paper we present the Digital Communication Network Simulator (DCNSimulator) – a tool for design, system-level simulation and analysis of networks. We show how we have used this tool to simulate and analyze a real onboard SpaceWire network from Russian missions and projects.**

I. Introduction

The Digital Communication Network Simulator (DCNSimulator) is a tool for design, system-level simulation and analysis of networks.

DCNSimulator is based on Qt and SystemC. It consists of the simulation engine and libraries of network components. The simulation engine is the general part that could work for simulation of any network. Libraries of network components are specific for particular network standards and could represent network components at various details level – from general virtual components to cycle-accurate models of particular devices. Simulated device models are written in C++. Application software algorithms could run at end nodes thus generating realistic traffic for the simulated network. The simulator also allows users to design networks graphically in MS Visio. The DCNSimulator runs in Windows and doesn't require any other third party software for its operation.

The previous version of this tool was described in more details in [1]. In the current DCNSimulator version we have improved simulator efficiency, performance and usability. We have also added more detailed information of channels workload and search of particular packet by different parameters: content, sending or receiving time, latency interval. An example of a real spacecraft network investigation with the DCNSimulator is presented in the article.

II. SpaceWire Networks

The DCNSimulator with its library completely supports SpaceWire networks. It implements all levels of the SpaceWire standard (excluding signal and physical ones) and provides models of a terminal node, a routing switch and a channel (parameterized point-to-point link), from which networks can be composed. It also supports error imitation for channels and devices.

With this tool, SpaceWire networks can be analyzed at the levels of characters and packets. So one can analyze control codes and data packets propagation, channel workload and all errors occurred in channels. The simulator displays appropriate charts, statistics and information about every transferred code and packet.

III. Case Study

Here we show an example of the real network which was analyzed with the simulator. This is a fragment of a spacecraft network shown in fig. 1. Its photo is in fig. 2, and traffic of this network is described in table 2. The abbreviations are explained in table 1.

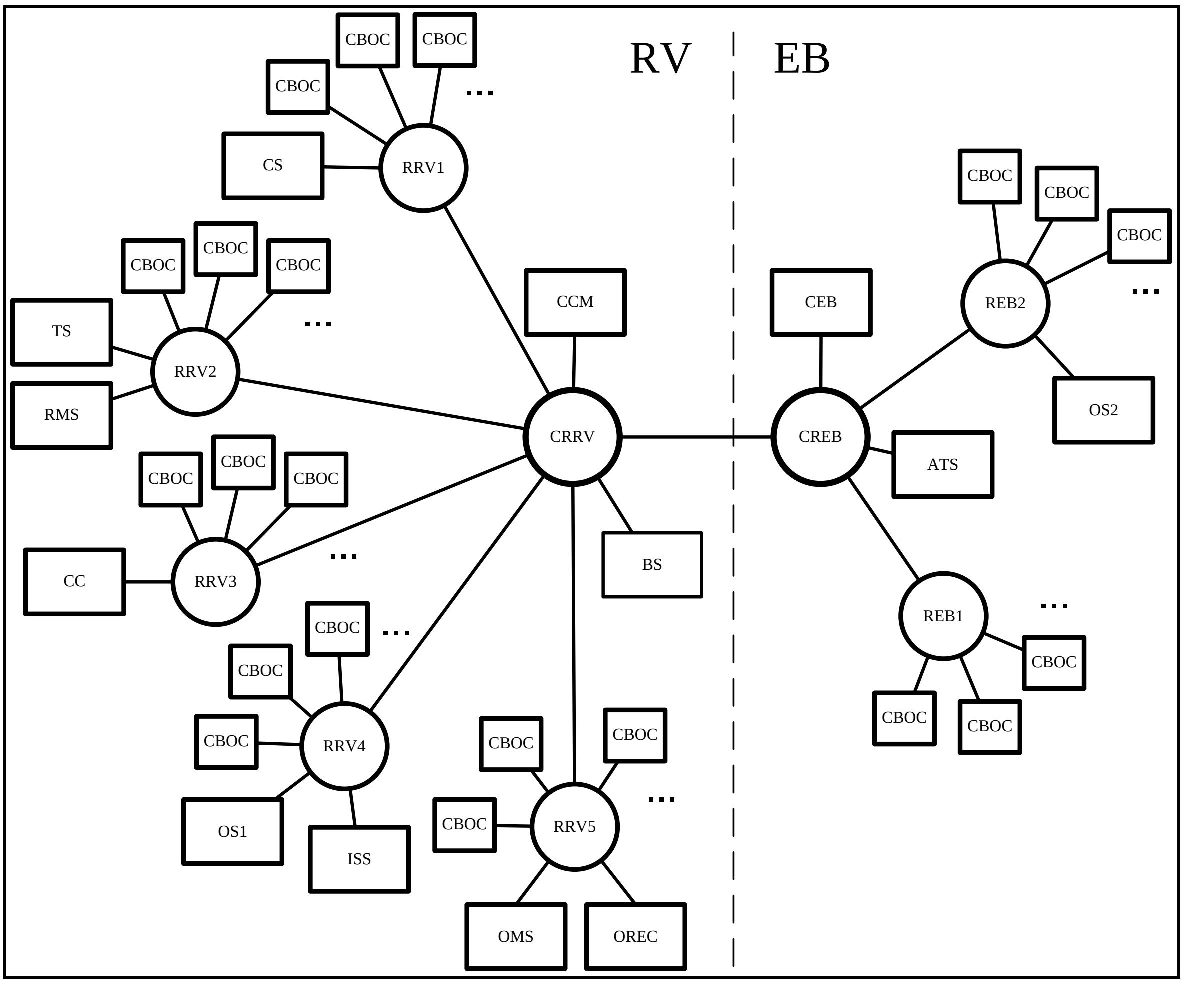


Figure 1 Fragment of spacecraft network (rectangles are terminal nodes and rounds are routers)

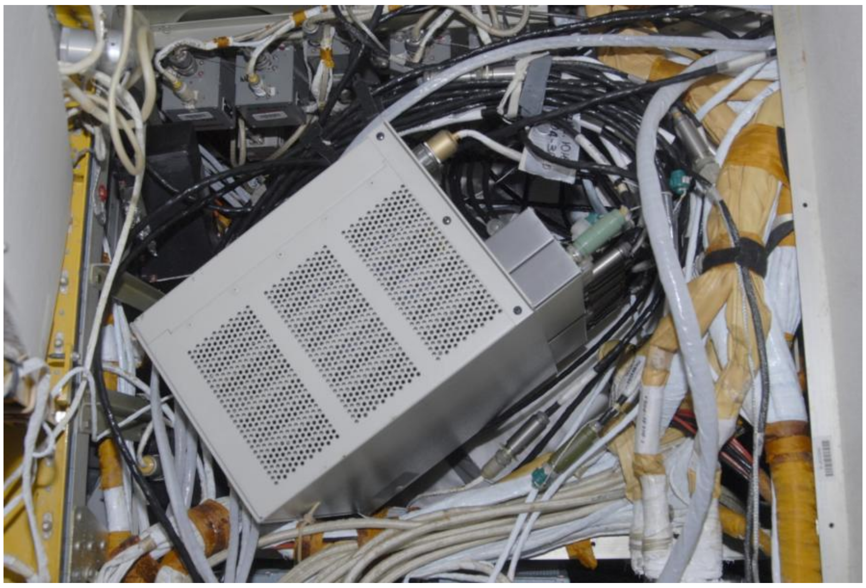


Figure 2 A fragment of a conventional spacecraft network cabling (photo)

TABLE I. ABBREVIATIONS

ATS	Automated Test System
OREC	Onboard Radio Engineering Complex
OS	Onboard Systems
CBOC	Control Block of Onboard Complex
RV	Re-entry Vehicle
EB	Engine Bay
CEB	Computer of the EB
RRV	Router of RV
REB	Router of EB
ISS	International Space Station
CC	Cosmonaut Consoles
OMS	Onboard Measurement System
RMS	Radiation Monitoring System
BS	Bearing System
CS	Communication System
TS	Telemetric System
CCM	Central Computing Machine
CRRV	Central RRV
CREB	Central REB

TABLE II. NETWORK TRAFFIC, IN MBIT/S

Receivers / Senders	ATS	OREC	CBOC	CEB	ISS	CC	OMS	TS	CCM
ATS	-	-	-	-	-	-	-	-	-
OREC	-	-	-	-	-	2	-	-	5
OS	-	-	-	-	-	1	1	-	3
CBOC	-	-	-	-	-	-	0.5	-	0.08
CEB	-	-	-	-	-	0.1	2	-	2
ISS	-	-	-	-	-	25	3	-	2
CC	25x3	-	-	-	-	-	2	-	0.2
OMS	5	-	-	-	3	0.1	-	-	5
RMS	-	-	-	-	-	1	1	-	1
BS	-	-	-	-	-	-	0.1	-	-
CS	-	-	-	-	-	-	0.2	-	-
TS	-	25	-	-	25	25	0.1	-	0.01
CCM	5	5	0.08	2	2	5	1	0.01	-

Channels rate is 125 Mbit/s. Any 25 Mbit/s and 25x3 Mbit/s in the table is a maximum video traffic which is generated all the time (uniformly) and transferred as frames of 1 Kbyte length. The rest traffic is generated every 200 ms. Packets between CBOCs and CCM have the highest priority. Each CBOC sends and receives to/from CCM one packet of 64 byte length, every 200 ms. There are 4 CBOCs on each RRV and 5 CBOCs on each REB.

Our goal was to verify that:

1. Latency of packets between CBOCs and CCM is $\leq$ 10 ms.
2. Latency of video frames is $\leq$ 100 ms.
3. The network is not overloaded.

We have created the network model with the DCNSimulator and simulated it for 1 second. We supposed that any big message is transferred as a group of 1 Kbyte packets. The general chart which displays latency of all packets is shown on fig. 3. This chart was expected because a lot of traffic is simultaneously sent every 200 ms. It also shows that the network is highly loaded every 200 ms but not overloaded – otherwise we would see continuously increasing latency.

Video frames (which are sent all the time) are transferred as shown on fig. 4. So latency of any video frame doesn't exceed 100 ms, but it "jumps" because video has no priority. In larger scale (fig. 5) and by statistic (fig. 6) we can see that the most of the frames are transferred even no longer than 0.5 ms (0.14 ms in average).

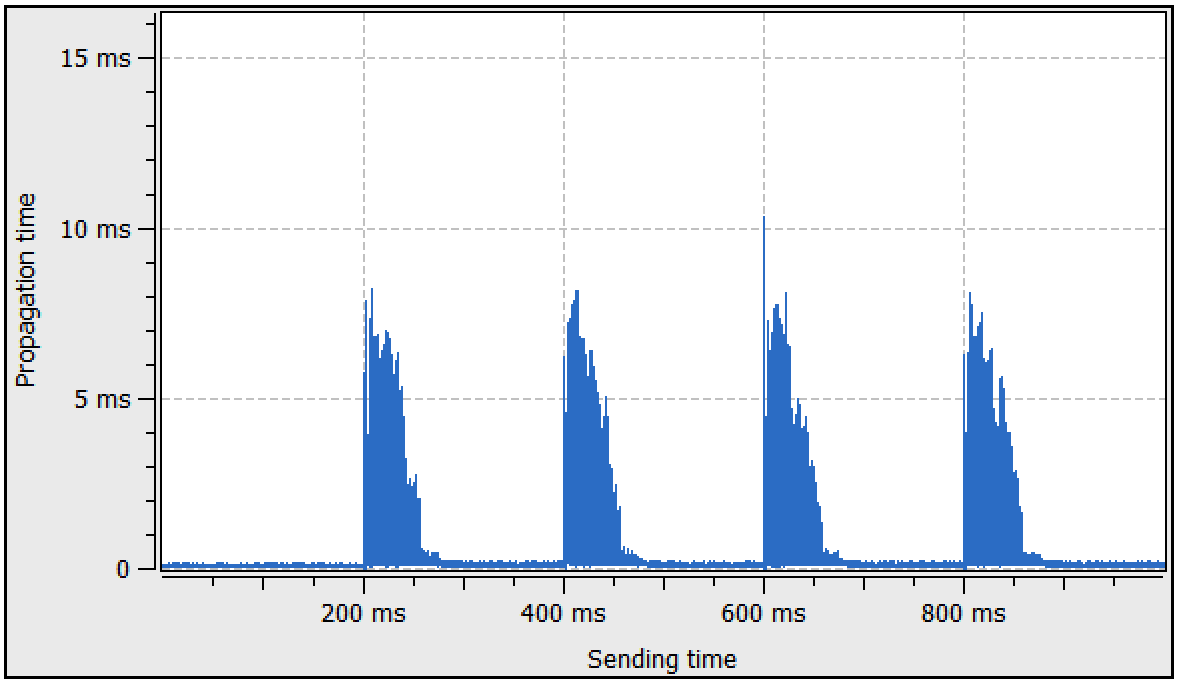


Figure 3 Latency of all transferred packets

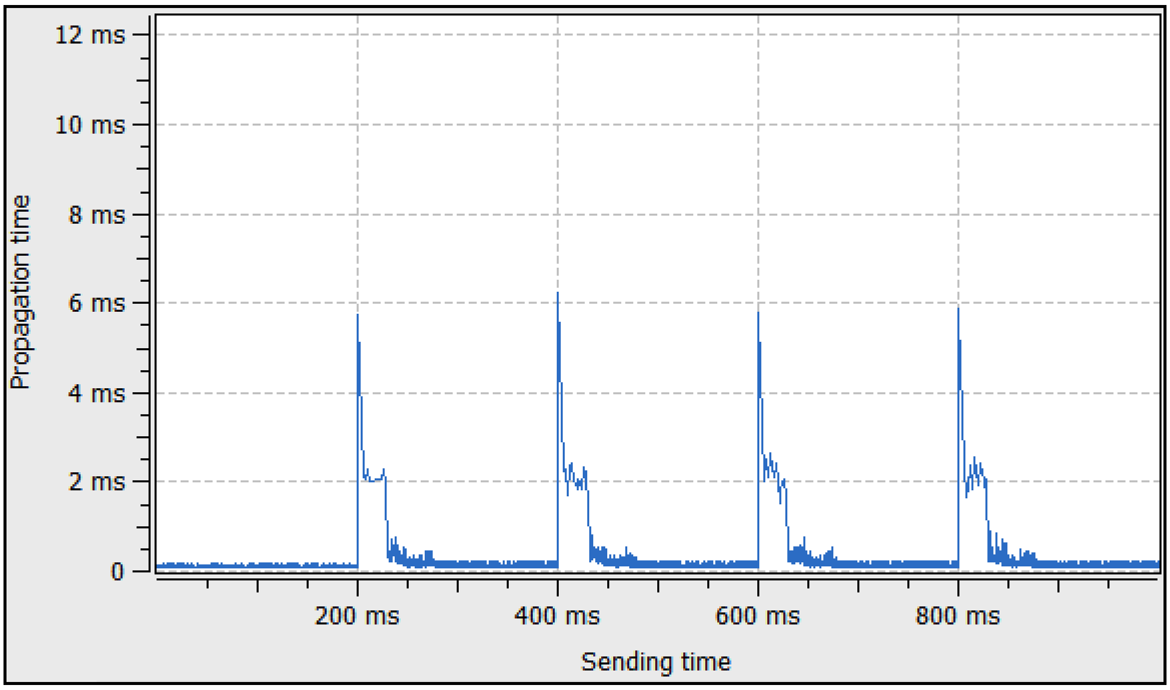


Figure 4 Latency of video frames

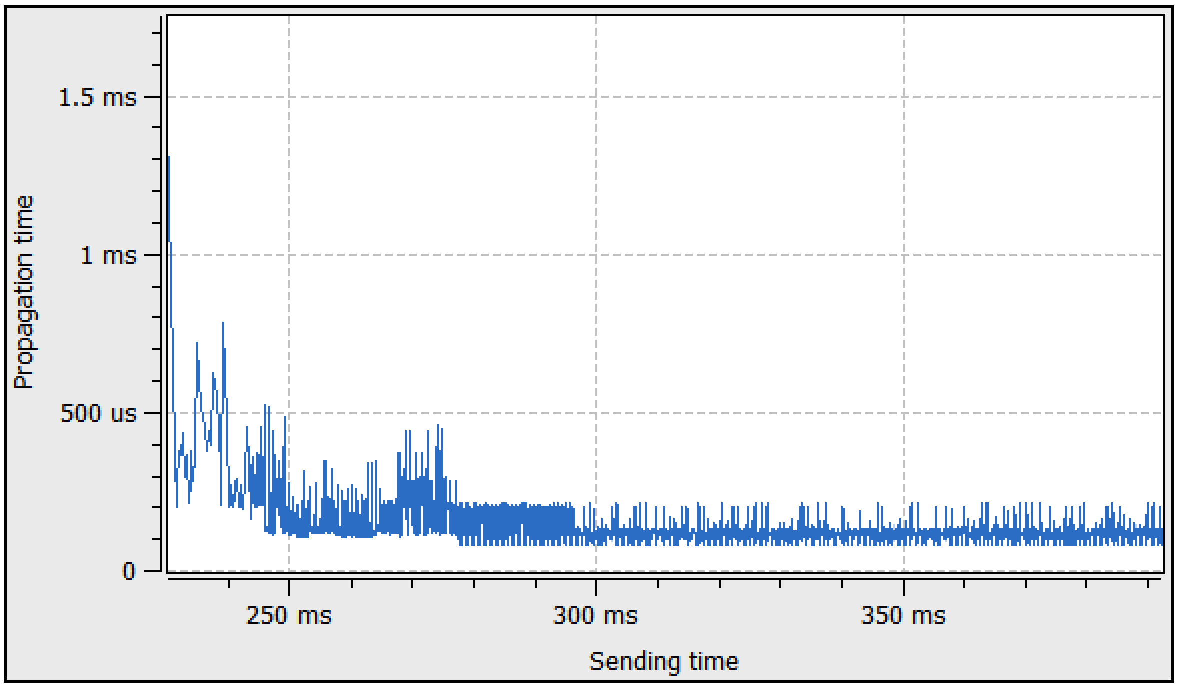


Figure 5 Latency of video frames in larger scale

Message Latency			Count
75.96 us - 1.311 ms			99.7% 14543
1.311 ms - 2.546 ms			0.3% 42
2.546 ms - 3.781 ms			0.0% 3
5.016 ms - 6.251 ms			0.0% 6
Min	Avg	Max	Total
75.96 us	138.383 us	6.251 ms	14594

Figure 6 Statistic for latency of video frames

Message Latency			Count
6.05 us - 821.982 us			81.7% 98
821.982 us - 1.638 ms			5.8% 7
1.638 ms - 2.454 ms			3.3% 4
3.27 ms - 4.086 ms			9.2% 11
Min	Avg	Max	Total
6.05 us	581.748 us	4.086 ms	120

Message Latency			Count
6.06 us - 230.792 us			61.7% 74
230.792 us - 455.524 us			3.3% 4
455.524 us - 680.256 us			21.7% 26
680.256 us - 904.988 us			10.0% 12
904.988 us - 1.13 ms			3.3% 4
Min	Avg	Max	Total
6.06 us	263.34 us	1.13 ms	120

Figure 7 Latency of packets for CBOCs -> CCM (on the left) and CCM -> CBOCs (on the right)

The statistics for packets from CBOCs and the CCM (fig. 7) shows that these packets are transferred for no longer than 10 ms (4 ms in the worst case), as required. We can also see that packets from CBOCs have much worse average latency. This is basically because CCM receives not only short packets from CBOCs but also 1 Kbyte packets from other terminals. And in spite of the priorities, packets from CBOCs can be delayed (for example, because some packet has occupied the CRRV -> CCM channel before anyone else). So one of the weak places in the network is the CRRV -> CCM channel; it can be improved by adding one more link along with a group-adaptive routing, [2].

We have also found that every 200 ms workload of some channels goes down for a while, and then grows back. It can be explained because some packets simply stop and wait in the CRRV until a lot of generated traffic passes. For example, the channel between the RRV2 and the CRRV is shown on fig. 8. But there is no such thing for CRRV's output channels because traffic goes out of this router continuously (fig. 9).

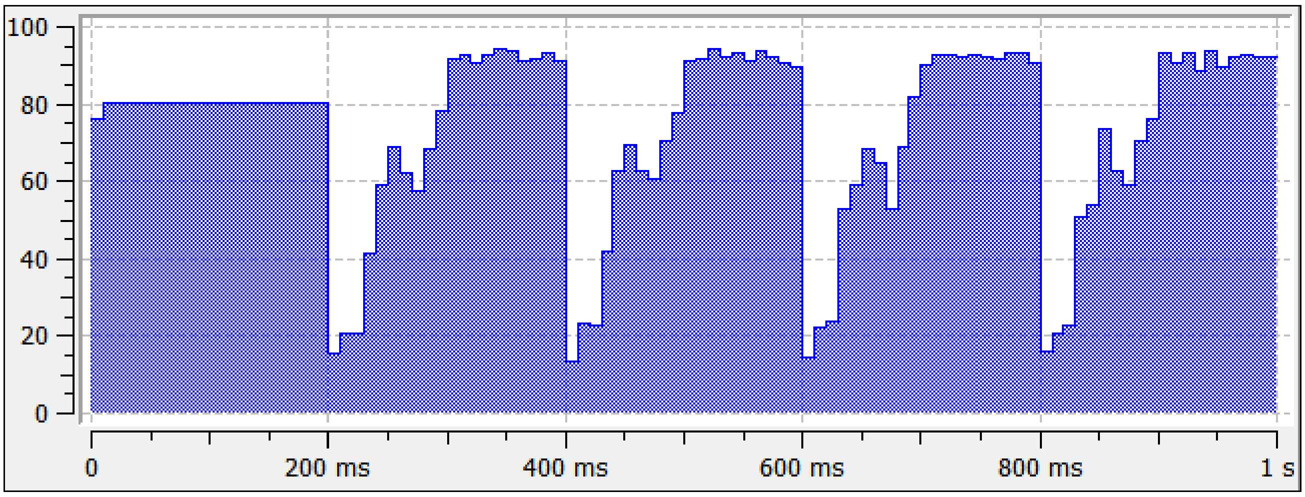


Figure 8 Workload of the channel RRV2 -> CRRV, in percent

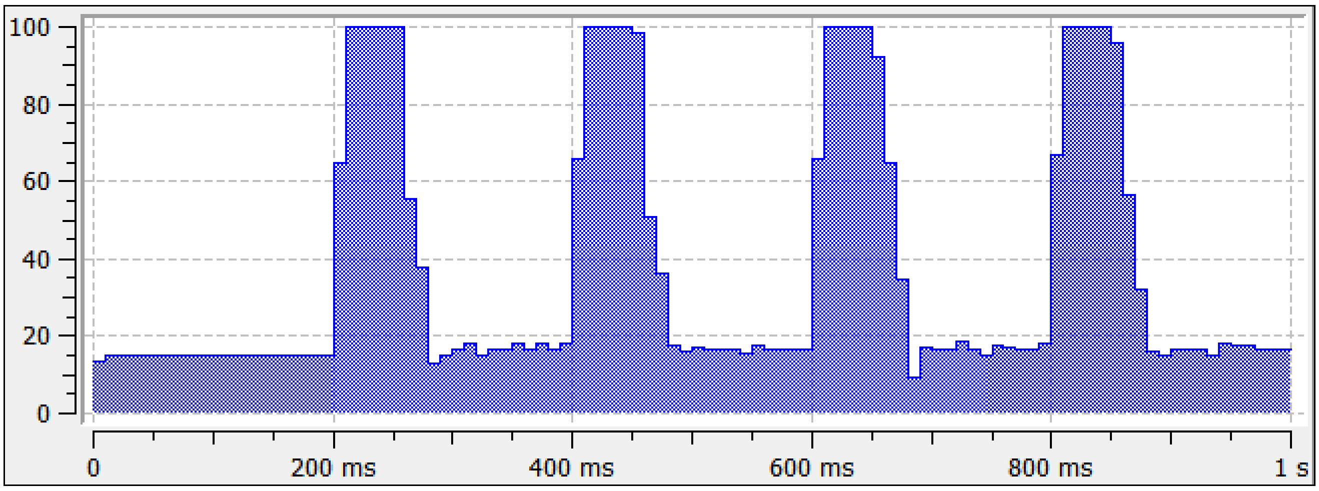


Figure 9 Workload of the channel CRRV -> RRV5, in percent

IV. Conclusion

The DCNSimulator is intended for system-level simulation of different networks. Here we have used it to simulate a fragment of the spacecraft network, and verified that it works as required. Its utilization in practical spacecraft networks design and investigation showed its efficiency and usefulness.

References

[1] A. Eganyan, L. Koblyakova, E. Suvorova. "SpaceWire network simulator". SpaceWire-2010. Proceedings of the 3rd International SpaceWire conference, St.Petersburg, 2010, pp. 403-406.

[2] L. Koblyakova, Yu. Sheynin, D. Raszhivin. Real-Time Services in Networked Embedded Systems. 7th Conference of Open Innovations Framework Program FRUCT, SPb, Russia, 2010. ISBN 978-5-8088-0529-3, pp.64-67.

Implementation and use of SpaceWire in the EPD instrument for Solar Orbiter

SpaceWire missions and applications, Poster Paper

Ronald Castillo, Javier Almena, Alberto Carrasco, Aaron Montalvo, Oscar Gutiérrez, Manuel Prieto, Sebastián Sánchez
Space Research Group, Computer Engineering Department, University of Alcala
Ctra. Madrid – Barcelona, Km. 33,600. Alcala de Henares (28805), Spain
{rcastillo, jalmena, acarrasco, amontalvo, ogm, mpm, chan}@srg.aut.uah.es

***Abstract*—In this paper the cold redundant SpaceWire link used to communicate the Energetic Particle Detector onboard Solar Orbiter with the spacecraft is described. The main objective of the Solar Orbiter mission is to address the central question of heliophysics: How does the Sun create and control the heliosphere? The spacecraft is equipped with a comprehensive suite of ten instruments including in-situ and remote instruments. The Energetic Particle Detector or EPD is part of the in-situ payload. EPD is responsible for studying suprathermal and energetic particles with different energy ranges, covering from 2keV to 200 MeV/n. This instrument is composed of four sensors, developed by universities and research centers across Europe and USA. All sensors are connected to the Instrument Control Unit or ICU, which acts as an interface among them and the spacecraft. The ICU is made up of a Common Data Processing Unit and a Low Voltage Power Supply. The CDPU processes and temporarily stores data captured by the EPD sensors, and also processes telecommands sent by the spacecraft. The LVPS provides a switchable power source for the CDPU and the EPD sensors. Communications between the satellite's computer and EPD instrument will be carried out using the CCSDS protocol over SpaceWire. This work explains the implementation of the EPD instrument, including the use of SpaceWire and other higher-level protocols in order to ensure correct communications with the satellite.**

***Index Terms*—ESA, Solar Orbiter, Energetic Particle Detector.**

I. Introduction

In order to better understand the impact of the Sun's behavior in the inner Solar System, the European Space Agency has devised the Solar Orbiter mission. Solar Orbiter will study the inner Solar System, studying not only the Sun but also the particles surrounding the spacecraft during the mission, performing a series of remote and local observations while facing extreme environmental conditions. This study will be carried out while the satellite develops an elliptical orbit around the Sun [1]. For complying with the mission objectives, the satellite contains ten instruments, which will perform the aforementioned observations and the study of the interstellar conditions and the Sun activities. One of these instruments is the Energetic Particle Detector (EPD), which will study the characteristics of the suprathermal and energetic particles surrounding the spacecraft. This instrument is composed of four sensors that will study particles with different ranges of energy. Control of the instrument is performed using an Instrument Control Unit (ICU). Some of the tasks of this unit are to receive and process data fed by the sensors and to process telemetry and telecommands. Data communications between the ICU and the satellite's computer is carried out using a SpaceWire link. This link is also used for synchronizing the data transfer between the EPD sensors and the ICU.

II. Overview of EPD

As mentioned earlier, EPD is composed of four sensors that measure the energy of particles located nearby the satellite. In detail, the range of energy levels that each EPD sensor is able to measure is as follows:

- SupraThermal Electrons, Ions, and Neutrals (STEIN): Approximately 3 to 100 keV for suprathermal particles, approx. up to 40 keV for ions and approx. up to 10 keV for neutrals [2].
- SupraThermal Ion Spectrograph (SIS): Approx. 0.008 to 10 MeV/nucleon for He to Fe [3].
- Electron and Proton Telescope (EPT): 20 to 400 keV for electrons, 60 to 7000 keV for protons [4].
- HET: 300 keV to 20 MeV for electrons, 10 to 100 MeV for protons and approx. 20 to 200 MeV/nucleon for heavy ions (species dependant) [5].

Data generated by these sensors are fed to a set of two cold-redundant Instrument Control Units (ICU) using Universal Asynchronous Receiver-Transmitter (UART) links where Low-Voltage Differential Signaling (LVDS) is used as the standard for the physical level signaling. LVDS was chosen because of its improved noise performance over standard (non-differential) connections and due to its low power consumption.

Figure 1 shows how the components of the EPD instrument are connected to each other and to the spacecraft. Each EPD sensor is connected to both ICU units (nominal and redundant), which are enclosed in the ICU Box. Data connections from the sensors go to the Common Data Processing Unit (CDPU) and power connections to the Low Voltage Power Supply (LVPS). The CDPU not only receives power from the LVPS, it also controls the power supply of the sensors and reads the LVPS status. Both ICUs are connected to the spacecraft using two SpaceWire links, one for each unit.

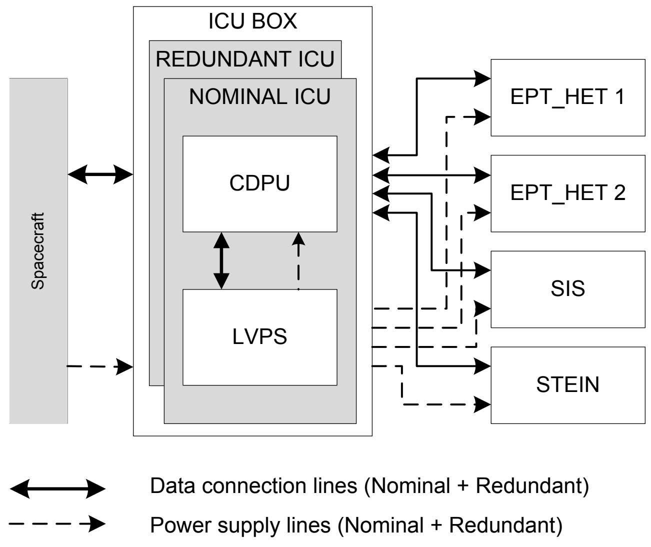

ICU BOX
REDUNDANT ICU
NOMINAL ICU
CDPU
LVPS
Spacecraft
EPT_HET 1
EPT_HET 2
SIS
STEIN
Data connection lines (Nominal + Redundant)
Power supply lines (Nominal + Redundant)

Fig. 1. Connection of the components of the EPD instrument inside Solar Orbiter.

III. ICU HARDWARE

Each ICU unit is composed of two parts: the CDPU and the LVPS. Two ICU units are present in the EPD instrument operating as cold-redundant system. The CDPU receives, processes and temporarily stores data generated by the EPD sensors. In addition, it also receives telecommands from the spacecraft, processes them and sends telemetry data as a reply. Connection between each CDPU and the satellite is carried out using SpaceWire links operating at 10 Mbit/s. More information about the use of SpaceWire in EPD can be found later in this document.

Each CDPU unit is composed of the following components:

- A 32 Kbyte PROM memory, which stores the boot software.
- A 1 Mbyte EEPROM memory that stores the application software and the data required for the correct operations of the EPD sensors.
- A 2.5 Gbit SDRAM memory. The application software is deployed in this memory during the boot process and after that it is executed from here. In addition, the data provided by the EPD sensors are stored in this memory until it is transmitted to the Earth. This memory also contains the information necessary for implementing the Error Detection and Correction (EDAC) functionality.
- A radiation-tolerant FPGA, where all data processing and communications will be performed.
- LVDS drivers and receivers for the UART and SpaceWire interfaces.
- Additional interfacing components.

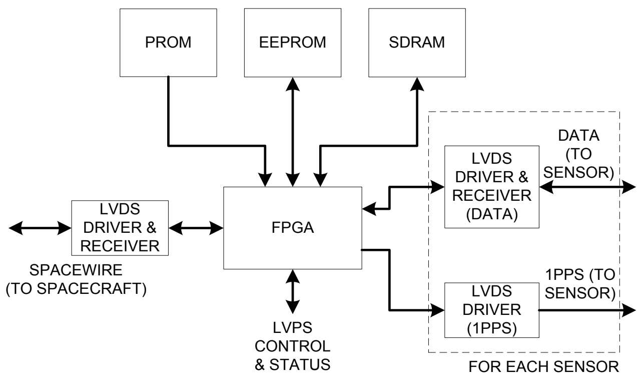


Fig. 2. FPGA Block diagram.

The FPGA handles data exchange between the EPD sensors and the spacecraft. For this reason, it contains the logic of several UART interfaces, a LEON2 processor and a SpaceWire interface, along with additional interconnection logic and memory controllers as shown in Figure 2. Connection of the LEON2 processor and the necessary peripherals is carried out using an Advanced Microcontroller Bus Architecture (AMBA) bus. In detail the FPGA, which operates at 20 MHz, performs the following tasks:

- Reception of data from the EPD sensors.
- Processing and compression of the received data.
- Temporary storage of the resulting data in the SDRAM memory.
- Reception of telecommands from the spacecraft.
- Telecommand processing and replying. This includes transmitting data provided by the sensors to the Earth.
- Control of the LVPS unit. This allows switching the power supply of the EPD sensors.

To relieve the processor from the task of managing data transfer operations to and from the EPD sensors, an IP core has been developed. It provides a configurable amount of UART interfaces and an AMBA master peripheral interface with DMA capabilities. This makes it possible to maximize the resources dedicated to the sensors' data processing.

The LVPS adapts the spacecraft's +28V power supply to be used by the EPD instrument. In addition, the power supplied by the LVPS to the EPD sensors can be switched on or off individually, following a command received from the CDPU. The LVPS also provides protection mechanisms, such as short-circuit and under-voltage protection [6].

IV. EPD SPACEWIRE IMPLEMENTATION AND DATA TRANSFER

Communications between the spacecraft and its instruments is carried out using SpaceWire links. Specifically for EPD, two SpaceWire links are provided by the spacecraft, one for each cold-redundant ICU. These SpaceWire links operate at 10 Mbit/s. SpaceWire is not only used for data exchange, Timecode characters are used for synchronizing the data transfer from the EPD sensors to the ICU. In addition, communications among different Solar Orbiter's instruments is also performed over this link.

The SpaceWire codec used in the ICU EPD has been developed by our group [7]. This SpaceWire codec has been tested to work correctly at bitrates up to 300 Mbit/s and provides the basic functionality specified in the SpaceWire standard [8]. A wrapper component was implemented over this SpaceWire codec that provides the interface with the AMBA bus, DMA functionality for direct data exchange with the external memory without CPU intervention and improved data packet and Timecode handling. A block diagram of this component is shown in Figure 3.

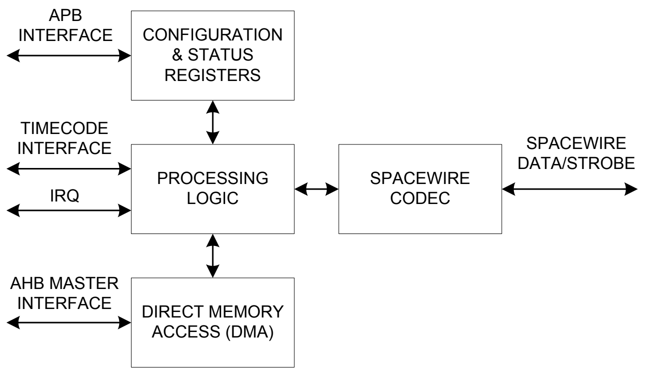


Fig. 3. Block diagram of the SpaceWire core used in the CDPU FPGA.

Data exchange and error reporting in this component is based on transmission and reception descriptors. For data transmission, the software being executed in the LEON2 processor prepares descriptors in external memory, which contain information such as the starting address of the data to be transmitted and its length. The software then configures the IP core so it knows that there are descriptors pending processing and where to find them. For data reception, a similar process is followed, but in this case the descriptors contain the position in memory where each received SpaceWire packet is to be stored and the maximum length allowed.

After the component processes these descriptors, it updates them to report any errors that occurred during the data transfer.

Exchange of telecommands and telemetry through the SpaceWire link is carried out using the CCSDS packet transfer protocol [9]. CCSDS data management is implemented in software.

V. Use of Time-Codes in EPD

SpaceWire Timecode characters sent periodically by the spacecraft play an important role in the communication process between the ICU and the EPD sensors. A mechanism that allows synchronizing the transmission of data from the sensors to the ICU has been implemented.

When the ICU receives a valid Timecode character over the SpaceWire link (one is expected each second), another IP core included in the FPGA is notified. This IP core then generates a 1 Pulse-Per-Second (1PPS) pulse that is sent to each EPD sensor over dedicated LVDS lines. The 1PPS is used to notify the sensors that they are authorized to send data. By defining the maximum amount of data that a sensor can send each time they receive a 1PPS pulse, we can determine the maximum amount of data that the ICU has to process in a period of time and, as a consequence, the amount of processing resources necessary in the system.

VI. Inter-instrument Synchronization and Burst Mode

In Solar Orbiter, instruments can share scientific data if necessary. This is possible thanks to the implementation of Packet Utilization Standard (PUS) Services [10] in the Spacecraft's computer and its instruments. In this implementation, the instruments send scientific data to the Spacecraft using Service 3 packets. The satellite's computer then processes the information received from all the instruments and sends them Service 20 packets with the resulting data. For EPD, this information may include a request to activate Burst Mode for one or more EPD sensors.

Before explaining the functionality behind the Burst Mode, it is important to know that the EPD sensors capture scientific data with a fixed cadence (each second, samples are taken by these sensors and sent to the CDPU). For sending this data to the Earth, the CDPU composes telemetry packets by compiling ten samples taken by a sensor for each packet, which is then sent to the spacecraft. It must be noted that the compiling process, that basically is a data integration operation, reduces the amount of data that must be sent to the Earth. This mode of operation is called Nominal Mode.

In addition to the telemetry data sent in Nominal Mode, during some specific periods of time, the CDPU may send high cadence scientific data from one or more EPD sensors. This mode of operation is called Burst Mode, and can be activated independently for each sensor for which this functionality is supported. In this scenario, the CDPU generates telemetry frames with high cadence data from a single sensor, so more information is available about that sample than in Nominal Mode. Figure 4 shows the difference between Nominal Mode of telemetry reporting and Burst Mode.

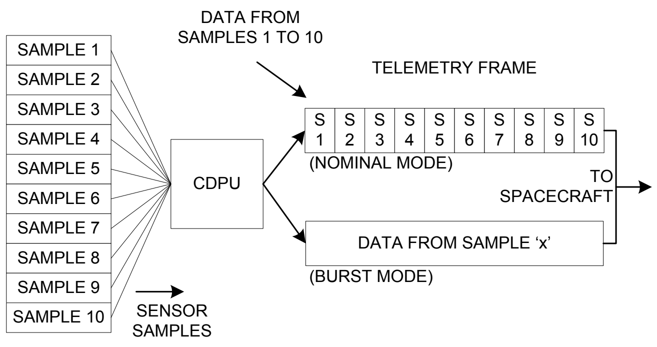


Fig. 4. Telemetry frame generation using Nominal and Burst Modes

Burst Mode can also be activated internally by one of the sensors present in the EPD instrument or remotely as a telecommand sent from the Earth.

VII. ICU Testing Environment

In addition to the development and implementation of the instrument's hardware and software, a testing environment has been set up with the purpose of testing the CDPU's functionality.

A Unit Tester application has been developed for execution in a Personal Computer (PC). This application, combined with embedded software also developed specifically for testing purposes (which is executed by the CDPU), is a powerful tool to assist in testing the CDPU's peripherals during and after the implementation process.

In this test scenario, commands are sent from the test PC to the CDPU under test using a SpaceWire link. The content of these commands follows a protocol also developed specifically for testing the CDPU. This protocol not only allows specifying different tasks to be carried out using the set of peripherals available in the CDPU, but also allows ordering tests where several peripherals are interconnected. For example, a possible test could be to read data from the SPI interface (which is used for reading the LVPS status) and send it using one of the UART interfaces, while another possibility would be to tell the CDPU that, for each SpaceWire packet it receives, it should send an amount of reply packets specified in the received packet. More complex tests can be implemented if so desired.

The PC application loads tests to be carried out from XML files. These files are human-readable, which eases the process of implementing tests or reading what an existing test is programmed to do. This application has been developed using the LabView environment [11]. To provide the PC used as a testing station with SpaceWire connectivity, a SpaceWire-USB Brick [12] from Star Dundee is used.

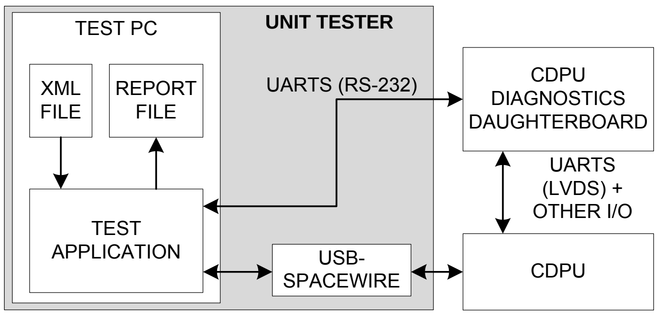


Fig. 5. CDPU testing environment.

Figure 5 shows the complete testing environment. The Unit Tester is connected to the CDPU through the SpaceWire interface as well as indirectly through its UART interfaces. An additional diagnostics board is used, which allows reading the values of the CDPU's output signals or setting value to its input signals. This board also converts the LVDS signals used for the UART interfaces in the CDPU to the RS-232 standard for connection to the test PC.

VIII. Conclusions

Solar Orbiter will let us acquire a greater knowledge of the Sun's behavior and the environmental conditions of the inner Solar System. This knowledge is very important for understanding the consequences of the Sun's environmental conditions in Earth, and for better preparing future space missions for the hazards of the interstellar medium. SpaceWire plays a key role in this mission, providing a robust, high-speed link for data transmission that also allows for synchronization of the exchange of information between the CDPU and the EPD sensors.

Acknowledgments

This work has been supported by the Spanish government, MINECO grant AYA2012-39810-C02-02.

References

[1] ESA Science & Technology. Solar Orbiter. http://sci.esa.int/solarorbiter. Retrieved on March 7th, 2013.

[2] Solar Orbiter's Energetic Particle Detector (EPD) – STEIN. http://www.ieap.uni-kiel.de/et/solar-orbiter/STEIN.php. Retrieved on March 7th, 2013.

[3] Solar Orbiter's Energetic Particle Detector (EPD) – SIS. http://www.ieap.uni-kiel.de/et/solar-orbiter/SIS.php. Retrieved on March 7th, 2013.

[4] Solar Orbiter's Energetic Particle Detector (EPD) – EPT. http://www.ieap.uni-kiel.de/et/solar-orbiter/EPT.php. Retrieved on March 7th, 2013.

[5] Solar Orbiter's Energetic Particle Detector (EPD) – HET. http://www.ieap.uni-kiel.de/et/solar-orbiter/HET.php. Retrieved on March 7th, 2013.

[6] Solar Orbiter's Energetic Particle Detector (EPD) – ICU. http://www.ieap.uni-kiel.de/et/solar-orbiter/ICU.php. Retrieved on March 7th, 2013.

[7] R. Castillo, J. A. Martín, J.Almena, M. Prieto, D. Guzmán and S.Sánchez, "Validation and testing of an IP codec for high bandwidth SpaceWire link". SpaceWire-2010. Proceedings of the 3rd International SpaceWire Conference, pp. 179-183.

[8] European Cooperation for Space Standardization. ECSS-E-ST-50-12C. Space engineering. SpaceWire – Links, nodes, routers and networks. 2008.

[9] European Cooperation for Space Standardization. ECSS-E-ST-50-53C. Space engineering. SpaceWire - CCSDS packet transfer protocol. 2010.

[10] European Cooperation for Space Standardization. ECSS-E-70-41A Ground systems and operations - Telemetry and telecommand packet utilization. 2003.

[11] National Instruments. LabVIEW System Design Software. http://www.ni.com/labview/. Retrieved on March 13th, 2013.

[12] Star-Dundee. SpaceWire-USB Brick. http://www.star-dundee.com/products/spacewire-usb-brick. Retrieved on March 11th, 2013.

Network Management Algorithm For High Speed Onboard Systems

SpaceWire networks and protocols, Poster Paper

Koblyakova Ludmila
SUAI
Saint-Petersburg, Russia
luda_o@rambler.ru

Oleynikova Stanislava
SUAI
Saint-Petersburg, Russia
o.stanislava@gmail.com

Khramenkova Ksenia
SUAI
Saint-Petersburg, Russia
ksu.khramenkova@gmail.com

***Abstract*— High speed evolution of onboard technologies leads to increasing of requirements to the algorithms, which provide administration, configuration and monitoring of network. The significant rise of number of devices in networks is caused by developing new algorithms for automatic network configuration without human intervention. During interworking huge sets of devices it is necessary not only to initialize it. Log-tracing and reconfigure every node and whole network are also urgent task. Therefore, the developing of algorithm for automatic network configuration and monitoring is high priority task.**

The paper gives the overview of configuration, administration and monitoring algorithms for modern onboard data transfer standards: InfiniBand, Fibre Channel, AFDX, MIL-STD-1553, SOIS and SPA. Definitions of terms of configuration, administration and monitoring are in the first and second part. The review of data transfer standards is in the third part. We consider only algorithms of administration, configuration and monitoring. Advantages and disadvantages are particularly explored. Also we analyze how it can be used with SpaceWire technology.

***Index Terms*—Administration, configuration, monitoring, Plug and Play, InfiniBand, Fibre Channel, AFDX, MIL-STD-1553, SOIS, SPA, SpaceWire.**

I. Introduction

The onboard network evolution and growing the number of devices in networks leads to creating new algorithms. Also need to take into account the wide variety of hardware, which may have a different interfaces, performance, features and capabilities.

The algorithms respondent to the administration, monitoring and reconfiguration of the onboard network should not only initialize all the devices, but also track the status of each device during the all network working time, providing to the operator a log about whole network and each device.

Algorithms can be completely different: centralized, decentralized, with input data and without, taking into account the error statistics, the presence of redundant channels and / or devices.

But they are used a number of requirements and restrictions, such as:

- Configure and network scan should not cause deadlocks;
- The algorithm should work correctly on any network;
- The algorithm should not significantly affect the ability of computer networks.

According to the structure the algorithms can be divided into several stages: administration, monitoring and reconfiguration onboard computing network.

Under the term of administration we understand that, if there is information about network devices send command for direct recording settings in them. If there is no input data, then to the administration stage the network discovering is added.

After the first stage the tracking of the network state and each device begins. During the monitoring the output log file is created, reflecting the status of the devices, links, and error statistics occurring in the network. The monitoring includes testing devices and making conclusion on the base of the receiving results about their efficiency.

If errors are found, it is necessary to take the decision about trying to restore settings or to transfer the management to the redundant device, if first device cannot be restored or the device is out of order. About the choice and actions are reported to the network operator through log files. All this actions are called reconfiguration of the network. This function will be called in the case of detection or disable devices, needs to input or output device in standby mode.

II. Monitoring and its Services

The monitoring function represents tracking the network state. It is executed continuously after network initialization. The monitoring checks intactness of network devices, backtraces the appearance new devices in network, guides statistical parameters which can be interested for network administrator, forms error report.

Also one of the monitoring functions is tests, for example, device interrogating for detection fails or new devices. In case of finding out the fail, the message is sent to network administrator and forms an output report file.

For example, during test the terminal node, the connected ports (connection, speed) and possibility device to send packets are checked. During test the router, the connections and possibility to send packets are checked as well. In addition the router table, adaptive group registers are checked.

In case of some fails the monitoring tries to dispose them by using configuration and administration tools and informs the network administrator. This stage is named network reconfiguration. Its initiator is signal from monitoring stage about appearance or disappearance device or several devices. This leads to creating new parameters and adjustments.

III. Overview of standards

A. MIL-STD-1553

This standard was developed for military purposes, and describes 1-megabit bus with time-division. Its special feature is the dual redundant data bus built on a "command-response" scheme. The second bus is backup bus, it is using when the primary fails.

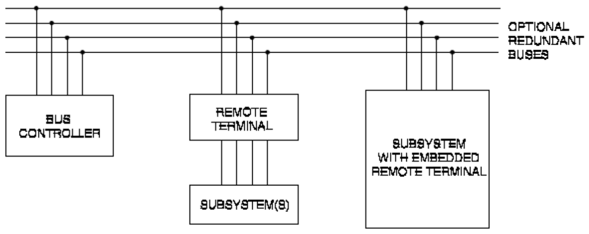


Fig. 1 Scheme of MIL-STD-1553 bus

All the actions that occur on the bus are controlled and executed by the main bus controller. Remote terminals are connected to the bus and asked for the commands the controller. Remote terminals can be up to 31. Also in the network can be a monitor bus.

The main function of the bus controller is controlling the flow of data for all transactions on the bus. The exchange of the data is taken place in the "command-response" mode; the device fully monitors the data transfer. It also detects and corrects errors that occurs on the bus and keeps log of errors. The controller records the changes in the network, and performs the appropriate actions, for example, connects the redundant devices. The bus can support several controllers, but at one time just one can work.

The remote terminal is the interface intended for connection of the bus and a subnet. The subnet is connected through such remote terminal and can contain up to 31 subaddresses. Terminal cannot begin transmitting data until doesn't asked be the bus controller. The remote terminal must be properly handles the protocol and electrical errors.

The monitor of the bus listens to all messages on it and writes some. It can store data for analysis in real time or after some time. The monitor can store all bus traffic or only a part, including protocol and electrical errors. The monitor is usually used for bus testing [1].

Considering possibility of application of methods of the MIL-STD-1553 standard for the onboard systems constructed on the basis of SpaceWire devices, it is possible to draw the following outputs:

- The standard MIL-STD-1553 controller analyzes and monitors the network. Algorithm is centralized, only one controller at the time can work.
- The bus MIL-STD-1553 monitor is the passive device which realizes traffic and history storage and bus tests. In the on-board network Space Wire will be useful to apply such device to gather information about current changes in the network (connecting and disconnecting devices, errors).
- Network built according to MIL-STD-1553, has a well-defined topology and the restriction on the number of devices. This condition is not reasonably to the Space Wire network.
- Initially the network built according to MIL-STD-1553 already set up and run. The main task is maintaining its performance. This condition is not always reasonably for Space Wire network.

B. InfiniBand

For the administration and monitoring of the network, the all network structure is divided into subnets.

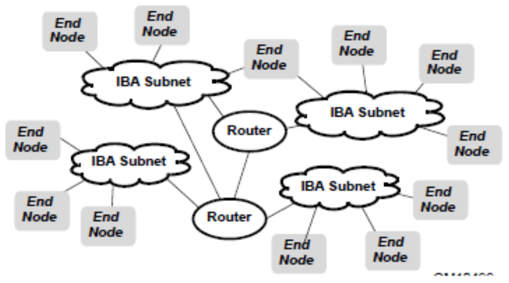


Fig. 2 IB Network and Subnetwork

For the process of administration, configuration and monitoring, there are managers and agents that support a range of services.

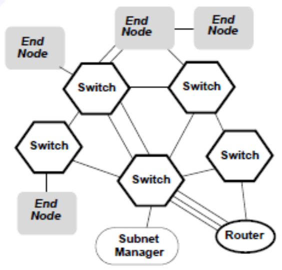


Fig. 3 IB Subnet Components

A Subnet Manager is an entity attached to a subnet that is responsible for configuring and managing switches, routers, and channel adapters. And each node provides a Subnet Management Agent that the Subnet Manager access through an interface called the Subnet Management Interface.

The manager can be in one of the following statuses: study, waiting, master or non-active.

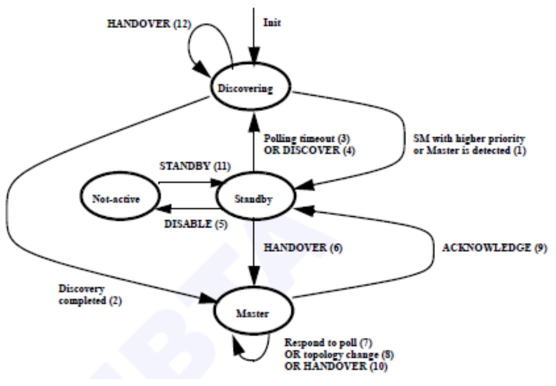


Fig. 4 IB Manager Stages

The first step is determining the master manager of subnet. If manager find out a node with a bigger ID then manager goes into waiting mode. In this mode it doesn't configure the network. It only checks once in some time the operability of the master. If for some reason the master fails or doesn't respond, then the manager again enters to the study state. He also enters this mode when receive a packet "study". If he gets the package "disable", then he needs to go into the "non-active" state. In this case, it does not perform any action. He can go into a waiting state when receiving an appropriate packet.

The master manager makes study of topology of a subnet, distribution of local addresses to the devices, compilation of paths to these devices, executes configuring of devices of a subnet and all subnet in case of appearance of new devices or switch-off the operating.

Subnetwork agents are contained in everyone subnet devices, in channel adapters, routers and switches. They provide the opportunity to interact between the device and manager. By means of the agent the access to the configurable parameters is made.

Control of IB provides a configuration and information collection about channel adapters, switches and routers, determination of topology and a subnet configuration.

InfiniBand management defines a common management infrastructure for

- Subnet administration - provides nodes with information gathered by the subnet manager and provides a registrar for nodes to register general services they provide.
- Communication establishment and connection management between end nodes.
- Mechanisms to discover and manage I/O devices "behind" channel adapters.
- Configuration management - an authority for assigning I/O resources to hosts.
- Performance management - monitors and reports well-defined performance counters.
- Baseboard management - provides for power & chassis management.
- SNMP Tunneling (SNMP) - provides method for sending and receiving information between management agents and management applications. This includes Simple Network Management Protocol (SNMP), Desktop Management Interface (DMI), and Common Information Model (CIM) [2].

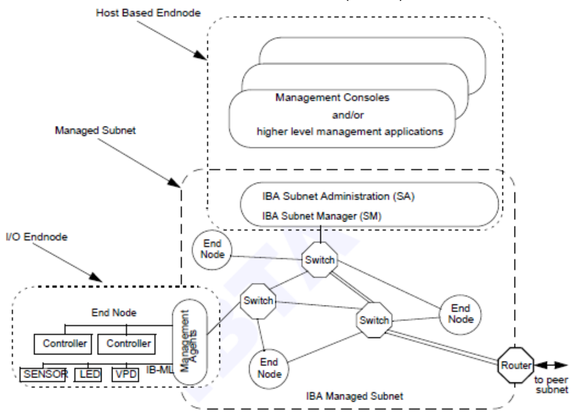


Fig. 5 IB Management

It is possible to draw the following outputs about the InfiniBand standard:

- The InfiniBand standard is centralized for subnets and decentralized for all system.
- No restrictions on the topology, devices can be duplicated if it is necessary.
- There is no need of continuous or periodic survey of devices, the agent being on the device reports about it.
- Each subnet must support at least two managers, and each network device must support the work of the agents. Directly from the device manager does not work.

At the InfiniBand standard is a row of advantages which can be used to implementation of algorithms of Space Wire. The using the agents in the network Space Wire are difficult because often terminal nodes are the sensors which aren't supporting installation on them the additional software. However, the switches may contain such software, adding to its function the monitoring of terminal nodes. The simple circuit of network monitoring can be created by the adding of manager to the network topology, which can process information from agents.

C. Avionics Full Duplex Switched Ethernet (AFDX)

AFDX combines concepts taken from asynchronous transfer mode and applies them to a variant of IEEE Std 802.3 (Ethernet). At the physical layer, AFDX consists of a star-topology, full duplexed switched Ethernet.

In order to improve the reliability AFDX provides a redundant network scheme.

Each frame is transmitted in parallel over two redundant networks and afterwards filtered by Redundancy Manager at the receiving End System. This shall reduce the probability of loosing frames and enable further operation even in presence of one faulty network [3].

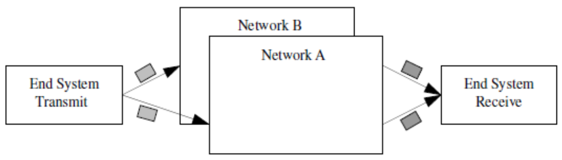


Fig. 6 AFDX Network

Management of an AFDX network is handled via a network management function that communicates with each AFDX network component (equipment, subscriber, and switch) to monitor the health and status of the network.

Network health is monitored via simple network management protocol (SNMP) agents running on each subscriber (line-replaceable unit (LRU)/partition) and end system (including the switch end system). Health status and errors are logged to the local MIBs, with status messages sent as requested by the network management function [4].

The main concept of SNMP protocol is that all necessary information for manage device is stored on that device in Management Information Base (MIB).

MIB is the set of variables characterized the state of management object. These variables can reflect such parameters as number of packets, processed by device; state of its interfaces; time period of function this device, etc.

For process the inquiries from control station received as SNMP packets the special module, named Management Agent, is existed. Agent receives SNMP packets and performs corresponding actions, for example, set value to parameter, update information in MIB.

The Control Station can be workstation of network administrator, if there run some management module which supporting the SNMP protocol.

The feature of this protocol is its simplicity. It includes just several commands.

- The command GetNext-Request is used by manager to get value of the next object (without its name) during the several reviews the table of the objects.
- The command Get-Response is used by SNMP agent for transmit to manager the answer on command Get-Request or GetNext-Request.
- The command Set is used by manager to change the value of some object. By using this command the device management is occurred.
- The Trap command is used by agent to send the massage to manager about raising the special situation.
- The SNMPv.2 adds to this set command GetBulk, which allows manager get several values of variables in one request [5].

This standard shows the high fault-tolerance by duplicate the flow of packets, but it demands the big hardware resources, because all devices between two end nodes are duplicated. Such approach can be used for SpaceWire networks. However the duplication all network can involve difficulties, because for onboard network is important to minimize the weight and power consumption, thus only very important parts of network can be duplicated.

In network must exist and permanently function the Redundancy Manager.

This standard doesn't discuss where the network manager realized configuration and monitoring should be hosted.

The SNMP protocol consists the minimum number of commands, offers the full access to variables of MIB different network devices and monitoring functions.

The concept of storage information about device state on that device can be successfully applied to SpaceWire networks. In this case devices should hold self-testing software and the network manager produces interrogation its parameters.

D. Fibre Channel

Fibre Channel is the set of protocols for high-speed data transfer. Fibre Channel is complex protocol consisted of 5 layers. On layer FC-3 the Management Services is placed. This is set of tools for access for management application to Fibre Channel network, its inner topology and configuration data. The management applications placed on network devices can, for example, indicate which ports can interact with each other. Other services allow management applications discover the behavior of interactions in Fibre Channel network.

The standard Fibre Channel supports 3 types of topology defined the principles of interaction between devices:

- Point-to-Point topology;
- Arbitrated loop topology;
- Switched-fabric topology.

Physically, the Fibre Channel is an interconnection of multiple communication points, called N_Ports, interconnected either by a switching network, called a Fabric, or by a point-to-point link. A Fibre Channel "node" consists of one or more N_Ports. A Fabric may consist of multiple Interconnect Elements, some of which are switches. An N_Port connects to the Fabric via a port on a switch called an F_Port. When multiple FC nodes are connected to a single port on a switch via an "Arbitrated Loop" topology, the switch port is called an FL_Port, and the nodes' ports are called NL_Ports. The term Nx_Port refers to either an N_Port or an NL_port. The term Fx_Port refers to either an F_Port or an FL_port. A switch port, which is interconnected to another switch port via an Inter Element Link (IEL), is called an E_Port. A B_Port connects a bridge device with an E_Port on a switch; a B_Port provides a subset of E_Port functionality.

Many Fibre Channel components, including the fabric, each node, and most ports, have globally-unique names. These globally-unique names are typically formatted as World Wide Names (WWNs).

The configuration is realized also with SNMP protocol which is described earlier in AFDX section.

MIB Fibre Channel consists of 11 groups:

- Instance Basic Group contains basic information about a Fibre Channel managed instance, including its name and description, the Fibre Channel function(s) it performs, and optional pointers to hardware and/or software components;

- Switch Basic Group contains basic information about a Fibre Channel switch, including its domain-id and whether it is the principal switch of its fabric;
- Port Basic Group contains basic information about a Fibre Channel port, including its port name, the name of the node (if any) of which it is a part, the type of port, the classes of service it supports, its transmitter and connector types, and the higher level protocols it supports;
- Port Stats Group contains traffic statistics, which are not specific to any particular class of service, for Fibre Channel ports;
- Port Class23 Stats Group contains traffic statistics that are specific to Class 2 or Class 3 traffic on Fibre Channel ports, including class-specific frame and octet counters and counters of busy and reject frames;
- PortLc Stats Group defines low-capacity (Counter32-based) equivalents for the Counter64-based statistics in the Port Class23 Stats Group;
- Port ClassF Stats Group contains traffic statistics that are specific to Class F traffic on the E_Ports of a Fibre Channel switch;
- Port Errors Group contains counters of various error conditions that can occur on Fibre Channel ports;
- Switch Port Group contains information about ports on a Fibre Channel switch. For an Fx_Port, it includes the port's timeout values, its hold-time, and its capabilities in terms of maximum and minimum buffer-to-buffer credit allocations, maximum and minimum data field sizes, and support for class 2 and class 3 sequenced delivery. For an E_Port or B_Port, it includes the buffer-to-buffer credit allocation and data field size;
- Switch Login Group contains information, known to a Fibre Channel switch,about its attached/logged-in Nx_Ports and the service parameters that have been agreed with them;
- Link Basic Group contains information known to a local Fibre Channel management instance, and concerning Fibre Channel links including those which terminate locally [6].

The Fibre Channel standard unlike other standards allows organize high-performance network without redundancy.

This standard is use the SNMP protocol for configuration and monitoring network devices. The 11 MIB groups are existed which provide all information about network and each device. The management applications are placed on devices. The using of management applications worked on devices for SpaceWire networks is not always possible, because some terminal nodes cannot support the addition software.

E. Space Plug-and-Play Avionics (SPA)

The Air Force Research Laboratory is developing a system for rapidly building spacecraft based on adapting "plug-and-play" (PnP) approaches for use in space. This space plug-and-play avionics (SPA) system is based on an interface-driven set of standards intended to promote the rapid development of spacecraft busses (platforms) and payloads. As such, SPA is an open systems framework, combining commercial standards with carefully chosen hardware and software extensions necessary for modern real-time embedded systems (e.g. fault tolerance, higher power delivery, self-description).

Space plug-and-play avionics (SPA) is defined as an interface-driven standard (or set of standards) intended to promote the rapid development of spacecraft busses (platforms) and payloads. The SPA standard comprises an open systems framework, which combines core commercial standards (such as USB) with carefully chosen hardware and software extensions necessary for modern real-time embedded systems.

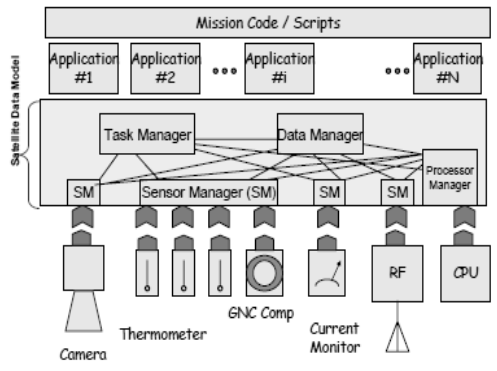


Fig. 7 Vertically-layer software engineering model for PnP

One abstraction of software engineering for PnP follows a vertically-layered, reminiscent of the well-known seven-layer open system interconnects (OSI). At the bottom of this stack are the PnP components themselves. The component layer connects into a "middleware" layer referred to as the satellite data model (SDM). Above this middleware is the application layer. Applications access the PnP object-services through API calls to the SDM, which enforces an insular discipline in systems development. It is not, for example, necessary to write code to control specific thermometers, which might require modification when different thermometers are chosen. Rather, this layered approach encourages device independence in application design, which is one of the principles that permit more rapid integration of components. It is possible to define a final mission layer, potentially as a script-driven interface to the application set.

The key innovation in the PnP software architecture is the SDM. The goal of rapid satellite design, integration, and test requires that established, but time consuming, concepts be rethought and revamped. The SDM does not focus on the electrical transport mechanism, so in principle any number of SPA-x interfaces could be devised. Rather, the SDM is based on the transport of data.

Ontology plays an important role in SDM. For the various aforementioned processes to understand each other, they must speak the same language. To do this, SDM requires a public Common Data Dictionary (CDD) whose contents are created by the community of process developers and managed as a public resource. The CDD concept is key to a data-oriented model, and it enables disparate teams to develop processes in different places and times that are able to understand what data each produces or requires. It permits a community understanding for the development of the device xTEDS, as well as the applications that exploit them in the various SPA components.

The SDM defines a series of interacting “function managers”:

- Processor Manager – resident on each processor and is responsible for keeping that processor busy;
- Data Manager – keeps track of all data available at any given time and supports data queries;
- Task Manager – keeps track of active and pending tasks;
- Sensor Manager – provides the PnP interface to the processing network; and
- Network Manager – explores the network and maintains routing tables.

The managers are logically a single function even though they can have a multi-instantiated distributed implementation. These “managers” support data access, task management, and network discovery. Data access accumulates descriptions of what data is produced by system processes and how that data can be accessed. Task management keeps track of what processes are executing on what processors and their statuses along with what additional tasks are needed. Network discovery determines what components are connected to the network, their addresses, and associated routing tables.

The processor manager bears special mention. It is a special process resident on each processor (since SDM is intrinsically designed to be distributed onto networks) that handles task acquisition and execution along with providing basic support functions. These functions include messaging between processes, maintaining a real-time clock, and providing a periodic heartbeat to the system (i.e., the task manager). The special “per processor” process continuously monitors activity of the parent processor and periodically checks for the existence of pending tasks that can be executed by the parent. If any are found, the appropriate executables are loaded and run. While no operating system is required per se, the process can be multithreaded, handle interrupts, and utilize an operating system as appropriate based upon the specific processor [7].

The functions of administrating are divided between several managers: processor manager, data manager, task manager, sensor manager and network manager. Each manager is responsible for restricted set of functions, which increase the data processing. Also the stability of single manager increases.

SPA use its own format of the packets, received information is described by xTEDS.

SPA use its own logical addressing which have to be modified to use in other subnetwork.

On SPA managers is placed the main work of service the network, storage data, paths and routing tables for different subnetworks.

SPA has the SPA-S realization special for SpaceWire.

F. Spacecraft Onboard Interface Services (SOIS)

The SOIS standardized services are intended to be applicable to all classes of civil missions, including scientific and commercial spacecraft, and manned and un-manned systems. These standardized services may apply to military missions, although military security requirements have not been considered in their specification.

On any given spacecraft, several types of data subnetworks may be used between specific data systems. The actual type of subnetwork used is determined by the required characteristics of the interaction between those entities. These may typically be categorized as:

- Multidrop Buses providing connection to a central bus master and a number of slaves. Communication is generally asymmetrical and often involves low-level read and write access to slaves. The central control of bus traffic results in a highly stochastic traffic profile well suited to applications requiring bounded communications timing.
- Point-to point serial interfaces used for instrument connection, possibly for bulk data transfer but also combined with instrument control. Again, these interfaces usually operate in a master/slave mode.
- LANs used on larger infrastructures where hosts have generally equal computing power and have a diversity of communication requirements. Communication is on a peer-to-peer basis with a level of variability in delay due to resource queuing.
- Point-to-point sensor and actuator interfaces used for gathering sensor readings or controlling spacecraft equipment.

Onboard applications should not be concerned with the nature of these subnetworks, and so the SOIS concept aims to provide a solution by recommending that applications interact only with a well-defined set of standard onboard data services.

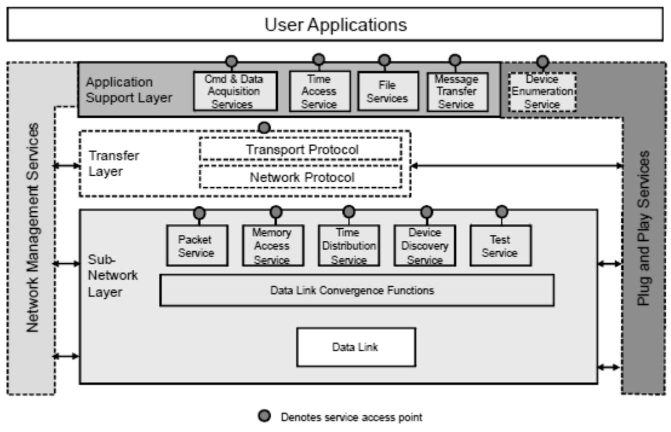


Fig. 8 SOIS Layers

SOIS services exist at three service interfaces:

- An Application Support Layer service interface.
- A Transfer Layer Service interface.
- A Subnetwork Layer service interface.

The Application Support Layer services provide a number of capabilities commonly required onboard a spacecraft, which need not be limited to communications. The Application Support Layer services make use of the Subnetwork Layer services either locally or remotely over a network. The services are defined in terms of protocols, procedures, protocol data units and a Management Information Base (MIB).

The Transfer Layer is assumed to be composed of extant CCSDS recognized protocols and services.

The SOIS Subnetwork provides a set of SOIS-defined services which support upper-layer Application-Support and Transfer-layer entities.

The services identified at the Subnetwork Layer are:

- Memory Access (memory location read/write, includes block move)—providing direct access to device memory.
- Time Distribution—providing transmission and reception of spacecraft time.
- Packet—providing packet delivery over a single subnetwork.
- Device Discovery—providing dynamic device recognition.
- Test Service—providing establishment of subnetwork functionality and availability.

As you can see, services attached to administration, monitoring and reconfiguration place on different layers. Consider it in details.

On Application Support Layer the function Device Enumeration Service (DES) is placed. It supports a dynamic configuration.

On Subnetwork Layer the functions Device Discovery and Test are located.

The SOIS device discovery service provides the capability to detect devices becoming active following a change in the hardware configuration of the spacecraft. This may occur when a cold redundant device is powered on, for example.

The Test Service is intended to be used for checking data system functionality and connectivity of the subnetwork. The service is used to check operation of the subnetwork aspects of the local data system as well as subnetwork connectivity to other data systems [8].

The present standard doesn’t have limitation on topology given an opportunity to create any topology in depend on network function.

In standard is presented such services as detection of connection the new device, tests, dynamic configuration and network discovering.

The adoption of automatic detection of connection new devices, dynamic configuration and test services are not unreasonable to add to SpaceWire network, this give us more opportunities for automation a network functions.

IV. Conclusion

Possibility of automatic network configuration is important factor in selection standard for onboard network.

In depend on goals and resources the different standards and protocols, different algorithms of administration, configuration and monitoring can be used.

During review the different standards, the set of addition for SpaceWire was detected. For example, efficiently to have the additional passive device which gather information about changes in network; the functions of router software responded to tracking the router state and connected to it nodes; the addition several managers greatly simplify administration, configuration and monitoring. For increasing the fault-tolerance it is possible to duplicate important parts in the network.

References

[1] AIM GmbH Avionics Databus Solutions, “MIL_STD_1553 Tutorial v 2.3”, pp. 1–82, November 2010. *(references)*

[2] InfiniBand^{SM} Trade Association, “InfinBand Architecture Specification Volume 1”, Release 1.2 Final Release, October 2004, pp.709-930.

[3] Jan Taubrich and Reinhard von Hanxleden, “Formal Specification and Analysis of AFDX Redundancy Management Algorithms,” 26th International Conference, SAFECOMP 2007, Nuremberg, Germany, September 2007, pp. 1-3.

[4] Ian Land and Jeff Elliott, “Architecting ARINC 664 Part 7 (AFDX) Solutions”, XILINX, XAPP1130(v.1.0.1) May 2009, pp. 1-11.

[5] ERLANG, “Simple Network Management Protocol (SNMP),” Ericsson AB., February 2013, pp. 1-20.

[6] K. McCloghrie, “Fibre Channel Management MIB,” Network Working group, Category Standards Track, May 205, pp. 1–5.

[7] Jim Lyke, Don Fronterhouse, Scott Cannon, Denise Lanza, Wheaton Byers, “Space Plug-and-Play Avionics”, 3rd Responsive Space Conference, April 2005, pp.1-13.

[8] CCSDS, “Spacecraft Onboard Interface Services – Information Report”, CCSDS 850.0-G-1, Green Book, Issue 1.0, 2007, pp. 1-49.

SpaceWire backplane for ground equipment

SpaceWire test and verification, Poster paper

Masaharu Nomachi, Shuhei Ajimura
Osaka University
1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan
nomachi@rcnp.osaka-u.ac.jp

Takayuki Yuasa, Tadayuki Takahashi
Institute of Space and Astronautical Science, JAXA,
3-1-1 Yoshinodai, Sagamihara, Kanagawa 252-5210, Japan

Iwao Fujishiro, Fumio Hodoshima
Shimafuji Electric,
8-1-15 Nishi-kamata, Ohta, Tokyo 144-0051, Japan

Abstract— **SpaceWire backplane for ground equipment has been developed based on micro TCA. Micro TCA is an industrial standard. It has a backplane for LVDS point-to-point data links. SpaceWire is implemented on the data links. Micro TCA system provides not only SpaceWire data links but also reliable power and cooling. It is applied to ground equipment such as test systems and prototype modules.**

***Index Terms*—Backplane, micro TCA.**

I. Introduction

To develop satellite data handling systems, ground equipment has important roll. Prototyping of the modules, emulator modules and test equipment are needed. We have to handle many modules for testing the data handling system. Power cables and SpaceWire cables connecting those modules are often being mess. Compact system to hold those modules is needed to make the test and evaluation easy and reliable.

COTS (commercial off-the-shelf) backplane systems are concerned for ground equipment. COTS backplane systems provide wide variety of products in cost effective way. They should provide LVDS point-to-point data link, good power supply and reliable cooling system. There are several backplane with differential point-to-point connections. Those are used for PCI express, Giga-bit Ethernet, Serial ATA and other serial protocols. Micro TCA is one of such backplane system [1]. We implemented SpaceWire on the differential point-to-point connection of micro TCA backplane. This work is supported by Jaxa and Japan Space Systems.

We have been used Advanced TCA system, which is also serial backplane [2], for particle physic experiments on the ground. SpaceWire-microTCA is designed based on this experience.

II. Micro TCA

Micro TCA is an industrial standard [1]. Micro TCA hosts AMC (advanced mezzanine card) modules. It can handle up to 12 AMC modules. Each module has point-to-point connection to the dedicated controller (MCH: MicroTCA Carrier Hub).

An AMC module has 20 AMC ports. Each port has LVDS transmitter and receiver connections. First 4 AMC ports are called “common option”. They are used for Gigabit-Ethernets and SATA connections. The next 4 AMC ports are called “Fat pipe”. They are usually used for PCI express or other standards. We assigned two SpaceWire port to Fat pipe region. The next 4 AMC ports are called “Extended fat pipe”. They are also used for PCI express or other standards. We assigned two SpaceWire port also to Extended fat pipe region. The last 8 AMC ports are called “Extended option”. They are not supported in COTS backplanes. Custom made backplane is needed to use Extended option region. So, we don’t use Extended option region for our application. Consequently, each AMC module has 4 SpaceWire connections, two in Fat pipe and two in Extended fat pipe. The port connection is shown in the table I.

TABLE I. AMC PORT ASSIGNMENTS

	AMC port	*Signal*		*Primary MCH*	*Redundant MCH*
Common option	0	1000BASE-BX		A	
	1				A
	2	SATA etc.		B	
	3				B
Fat pipe	4	SpaceWire-0	D	D1-D12	
	5		S	E1-E12	
	6	SpaceWire-1	D	F1-F12	
	7		S	G1-G12	
Extended fat pipe	8	SpaceWire-2	D		D1-D12
	9		S		E1-E12
	10	SpaceWire-3	D		F1-F12
	11		S		G1-G12
Extended option	12				
	13				
	14				
	15				
	17				
	18				
	19				
	20				

An AMC port of even port number is assigned to “D” in D-S link. An AMC port of odd port number is assigned to “S” in D-S link. Fat pipe is connected to a primary MCH. Extended fat pipe is connected to a redundant MCH. Each MCH has 24 (2ports × 12 AMC modules) SpaceWire connections. Figure 1 shows schematic view of SpaceWire connection on COTS 12 slot micro-TCA backplane.

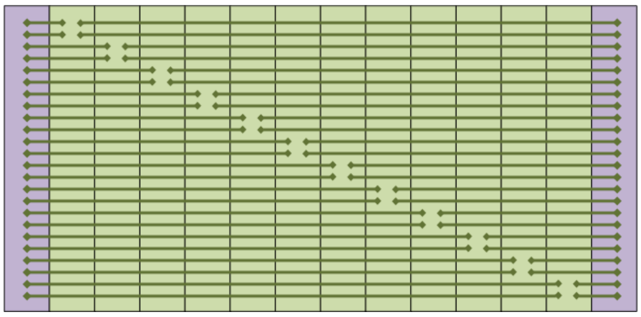


Figure 1. SpaceWire backplane using commercial micro TCA backplane. Two MCHs are placed at the both end.

In order to house less number of modules, 6 AMC slot system was developed. One MCH has 24 SpaceWire ports. One MCH can handle up to 6 AMC modules. However, in micro-TCA standard, Extended fat pipe is not connected. It goes to a redundant MCH. Therefore, we developed custom backplane so as to connect all SpaceWire to one MCH. The port connection is shown in the table II.

TABLE II. AMC PORT ASSIGNMENTS (CUSTOM BACKPLANE)

	AMC port	***Signal***		***MCH***
Fat pipe	4	SpaceWire-0	D	D1-D6
	5		S	E1-E6
	6	SpaceWire-1	D	F1-F6
	7		S	G1-G6
Extended fat pipe	8	SpaceWire-2	D	D16-D7
	9		S	E16-E7
	10	SpaceWire-3	D	F16-F7
	11		S	G16-G7

Figure 2 shows schematic view of SpaceWire connection on the custom 6-slot micro-TCA backplane.

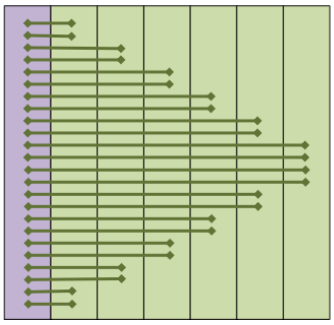


Figure 2. 6-slot SpaceWire backplane using custom micro TCA backplane. All SpaceWire links are connected to one MCH on the left hand side.

The 6-slot custom SpaceWire backplane was developed by UBER [3]. Figure 3 shows the system.

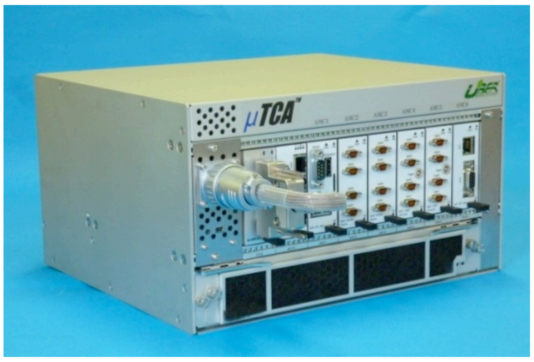


Figure 3. 6-slot SpaceWire backplane system using micro TCA system

III. SPACEWIRE ROUTER

All SpaceWire links are connected to MCH. A SpaceWire router is placed at the MCH. We have developed custom MCH which has SpaceWire router. Figure 4 shows MCH with 28 ports SpaceWire router. The MCH is developed by SHIMAFUJI Electric.

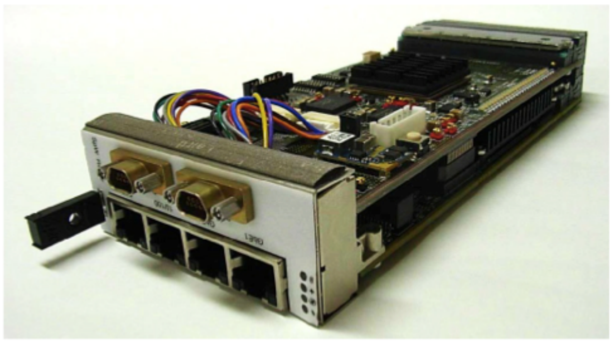


Figure 4. MCH with 28-port SpaceWire router.

28 port router is placed on a FPGA（Xilinx XC6VLX75T-3FFG784）. 24 ports are connected to the backplane. Up to 4 ports are connected to external port. SpaceWire ports work up to 200 Mbps data rate. We also developed MCH with SpaceWire-to-GigabitEther is also available.

MCH controls power of AMC modules and cooling.

IV. AMC MODULE

Several kinds of AMC modules with SpaceWire interface are developed. General-purpose module is developed. The module has 4 SpaceWire ports to backplane and 4 SpaceWire ports to front panel connector. 128 MB SDRAM is attached to the FPGA. Figure 5 shows SpaceWire interface module.

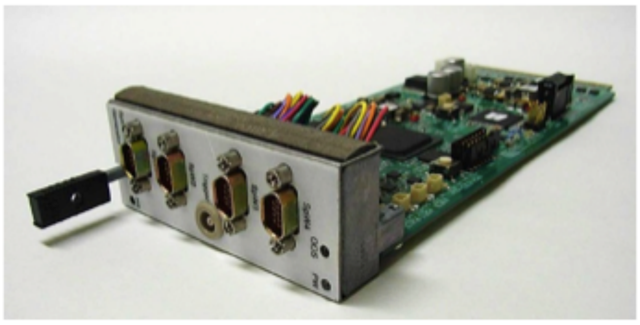


Figure 5. General-purpose SpaceWire Interface.

V. Applications

The micto TCA system will be used for test equipment. Micro-TCA system has high reliability. It is ideal for the test equipment. Traffic Generator system using General-purpose SpaceWire interface is presented by Yuasa et al. [4]

The micro TCA system will be also used for emulator of data handling system. General-purpose interface can emulate the sensor nodes or a data handling system by changing the IP on the FPGA. Reusing existing hardware, development time can be saved.

The micro TCA system will be also used for prototyping of payload electronics. A small form factor of the micro TCA system provides compact development system.

The SpaceWire backplane on micro TCA system will be applied for most of ground applications for SpaceWire.

References

[1] MTCA.0 R1.0 specification, Short Form is available at PCI Industrial Computer Manufacturers Group (PICMG) Web cite, http://www.picmg.org/pdf/MicroTCA_Short_Form_Sept_2006.pdf

[2] Serial Data Link on Advanced TCA Back Plane, M. Nomachi and S. Ajimura, IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. **53**, NO. 5, OCTOBER 2006S

[3] http://www.uber-corp.co.jp/ (in Japanese)

[4] SpaceWire Traffic Generator: a highly-scalable packet generation device, T. Yuasa *et al.*, contribution to this conference.

A Software SoCWire Protocol Handler for NoC Management

Missions and Applications, Poster Paper

Adrian Belger, Björn Fiethe, Frank Bubenhagen, Holger Michel, Harald Michalik

Institute of Computer and Network Engineering (IDA)
TU Braunschweig
Braunschweig, Germany
belger@ida.ing.tu-braunschweig.de

Abstract— **Limited telemetry rates combined with large amounts of information retrieved from the sensor systems of scientific space instruments demand that classical ground processing steps like determination of scientific parameters need to be performed already on-board. FieldProgrammable Gate Arrays (FPGAs) with large logic density provide a highly flexible platform to implement sophisticated data processing. Specifically, radiation tolerant space qualified SRAM-based FPGAs allow to build dynamically and even partially reconfigurable hardware designs, offering significantly improved flexibility for high reliable systems. Our own SpaceWire based System-on-Chip Wire (SoCWire) communication architecture and the RMAP inspired SoCWire Protocol (SoCP) provide an adequate Network on Chip (NoC) communication infrastructure. In this paper a software SoCWire Protocol Handler (SoCPH) implementation for SoCP is presented. This handler contains autonomous mechanisms for determining current network state to provide valid and up to date network state information to the high level software. Additionally, the SoCPH provides autonomous packet header generation. It supports concurrent network interactions from multiple tasks with an automatic response dispatching routine. Synchronous and asynchronous message transfers are supported. These key features significantly lower the otherwise necessary network management overhead for the application software developer and provide easy access to the existing processing nodes.**

Index Terms— **SpaceWire, SoCWire, protocol, network on chip.**

I. Introduction and Heritage

Limited telemetry rate combined with the large amount of scientific raw data retrieved from modern sensor systems demand that classical ground processing steps like scientific parameter extraction and subsequent data evaluation already need to be performed on-board. To cope with these sophisticated on-board processing capabilities, state-of-the-art radiation tolerant space qualified SRAM-based FieldProgrammable Gate Arrays (FPGAs) with large gate count offer an attractive solution to speed up processing by utilizing the parallel structures of FPGAs. Usually, these processing steps need to be revised during a space mission and therefore the scientists request that on-board processing needs to be adaptable to changing mission specific requirements. Fortunately, this type of FPGA provides the capability for in-flight dynamic partial reconfiguration, i.e. exchanging parts of user logic while the remaining user logic is still operating. Therefore an advanced System-on-Chip (SoC) can be implemented with such devices, however overall system availability and qualification has to be guaranteed in the harsh space environment.

We have already demonstrated the successful usage of SRAM-based FPGA devices for scientific instruments with e.g. the Venus Monitoring Camera (VMC) on ESA’s Venus Express mission launched 2005 [1] and the Dawn Framing Camera on NASA’s Dawn mission launched 2007 [2]. VMC was the first European SoC computer in space and to date is operational since more than 7 years, with only a few numbers of predicted resets due to radiation induced Single Event Effects (SEEs). But, the reconfiguration ability was only used during the development phase on ground and no support for in-flight reconfiguration was built-in. To be able to update the processing modules, the reconfigurability of these SRAM-FPGAs has to be used also in space. This is a major improvement in terms of maintenance and performance, which is essential for future space instruments because of its ability to adapt to unforeseen situations and events.

Since complete parallel real-time processing is not achievable in most cases and not all of the functional modules need to operate concurrently, it would be sufficient if a Partial Reconfigurable Module (PRM) could be requested to be instantiated and run in a FPGA on demand. The ability of SRAM-based FPGAs to support dynamic partial reconfiguration allows this flexible use of the available HW platform in a Time Space Partitioning (TSP) manner even for complex algorithms.

II. SoCWire

The SpaceWire based System-on-Chip Wire (SoCWire) communication network has been developed by IDA as a fault tolerant high-speed Network-on-Chip (NoC) architecture, which is able to connect PRMs to a host system with the capability to isolate these PRMs logically and physically from the system [3]. This means that glitch effects, which may occur during the reconfiguration process of PRMs, do not affect the operation of the host system and thus SoCWire provides a safe way to dynamically reconfigure parts of the FPGA during flight.

A SoCWire link is always a point to point connection of two CODECs with receiver and transmitter interface. Since normally more than two nodes need to be connected, a simple path addressing SoCWire switch with round robin scheduling is placed between them, see figure 1. The data transfer in a SoCWire network is controlled and supervised by a host system. Typically the host system consists of a variant of the space qualified LEON processor. As a bridge between the AMBA based host processor bus and the SoCWire network, the AHB to SoCWire bridge (AHB2SOCW) was developed [4], supporting 16bit and 32bit SoCWire networks. To provide highest data rates with low host processor involvement, the AHB master of the bridge is controlled by two Direct Memory Access (DMA) engines. FIFO buffers establish the connection between DMA controller and SoCWire CODEC.

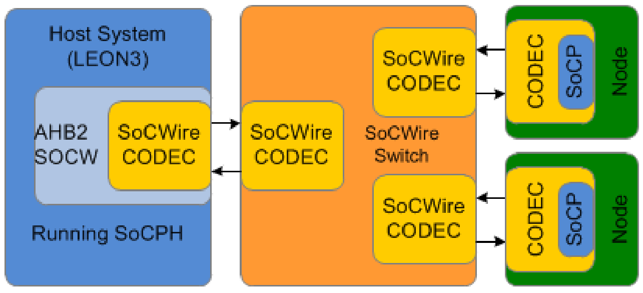


Figure 1. Basic SoCWire Network

III. SoCWire Protocol

Whilst the pure SoCWire network represents only the physical link and enables the data transmission between several SoCWire nodes within the on-chip network, a protocol is required to define rules and conventions for the communication between the different nodes. The nodes have to know how to interpret the meaning of received data. Our SoCWire protocol (SoCP) presented in [5] is inspired by the Remote Memory Access Protocol (RMAP), but adapted to the requirements for on-chip data processing chains and considerably simplified to limit resource consumption. Whereas RMAP is mainly used for remote memory accesses to nodes in a SpaceWire network, the processing nodes within a SoCWire network rather process the data on consecutive data blocks. In the context of a macro pipeline one processed data block is directly transmitted to further processing nodes or into mass memory.

The setup is such that the hardware protocol controller is placed between the SoCWire CODEC and the actual processing core as depicted in figure 2. The core creates replies to requests send, supplies the processing core logic with streaming data, and provides read and write registers to set and read parameters from the processing core. The implementation of user registers is optionally set by generics, which helps to avoid FPGA resource consumption when not needed. To make the protocol efficient and limit the registers in the protocol handler, the network must not have more than 3 switches and port 0 in every switch must be the route to the host processor.

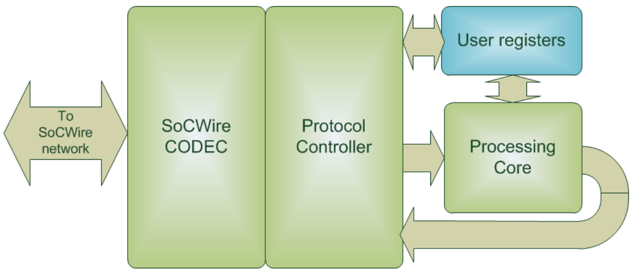


Figure 2. Detailed SoCWire Hardware Node

The general protocol structure is shown in figure 3. Every packet begins with up to 3 address descriptors, which contain the outgoing port number of one SoCWire switch on the path to the destination node. The destination is system wide uniquely determined in the Hardware ID field. The Transaction ID contains a counter in order to assign a response to a request, while Instruction tag identifies the packet type and additionally contains some flags for error handling. Each packet ends with an end of packet (EOP) behind the payload data.

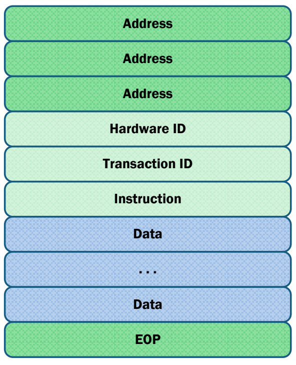


Figure 3. SoCWire Protocol Structure

IV. Software SoCWire Protocol Handler

The NoC and processing nodes need to be controlled and supervised by software on a host processor system. Depending on the number of nodes in a NoC, frequency of node replacement and maximum hop count generating suitable routing information the network management should not be underestimated in terms of complexity. Furthermore, developers of scientific application software have to keep track of initial NoC configuration and necessary changes over time. The host processor is in charge of all register reads, register writes, process requests and reception of plug and play messages, however streaming packets can be passed between nodes autonomously. Before nodes can pass streaming packets the path has to be set by the host processor otherwise all packet would be sent to the host processor by default. In principle high level software must be aware of the forwarding path to every PRM. Every node device driver has to be set up and updated separately. If reconfiguration is done this leads to the necessity of updating one or a whole set of software entities at the same time.

The main focus of the software SoCWire Protocol Handler (SoCPH) is on abstracting the NoC complexity and management overhead from the application software development. Therefore, the basic SoCPH design consists of four main components:

- Packet Header Generator,
- Response Dispatching Machine,
- NoC State and Routing Information Database (SRID),
- NoC State Surveillance Score.

The overall integration scheme and components are shown in figure 4. The SoCPH obtains control over the NoC by accessing the low-level SoCW interface driver. The SoCWire driver communicates with the hardware itself via shared circular DMA buffers. Towards the device node driver, the SoCPH provides a set of procedures mapping the SoCP.

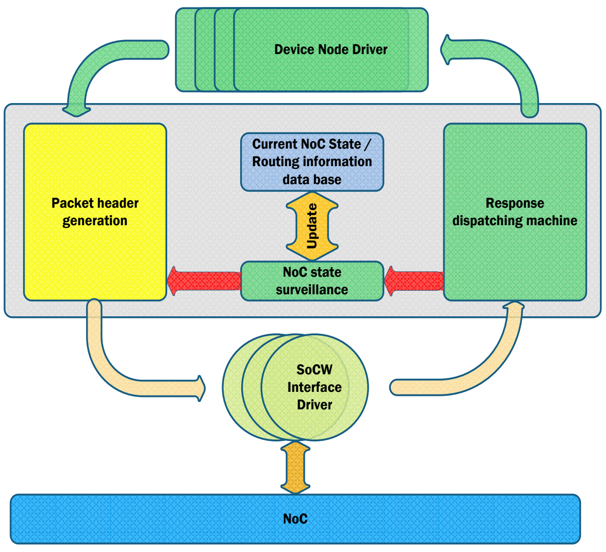


Figure 4. SoCPH Integration Scheme and Components

A. NoC State & Routing Information Database

SRID is allocated in memory of the host processor. It reflects the currents state of NoC configuration. Based on unique Hardware ID information as primary key it contains corresponding network path information from the host processor to destination PRM, including responsible SoCW interface driver instance.

B. NoC State Surveillance

After system power on and initial FPGA configuration or partial reconfiguration, initiated PRMs generate a Plug and Play Initialization Notification (PPIN) and transmit it via SoCP to the host processor. The PPIN contains the unique Hardware ID of a PRM. This information is received and interpreted by the NoC state surveillance monitor core, which updates the SRID by adding changed node information accordingly. In order to keep the database up to date, the monitor is triggered each time an invalid Hardware ID flag is set within incoming packets. Furthermore, SoCW switches transmit Plug and Play Link Error Notifications (PPLENs) in case of discontinuation of an established link, comparable to the PPIN. Any PPLEN or invalid Hardware ID flag set forces the NoC state surveillance monitor to remove or update the link information in SRID. The surveillance monitor is able to poll periodically for node status register information within an adjustable interval. So otherwise unrecognized errors might be detected and corrected in SRID.

C. Packet Header Generation

With regard to the SoCP design described in [6] every successfully configured PRM in the NoC setup is uniquely identified via its Hardware ID. So data can easily be transmitted to the corresponding PRM by calling the SoCPH primitives with the corresponding Hardware ID and the information to be send without knowledge of any routing information. Data is automatically split to fit into separate packets. Address descriptors are added accordingly to the current NoC state available in the SRID. Transaction IDs are prepared by the SoCPH and the Instruction Code is set correctly.

D. Response Dispatching Machine

Every packet received by one of the SoCW interface drivers is accepted by the dispatching machine and forwarded to the destination task depending on the Transaction ID, Instruction Code or Error Flags. In case of direct register read or write access, the interaction is mapped onto a synchronous network communication, which blocks calling task as long as register content is not responded from the PRM. This capsules NoC access, so the request appears to be satisfied locally for the developer. The SoCP processing and streaming requests are implemented as asynchronous transfers since computation might consume some time or results are not transmitted back to the host processor in a direct manner. For such asynchronous transfers, the SoCPH provides an internal buffer management which allows tasks to allocate incoming data buffers if a response is anticipated from a dedicated PRM. This will enable the software to process different tasks instead of being stalled. Since in SoCP each network node is treated equally including host processor, streaming case is not limited in the direction towards a PRM. Streaming data from PRM to host processor is also possible. The SoCPH allows registering a so called input stream handler function to every PRM existing in the current NoC. If streaming packets are received, the SoCPH autonomously invokes a registered handler to process the incoming packet. This avoids buffer overhead and the need to continuously poll for streaming information in the device node driver.

Depending on the underlying platform the SoCPH can be easily extended with more sophisticated functions by combining the previously described operations to hide complex setup procedures from high level software developer, e.g. reconfiguration of the whole FPGA platform or a single PRM.

V. Conclusion

In-flight reconfigurability enhances space applications with both, maintenance and performance improvements. Dynamic reconfiguration enables mission specific adaptability on demand and adds significant operational flexibility to the instrument. Thus, this is a favorable solution for the sophisticated data processing requirements within the very tight power and thermal constraints of scientific space missions.

SoCWire combined with SoCP supplies a highly flexible hardware and NoC platform solution. It enables on the fly partial reconfiguration of PRMs to accomplish changing processing demands over time without interfering any concurrently active PRM. The described SoCPH implementation on top provides an easy to use management and communication framework for SoCP based NoC. Processing modules s can be accessed without knowledge of network setup. Included functionality widely hides the overall network complexity and necessary administration workload from the high level application developer and user. In symbiosis with autonomous path determination and NoC surveillance it provides an easy to use framework for partial reconfigurable systems without loss of flexibility.

References

[1] B. Fiethe, H. Michalik, C. Dierker, B. Osterloh, G. Zhou, Reconfigurable System-on-Chip Data Processing Units for Miniaturized Space Imaging Instruments, Proceedings of the conference on Design, automation and test in Europe (DATE), pp. 977-982, ACM, 2007, ISBN 978-3-9810801-2-4

[2] H. Sierks, H. Keller, R. Jaumann, H. Michalik, T. Behnke, F. Bubenhagen, I. Büttner, U. Carsenty, U. Christensen, R. Enge, B. Fiethe, P. Gutierrez Marques, H. Hartwig, H. Krüger, W. Kühne, T. Maue, S. Mottola, A. Nathues, K.-U. Reiche, M. Richards, T. Roatsch, S. Schröoder, I. Szemerey, and M. Tschentscher, The Dawn Framing Camera, Space Science Reviews, vol. 163, pp. 263–327, 2011

[3] B. Osterloh, H.Michalik, B. Fiethe, “SoCWire: A SpaceWire inspired fault tolerant Network-on-Chip for Reconfigurable System-on-Chip Designs in Space Applications”, ISC, Nara, Japan, 2008

[4] H. Michel, F. Bubenhagen, B. Fiethe, H. Michalik, “AMBA to SoCWire Network on Chip Bridge as a Backbone for Dynamic Reconfigurable Processing Unit“, AHS, San Diego, California, USA, 2011

[5] F. Bubenhagen, H. Michel, H. Michalik, B. Fiethe, B. Osterloh, W. Sullivan, A. Wishart und J. Ilstad, "IMPLEMENTATION OF THE SOCWIRE PROTOCOL (SOCP) WITHIN THE DYNAMIC RECONFIGURABLE PROCESSING MODULE" in Proceedings of the 4th International SpaceWire Conference (Space Technology Centre, Ed.), San Antonio, United States, 2011.

[6] H. Michel, A. Belger, F, Bubenhagen, B. Fiethe, W. Sullivan A.Wishart and J. Ilstad, “The SoCWire Protocol (SoCP): A Flexible and Minimal Protocol for a Network-on-Chip”, in Adaptive Hardware and Systems (AHS), 2012 NASA/ESA, 2012

MASCOT On-Board Computer Based on SpaceWire Links

SpaceWire Onboard Equipment and Software
Poster

Sandi Habinc, Anandhavel Sakthivel, Jonas Ekergarn, Arvid Björkengren
Aeroflex Gaisler AB
Gothenburg, Sweden
info@gaisler.com

Richard Pender
Pender Electronic Design GmbH
Zurich, Switzerland
richard@pender.ch

Sven Landström
Hirel Design
Oegstgeest, The Nederlands

Federico Cordero, Jose Mendes
Telespazio VEGA Deutschland GmbH
Darmstadt, Germany

Tra-Mi Ho, Kai Stohlmann
Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Bremen, Germany

***Abstract*— Aeroflex Gaisler (SE) has together with Pender Electronic Design (CH) and Hirel Design (NL) has under DLR (D) contract and VEGA (D) management developed the Onboard Computer (OBC) engineering model (EM) for the MASCOT asteroid lander.**

***Index Terms*—SpaceWire, Networking, Spacecraft Electronics**

I. Introduction

The general concept of the “Mobile Asteroid Surface Scout” (MASCOT) is to provide a small landing system intended to be deployed from a supporting main spacecraft. It is specifically designed to be compatible with JAXA’s Hayabusa 2 (HY2, scheduled for launch in 2014) mission design and the environment given by the target asteroid 1999JU3.

Two major mission phases can be defined for MASCOT: A cruise phase, attached to the main-SC, lasting about 5 years, and a nominal phase, detached from the main-SC, lasting about 16 hours.

The design foresees an on-board computer (OBC) for gathering, processing, compressing and storing of the scientific payload and the housekeeping data and to run system and subsystem tasks.

II. Overall Concept

The MASCOT OBC comprises two fully redundant CPU and IO boards, where the CPU boards are operated in cold redundancy and the IO boards in hot redundancy.

The OBC is composed by a set of boards in the common electronics box.

The two sides of the OBC are named OBC-M (Main) section and OBC-R (Redundant) section and are power by two independent power supply lines from the PDCU.

The OBC has the following main functions:

- Interfacing with the payload instruments for commanding, housekeeping and science data acquisition.
- Interfacing with the lander’s equipment, like the power subsystem, mobility mechanism, attitude sensors for commanding and housekeeping data acquisition.
- Interfacing with the RF communication equipment (COM) to transmit CCSDS telemetry packets to and receive CCSDS telecommand packets from HY2 SC, which routes the packets to/from the ground.
- Execution of specific algorithms for science data processing, data reduction into scientific data products for storage into mass memory and downlink.
- Execution of specific algorithms for the mobility equipment and guidance/navigation on the asteroid.
- Overall autonomous control and management of the lander: Due to the nature of the mission and its short lifetime, operations will be highly automated on-board with no ground intervention. This includes the nominal mission timeline, which will be mainly event driven, and the Failure Detection, Isolation and Recovery (FDIR). Command and control capability from the ground, via HY2 SC is however foreseen, but its nominal use is limited during the cruise phase, mainly for updating the on-board software and mission parameters

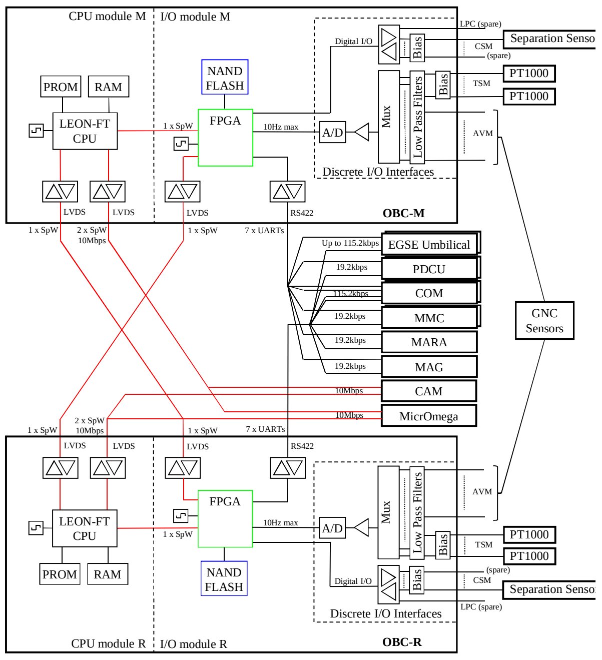


Fig. 1. Overall OBC structure

III. MASCOT CPU Board

The MASCOT CPU board is based on the GR712RC device, which is System-on-Chip (SoC) is a dual core LEON3FT system suitable for advanced high reliability space avionics. It is the first of its kind, offering the space community powerful multi-core processor capability in combination with multiple RMAP enabled SpaceWire links fully compliant with ECSS standards. The device is configurable and can operate in many different applications, ranging from platform to payload processing.

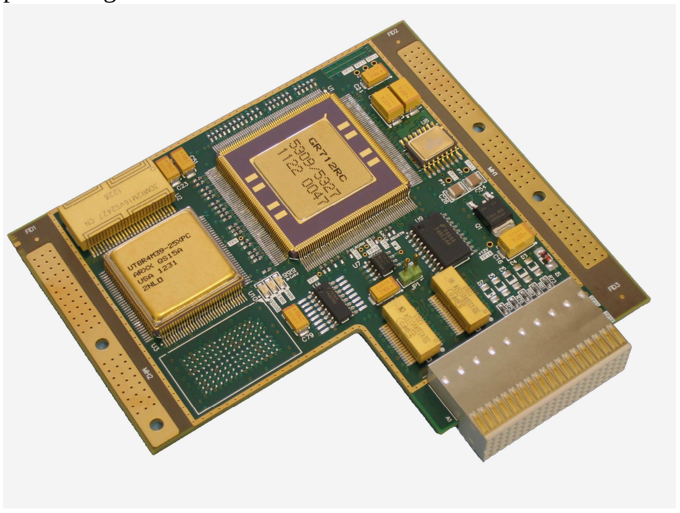


Fig. 2. MASCOT CPU board

The board comprises MRAM, SRAM and optionally SDRAM memory. External LVDS drivers are provided for four SpaceWire links, of which two are used for communication with the nominal and redundant MASCOT IO boards. It includes local voltage regulation is for the processor core voltage, as well as local power down of the analogue acquisition functionality for optimized cold sparing.

IV. MASCOT IO Board

The MASCOT IO board is based on Microsemi RT ProASIC3 FPGA technology.

The FPGA implements two SpaceWire links with RMAP target support in hardware. It provides an SPI interface towards a multichannel ADC, control of additional analogue multiplexers, including automatic sequencing for the analogue acquisition, several UARTs with support for large (up to 2KiB) buffers, control of low power commands (LPC) and digital sensor logic (CSM), a parallel interface to a NAND Flash memory which is protected by a Reed-Solomon code, and finally switch over (SO) control logic.

The SO logic is in charge of the overall OBC supervision, reconfiguring to the redundant CPU board in case of hardware or software anomaly, independently from the OBC software. All the resource can be accessed from the nominal and the redundant MASCOT CPU board.

V. Analogue Acquisition Chain

Analogue acquisition provides 15 fixed differential analogue acquisition channels (AVM), 12 biased PT1000 acquisitions (TSM) and 4 channels being configurable between AVM and TSM mode. The analogue acquisition system is implemented with additional HW to support an in-orbit SW calibration procedure of offset drifts and SW FDIR detection.

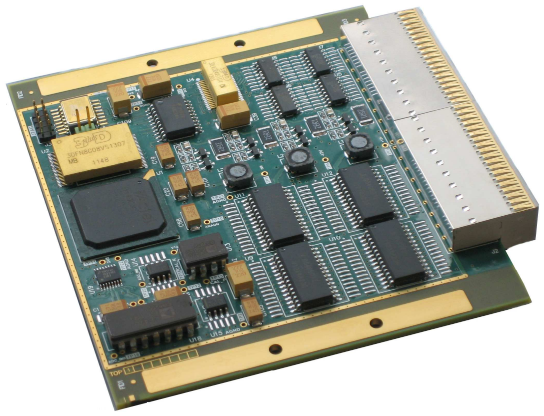


Fig. 3. MASCOT IO board

VI. FPGA ARCHITECTURE

The FPGA architecture is based on the on-chip AMBA bus, which is supported both by ESA's and by Aeroflex Gaisler's IP cores. It is therefore a very open architecture into which cores from different sources can be integrated.

Aeroflex Gaisler has extended the AMBA on-chip bus with a plug-and-play capability at the IP core level, which can be utilized by software developments tools and device drivers for real-time operating systems, as explained further down.

The plug-and-play information on IP core level allows for distributed address decoding, interrupt steering, etc. This enables automatic generation of a table including vendor and device identifier for each core, including version and interrupt information. Software and hardware debuggers can scan the table to install corresponding drivers etc.

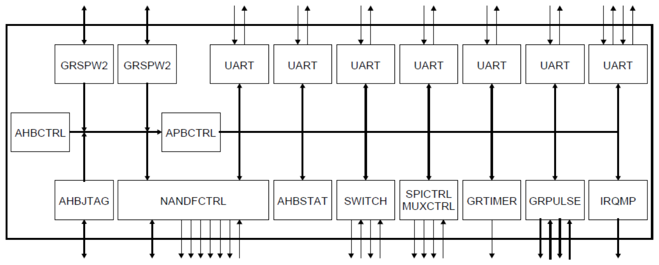


Fig. 4. MASCOT FPGA architecture

VII. USE OF SPACEWIRE AND RMAP

Each MASCOT CPU board has four SpaceWire links, of which two are used for communicating with the payload, and two are used for internal communication with the FPGAs on the IO boards. There is thus no direct SpaceWire connection between the two CPU boards.

The communication between processor on the MASCOT CPU board and the FPGA on the MASCOT IO board is done by means of RMAP over the two internal SpaceWire links. Via RMAP read and write commands the device status can be observed and it can be controlled in a safe (verified-write command) and standardized way (ECSS standard).

The processor does not need to implement RMAP in hardware. An RMAP initiator can be any device that can generate standard SpaceWire packets. The RMAP command is just a SpaceWire packet sent from the processor using its SpaceWire core. The RMAP response is just a SpaceWire packet sent from the TC FPGA to the processor. A complete RMAP initiator software stack has been implemented for the RTEMS real-time operating system which has been used to demonstrate the functionality of the system.

The processors on the CPU boards are connected both the FPGA on the nominal and the redundant IO board. This way the active processor can access all redundant external interfaces and on-board resources such as the NAND Flash memory.

The SpaceWire node in the FPGA has been based on the GRSPW IP core. The core is configured in an RMAP target only configuration, which means that it is not capable of initiating any SpaceWire transmission on its own, with a master interface to the internal AMBA bus in the FPGA.

VIII. SPACEWIRE ROUTER

An enhancement of the overall redundancy and communication concept is to replace the two GRSPW IP cores located in the FPGA with a three port SpaceWire router. The router has two SpaceWire ports and an internal AMBA master port with a built in RMAP target, thus similar interfaces as used above.

The main difference is that the routing functionality would allow one processor to access the memory space and the Debug Support Unit of the other processor, via either of the two FPGAs. This requires that both processors are powered, which is possible in the architecture. The benefit is that the active processor can modify the contents of non-volatile memory on the non-active processor, or upload software directly to volatile memory, etc.

This remote debug scenario via SpaceWire has previously been demonstrated in an ESA activity, where it was shown that for example a star-tracker could operate without the need for power consuming PROM memories, since there was always the possibility to upload software to the SRAM via the SpaceWire RMAP protocol.

The SpaceWire router functionality has been implemented in the FPGA and initial tests have been performed successfully. The GRSPWROUTER IP core from Aeroflex Gaisler has been used. The router functionality had to be reduced since the FPGA was already completely filled. This specific FPGA implementation does not support logical addressing; it is instead restricted to path addressing which reduces the need for a routing table. The AMBA master port has also been limited to only implement the RMAP target functionality, thus no accesses can be initiated from the inside of the FPGA on the AMBA side. This is the same limitation as for the GRSPW usage discussed above.

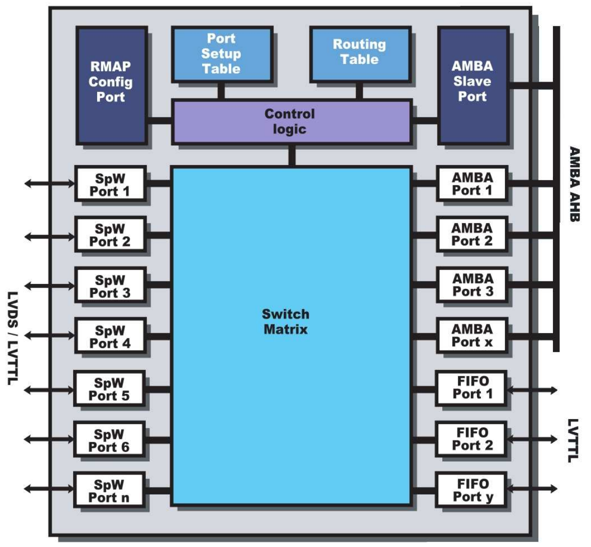


Fig. 5. GRSPWROUTER IP core

IX. EGSE

The EM verification and software development is performed using the MASCOT EGSE, which is a 19” crate with two internal backplanes, one for power conditioning and distribution, and one hosting two CPU boards and two IO board and also providing all external connectors for interfaces such as SpaceWire, UART, PT1000 elements, JTAG debug etc.

The MASCOT EGSE emulates as complete redundant OBC. The power power conditioning and distribution allows the nominal and redundant lane to be individually powered and can be controlled remotely via a digital test interface, allowing integration in the overall test equipment.

The backplane interconnecting the CPU and IO boards implements the routing of all internal SpaceWire links. The four external SpaceWire links (going to the payload) are routed to MDM9 connectors on the front-panel.

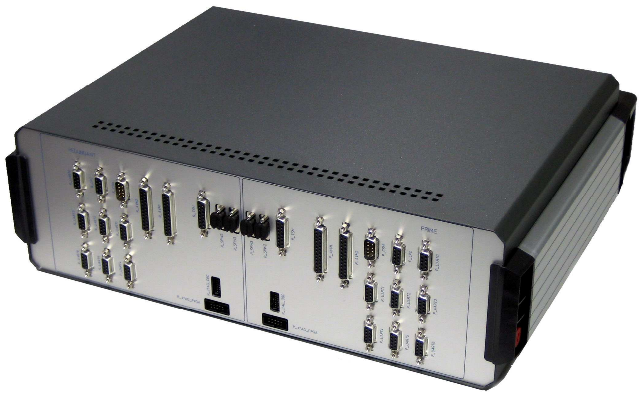


Fig. 6. MASCOT EGSE

X. The use of the EGSE

One of the objectives of the EM boards and EGSE is to support the MASCOT OBC Flight Software (FSW) development, done by VEGA. In this context the EM boards will be integrated in a Software Development and Verification Facility (SDVF), which will provide the I/O acquisition/stimuli, enabling FSW closed loop testing with the OBC Hardware In the Loop (HIL).

The SDVF is based on existing ESA SimSat kernel and provides a complete real time simulation environment of the MASCOT subsystems, including the payload SpaceWire links to the OBC. The FSW uses RODOS as RTOS and is developed in C++, applying a tailored version of JSF++ standard.

The OBC EM is also used by DLR for functional system and spacecraft level integration and testing.

The current MASCOT OBC engineering model is based on the latest GR712RC dual-core LEON3FT technology with SpaceWire links being used for both internal and external communication, utilizing the RMAP protocol to its full.

XI. Conclusion

The MASCOT development has shown that it is feasible to develop highly miniaturized spacecraft control systems based on SpaceWire networks. The SpaceWire network has been used for both control and for payload data, utilizing pre-existing building elements such as the GR712RC LEON3-FT dual core processor and SpaceWire controllers from the GRLIB IP core library.

The replacement of the SpaceWire controllers with a SpaceWire router has added additional capabilities to the spacecraft, enhancing the redundancy concept.

SpaceWire-HS Host Adapter – An FPGA based PCI Express Device for Versatile High-Speed Channels

Poster Session

S. Jörg, M. Nickl, T. Bahls, and S. Strasser
Robotics and Mechatronics Center
German Aerospace Center (DLR)
Oberpfaffenhofen, Germany
stefan.joerg@dlr.de

***Abstract*—Robotic systems like the DLR Hand Arm System that feature control cycles beyond 1 kHz demand a deterministic and low latency communication. Therefore, DLR is working on high-speed SpaceWire. This paper presents the SpaceWire-HS host adapter, a FPGA driven PCI Express device for high-speed SpaceWire. The adapter provides a generic host interface for QNX real-time hosts, supported by a client C++ library. Two implementation variants of the adapter's communication architecture and host interface are presented. The performance of both variants in terms of bandwidth and latency is discussed.**

***Index Terms*—host adapter, high-speed SpaceWire, robotics**

I. Introduction

DLR has been using SpaceWire as communication backbone for several of its lightweight robots. The latest and most complex system using SpaceWire is the DLR Hand Arm System, an anthropomorphic arm that comprises of 52 motors and over 430 sensors. To operate that many actuators and sensors precisely at high feedback control cycles beyond 1 kHz requires deterministic system behavior and low communication latencies. To achieve this, DLR implemented a 1 GBit/s SpaceWire modification (see [1]). Currently, DLR is working on a more efficient implementation of high-speed SpaceWire links capable of providing more than 1 GBit/s bandwidth [2].

The interface of the SpaceWire communication backbone to the PC-based real-time control hosts is a crucial point of the communication infrastructure. There the network implementation meets the non-determinism of a state-of-the-art workstation and its memory-based peripheral interface, PCI Express (PCIe).

To benefit from the performance of the high-speed links the authors have designed the SpaceWire-HS host adapter, a PCIe interface card that features four physical high-speed links (see Fig. 1). The SpaceWire-HS host adapter is equipped with a Xilinx Virtex-5 FPGA that implements the SpaceWire communication architecture. Thus, the communication architecture can be easily adapted to application requirements. Nevertheless, the SpaceWire-HS host adapter is designed as a general-purpose SpaceWire network endpoint.

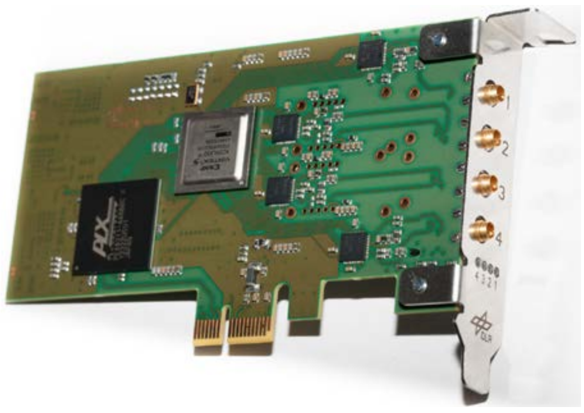


Fig. 1. The SpaceWire-HS Host Adapter Board with four copper HS-Links

Instead of increasing the performance by implementing application-specific algorithms and data structures on the host adapter FPGA, the focus is on the implementation of general-purpose packet channels whose performance in terms of bandwidth and latency can be configured by only a small set of parameters (e.g. buffer size).

This paper presents the communication architecture of the SpaceWire-HS adapter including two implementation variants. The next section introduces the hardware architecture of the SpaceWire-HS host adapter. Section III and IV present two variants of host interface implementation, which consists of the FPGA firmware and a C++ library. Section V presents the experimental evaluation of both implementation variants.

II. The SpaceWire-HS Host Adapter Architecture

The hardware design of the SpaceWire-HS host adapter is based on the design presented in [3]. Fig. 2 depicts the host adapter architecture. The PCI Express interface is implemented with a PLX PEX 8311 ExpressLane Bridge chip (see [5]), a one-lane master-capable host interface. A Xilinx Virtex-5 (5VLX50) FPGA connects PCIe interface, flash memory, housekeeping infrastructure and four physical layer interfaces. Four Texas Instruments TLK1221 IEEE802.3 Gigabit Ethernet compliant physical layer circuits implement the physical layer interfaces. Character encoding and link layer are implemented by the firmware on the FPGA. Therefore, the support of the new SpaceWire-HS high-speed link protocol (see [2]) only required the adaption of the FPGA firmware.

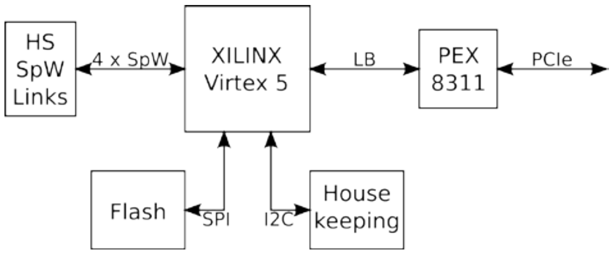


Fig. 2. The SpaceWire-HS Host Adapter Architecture

The Printed Circuit Board (PCB) format of 68.9x119.0mm conforms to the PCIe low profile form factor. Thus, it also fits into small-form factor cases. The board can alternatively be equipped with up to four fiber and/or copper links. Fig. 1 shows the adapter PCB with four copper HS-Links.

III. The Communication Architecture

The communication architecture has two main parts: The local SpaceWire Routing Switch and the Host Interface (see Fig. 3). The four physical SpaceWire-HS links are connected to the local Routing Switch, which is implemented as a standard SpaceWire wormhole routing switch. All HS-Links are configured for a fixed bandwidth and start automatically if connected to a peer.

The FPGA's configuration flash memory is connected via a SpaceWire/SPI bridge to the local routing switch. Thus, the FPGA's firmware can be programmed via the SpaceWire network.

For the implementation of clock synchronization, a configurable TimeCode (TC) master is connected to the local router. A Spacewire/I2C bridge provides access to the adapter's housekeeping infrastructure.

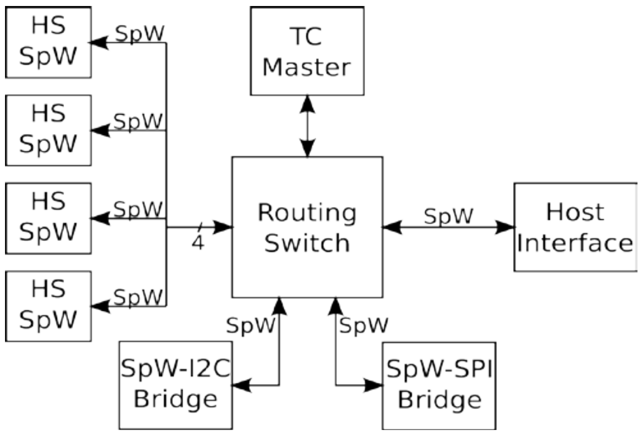


Fig. 3. The communication architecture consists of two main parts: A Routing Switch and the Host Interface. Two implementation variants of the Host Interface are discussed (see Fig. 4 and Fig. 5).

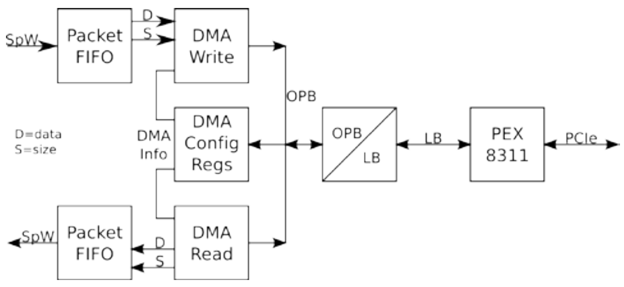


Fig. 4. Single-Channel Host Interface Architecture

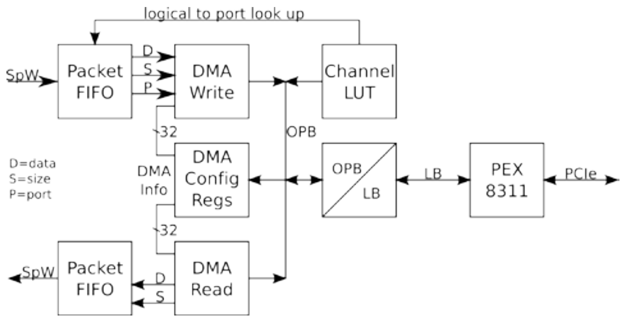


Fig. 5. Multi-Channel Host Interface Architecture

In the following, two implemented and evaluated variants of the Host Interface are presented. For both variants, the Host Interface consists of memory mapped status, configuration registers (DMA Config Regs in Figs. 4/5), and a number of DMA Read/Write interfaces. Those interfaces on the FPGA are connected via a CoreConnect On-Chip-Peripheral Bus (OPB). An OPB/PEX Local Bus bridge connects the FPGA's OPB to the PEX 8311, which implements the bridge to PCIe.

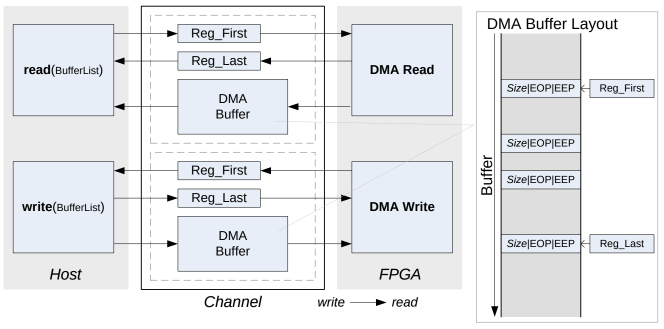


Fig. 6. A bi-directional ring buffer channel connects host software (read/write) to FPGA firmware (DMA Read/Write) via PCIe. The DMA Buffer layout is an in-place linked packet list.

The first variant is the Single-Channel Interface (Fig. 4). This implementation was already used in the SpaceWire-1Gb of the Hand Arm System [1] and is presented in more detail in [3]. The Single-Channel has one large ring buffer interface for each communication direction. Both the DMA Write and DMA read channel are a 1Mbyte fixed-sized ring buffer. The resulting implementation creates a bi-directional FIFO-channel from host software to FPGA firmware as depicted by Fig. 6.

The ring buffer synchronization is implemented as follows: Host Software and Firmware communicate via the shared

REG_FIRST, REG_LAST ring buffer registers (see Fig. 6). The ring buffer concept governs the concurrent access to these registers. The firmware is synchronized by simply polling those ring buffer registers. The host software is synchronized by interrupt. If enabled by the host software, the firmware raises a PCIe interrupt if the DMA read ring buffer is no longer empty or the DMA write ring buffer is no longer full.

Most applications require more than one endpoint at the host. With only a single channel, the host software needs to implement packet routing, which is an expensive operation. To avoid the packet routing on the host, the second variant provides 32 parallel endpoints to the host software.

Therefore, the Multi-Channel variant (Fig. 5) implements 32 configurable DMA read/write ring buffer interfaces. The functionality of each channel is the same as for the Single-Channel implementation. However, the host software configures the size of each ring buffer. Similar to a routing switch, the channel lookup table (Channel LUT) implements a routing table for the mapping of physical/logical ids to one of the 32 channels. The host software also configures the Channel LUT. Channel arbitration is implemented as a fair round-robin scheme.

IV. Client Packet Interfaces

The client packet interface implements the host software SpaceWire end-point interface. It consists of a POSIX I/O driver and a C++ software library. Both are implemented for the QNX 6.x real-time operating system. The I/O driver provides an open/close/read/write POSIX I/O interface for each of the host adapter's DMA channels. The driver uses the efficient message passing implementation of the QNX kernel to copy the data packets via read/write function calls. Thus, shared memory between client and driver is avoided.

As depicted by Fig. 6, packets are stored as an in-place linked list layout in the DMA ring buffer. Each packet in the ring buffer starts with a 4-byte header that contains a 16-bit packet size and packet tags, such as EOP, EEP.

The I/O driver read/write interface uses the same data structure to communicate with its clients. Thus, each call to read/write is able to transfer more than one packet. This is supported by the C++ client library class *over::pci::BufferList*, which implements the in-place linked list layout of the DMA ring buffer. Additionally, the C++ client library provides protocol-specific packet data structures, a routing switch implementation, end-point classes, a network topology configurator, and more convenient functionality. Therefore, an application does not need to use the I/O interface directly.

For example, the packet data structures allow an application to pre-allocate packets as required at an initialization phase. Then, during operation, only the payload of the packets in the pre-allocated data structure has to be updated. Fig. 7 exemplifies how the pre-allocation, update and send is implemented using the C++ library. Packet reception works similar. A blocking read on a DMA channel (done in link.receive()) yields all packets available in the DMA. Fig. 8 exemplifies how to iterate through all received packets and route them to their destination node (i.e. buffer).

```
// create packet buffer list
over::pci::BufferListInstance<> tx_pkts(10xspacewire::MAX_PACKET_SIZE);
...
network::Packet packet(25,2);  // 25 payload and 2 address bytes
packet.address()[0] = 42;
packet.address()[1] = 2;
tx_pkts.push_back( packet );  // add packet to send buffer
...
tx_pkts.push_back( packet2 );  // add another packet to send buffer
...
// update payload of every packet to be send
for ( over::pci::BufferList<>::iterator p= tx_pkts.begin(); p!= tx_pkts.end(); ++p )
        std::memcpy((*p).payload.data(),src.data(),(*p).payload.size());
link.send( tx_pkts );   // deliver all packets with one write() to driver
```

Fig. 7. Example of pre-allocated packets. Only the payload needs to be updated before sending all pre-allocated packets at once.

```
// create packet buffer list
over::pci::BufferListInstance<> rx_pkts(10xspacewire::MAX_PACKET_SIZE);
...
link.receive( rx_pkts );   // blocks until at least one packet arrives
for ( over::pci::BufferList<>::iterator p =  rx_pkts.begin(); p != rx_pkts.end(); ++p )
  {    // route all received packets
     network::Packet<BufferReference<> > recv(*p);
     Node& node = router_table->find(recv.address()[0]);
     node.push(recv);
  }
```

Fig. 8. More than one packet can be received with one read(). This example demonstrates, how to route every packet to its destination by simply iterating through the received packets.

A different I/O driver implementation is required for the Single-Channel and Multi-Channel variants since the I/O driver has to provide an open/close/read/write interface for each of the host adapters DMA channels. Since both variants use the same DMA channel layout, the same C++ library is used for the Single-Channel and Multi-Channel implementation variants.

V. Experimental Results

The performance in terms of bandwidth and latency of the two implementation variants Single-Channel and Multi-Channel has been experimentally evaluated. Therefore, the following experiments have been conducted for each variant:

1. **Roundtrip Latency:** Packets are looped via a HS link.
   measured: Roundtrip time for each packet
   parameters: packet size: 5-1017,
   parallel communication: 1-9 channels
2. **Receive Bandwidth:** External source sends packet to host. Packet load is increased to find the stable limit.
   measured: received packet bytes per second
   parameters: packet size: 9-1017,
   number of packets per second
   parallel communication: 1-4 channels

Both experiments were conducted with a DELL Optiplex Intel i7-3770 host running QNX 6.5 and a SpaceWire-HS card with four copper links. Fig. 9 depicts the setup for both experiments for the Single-Channel (top) and Multi-Channel (bottom) variant. The main difference in the test setup is the additional client library router required for the Single-Channel.

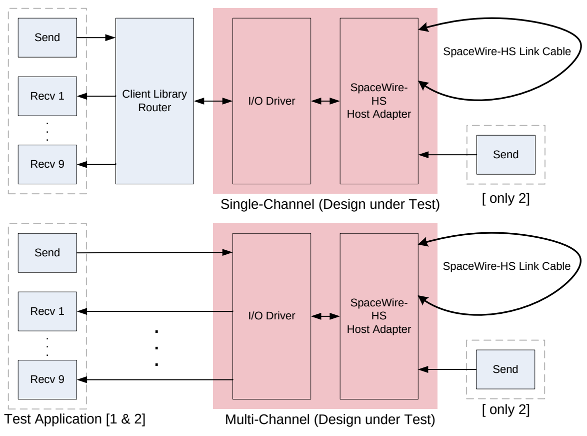


Fig. 9. The Experiment Setup for the Single-Channel (top) and Multi-Channel (bottom) variant

1) Roundtrip Latency

The test software consists of a packet source, a packet receiver for each channel, and a time measurement. The packet source sends a packet to 1-9 receivers, looped through a HS-link cable. The total time was measured it took to send all packets until the reception of all sent packets. Depending on the number of channels, 1-9 packets were send/received in one cycle. Each measurement was conducted for 30 seconds, which means 200.000-300.000 cycles.

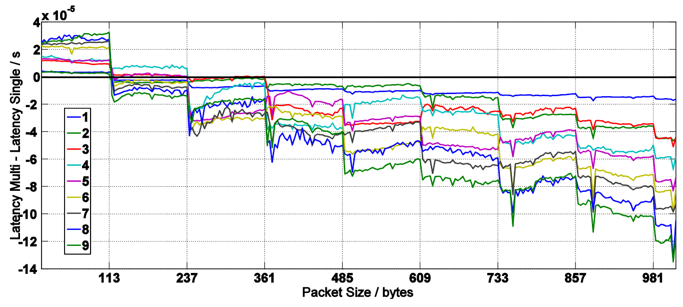


Fig. 10. Multi-Channel – Single Channel Roundtrip latency over Packet Size

Fig. 10 depicts the difference between the measured latencies $L_{\text{Multi-Channel}}$-$L_{\text{Single-Channel}}$ for 1-9 used channels over rising packet size. The steps that appear in the graph at 128 byte intervals are directly related to the PEX8311's 128 byte maximum payload size of a Transaction Layer Packet (TLP) (see [4]). For packets>237 bytes, the Multi-Channel is always faster, even if only one channel is used. This is because no software router is required. For packets of 113–237 bytes both implementations yield nearly the same latency. For packets<113 bytes the Single-Channel is slightly faster (~2us).

2) Receive Bandwidth

The second experiment uses an external packet source that sends packets of varying size at deterministic intervals to up to four destinations. For each packet size, the time interval between each packet was reduced until the stability limit of the host was reached. Fig. 11 depicts the measured maximum stable bandwidth over increasing packet size. The achieved bandwidth saturates at 57.7x10 bytes/s for the Single-Channel and at 61.0×10^6 bytes/s for the Multi-Channel. Both variants come close to the maximum input bandwidth, which is determined by the internal FPGA SpaceWire link rate of 62.5×10^6 bytes/s. Due to higher routing and DMA data handling effort small packets achieve only a lower bandwidth.

Since the Multi-Channel variant does not require packet routing by the host software, it saturates at a higher bandwidth and reaches that bandwidth for smaller packets.

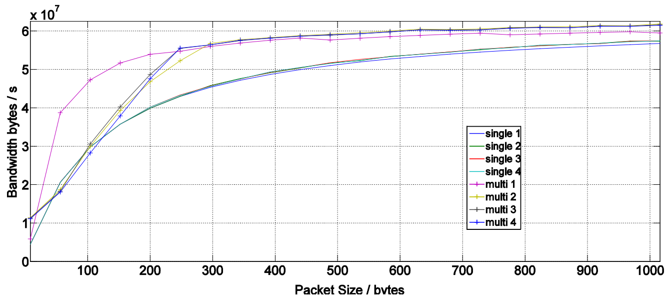


Fig. 11. Bandwith over packet size: Multi-Channel achieves higher bandwith

VI. Conclusions

The demand for a deterministic communication backbone with low latency motivated DLR's work on high-speed SpaceWire. For robotic applications, a deterministic host interface is an essential component of a high-speed SpaceWire network. Starting from the already available Single-Channel implementation, the goal has been to get a more flexible and efficient host interface, which features multiple endpoints. The challenge has been to achieve a deterministic and efficient behaviour despite the higher complexity of the Multi-Channel implementation. The results show, that not only the Multi-Channel implementation is deterministic but also matches the Single-Channel in performance.

Future work will be to add Quality-Of-Service parameters to the Multi-Channel implementation such as channel priority or guaranteed channel bandwidth. Furthermore, we intend to evaluate how the SpaceWire-HS host adapter with its four physical links can be used as a building block for very complex networks such as multi-robot systems.

References

[1] M. Nickl and S. Jörg, T. Bahls A. Nothhelfer, S. Strasser, "SpaceWire, A Backbone For Humanoid Robotic Systems", Int. SpaceWire Conference, San Antonio, 2011

[2] M. Nickl, S. Jörg, T. Bahls and B. Cook, "Towards High-Speed SpaceWire Links", Int. SpaceWire Conference, Gothenburg, 2013

[3] T. Bahls, "Entwicklung einer latenz- und bandbreitenoptimierten Bridge zur transparenten Anbindung von FPGAs an Standard-CPUs", Master Thesis (in German)

[4] R. Budruk, D. Anderson, T. Shanley, „PCI Express System Architecture", MindShare, Inc., Addison Wesley, 2004

[5] "ExpressLane PEX 8311 PCI Express-to Generic Local Bus Bridge Data Book", PLX Technology, www.plxtech.com, 2009

SpaceWire Standard Revision

SpaceWire Networks and Protocols session, Poster Paper

David Jameux
On-Board Data Systems (TEC-ED)
ESA/ESTEC
Noordwijk, Netherlands
david.jameux@esa.int

Antonis Tavoularis
Systems Engineering Group
Teletel SA
Athens, Greece
a.tavoularis@teletel.eu

Abstract— **In this paper, we recall the need for a revision of the current SpaceWire standard as well as the improvements foreseen to be developed, breadboarded and documented in ECSS standardisation format through the ESA/TRP activity "SpaceWire Evolutions". We inform about the final achievements of the project team. Finally, we propose solutions for the revised standard and provide justification for the minor improvements to be endorsed by the ECSS Working Group in charge of the standard revision.**

Index Terms— **SpaceWire standard revision.**

I. Introduction

Through several years of standardisation and technology development activities, ESA has prepared the SpaceWire technology communication protocol that allows embarking high speed data networks on board spacecraft. This new technology has become widely adopted not only by ESA missions but also by other agencies and industries. However, some evolutions of the SpaceWire standard have been proposed by the SpaceWire Working Group.

The working group identified shortcomings of the current protocol for the support of SpaceWire device/network discovery and configuration capabilities. The technical investigations on these issues also rose the awareness that the behaviour of "nodes" have to be clarified as well as their definition in the current standard, because this definition is not in line with international telecommunications core definitions of network items, and is in fact ambiguous. While the introduction of SpaceWire operating in half-duplex or simplex mode over wire-limited harness was discarded after thorough assessment of added value versus standardisation effort, sideband signalling for interrupt distribution will be added to the standard.

These limited evolutions to SpaceWire standard have been assessed, refined, and prototyped in the frame of the ESA/TRP "SpaceWire Evolutions" from May 2011 till December 2012.

II. The "SpaceWire Evolutions" ESA/TRP study

Within the SpW Evolutions project three evolutions, proposed during the SpW WG meetings, were addressed:

- Clarification of the node definition which is ambiguous in the existing specification
- Specification of an interrupts distribution mechanism
- Low mass SpaceWire through simplex and half-duplex SpaceWire

A. Terminology

Regarding SpaceWire node definition, the existing specification becomes ambiguous with the emergence of SpaceWire device/network discovery and configuration which constitutes link-configurable switches as sources or destinations of (configuration) packets. In addition, the current standard does not allow the design of SpaceWire-based System-on-Chip components implementing links using the SpaceWire level stack down to the Character Level only, since not conforming to the electrical and physical levels of the specification currently leads to a breach of compliance. Taking SpaceWire device/network discovery and configuration and complex devices as example cases, the study team performed a conceptual redesign of the SpaceWire standard a) proposing a clear distinction between the physical and functional networks with a layer structure similar to the OSI stack, allowing easier analysis and modelling of SpaceWire devices and traffic, b) allowing for devices under development and existing devices to be "SpaceWire compliant" (e.g. SpW10X FIFO ports) and c) proposing the use of new terms which remove the ambiguities of the existing standard and clearly specify the entities on which higher layer protocols will be based by providing the terms/resources for identification, configuration and device status acquisition.

B. Low latency signalling, Distributed Interrupts

The study team also analysed technical solutions for low-latency signalling of events in SpaceWire networks, from their functional, performance and operation-under-failure points of view, taking into account that the solution shall be interoperable with existing SpaceWire equipment, having as a basis a proposal by the St Petersburg University of Aerospace Instrumentation (SUAI) [24]. Within the frame of the project, the consortium collected requirements for signalling distribution from ESA and from the space industry, investigated proposed solutions and cross-checked them against the collected requirements and performed trade-off analyses for all of them. The project concluded with the definition of a solution based on the SUAI proposal, complemented with recommendations that a) make the solution more robust under the presence of failures, b) provide two options, with and without acknowledgement, in order to provide for flexibility on the use of the proposed mechanism, c) provide interoperability with existing equipment thus making the existence of legacy and new devices in the network possible. The proposed mechanism was described according to ECSS standardisation guidelines and was validated through two independent developments, including validation scenarios that proved issues related to the performance and operation under failure of the mechanism that should be described either as informative parts of the next SpaceWire specification or as sections in the SpaceWire handbook.

C. Simplex and Half-Duplex SpaceWire

For low mass SpaceWire the consortium performed work on two proposals, Simplex SpaceWire and Half-Duplex SpaceWire, both of which have been presented during the SpW Working Group meetings by SUAI [9][13] and 4Links Ltd [12][14][17].

Early in the project the study team revealed a number of issues in the Simplex SpaceWire proposal which make it unsafe for use in on-board networks. The issues were solved by the proposal of an updated specification, making Simplex SpaceWire robust under the presence of failures. The updated proposal was specified following the ECSS standardisation guidelines and validation scenarios for the functional, performance and operation under failures were defined.

The work for Half-Duplex SpaceWire included the review of the existing technical proposal which proved to have weaknesses from the robustness point of view under improbable, but theoretically possible, failures. For example, in the original proposal, the end of the link that was detecting an error (e.g. parity) was immediately returning to the ErrorReset state without waiting the other end to cease transmission; under rare cases this could result to link degradation with one end of the link possessing only one FCT instead of 7. An updated mechanism was proposed solving functional and robustness issues. The work continued with the evaluation of Half-Duplex SpaceWire from the performance point of view, including trade-off analyses of various parameters, in which it became apparent that Half-Duplex SpaceWire is a candidate for connection of devices at the periphery of SpaceWire networks only, due to its excessive worst-case latency for signalling characters. The proposed mechanism was specified following the ECSS guidelines and validation scenarios for the functional, performance and interoperability with existing devices and the rest of the SpW Evolutions were defined. The work concluded with the Half-Duplex SpaceWire modelling in VHDL, based on the UoD SpW Core, which is extensively used by ESA. This modelling proved that implementation of Half Duplex on mainstream IP Cores which follow the SpaceWire concept of recovering the remote end's transmission clock by XORing the Data and Strobe signals results in poor performance; and that, in order to support Half-Duplex, SpaceWire extensive modifications are required which make it an expensive solution in terms of gate count and power consumption.

III. Clarification of the SpW internal protocol stack

A. Clarification of the protocol levels

1) Merging levels: Like any communication protocol specifying a solution down to the communication media, SpaceWire is organised in layers. For SpaceWire, these layers are called “levels” and the clauses that specify the protocol are gathered into sections according to the “level” that they describe. Unfortunately, in ECSS-E-ST-50-12C, some levels define only part of what should be a consistent protocol layer (syntax, synchronisation, and behaviour). For example, the Packet Level specifies the syntax (the packet structure) while the Network Level specifies the behaviour (“packet routing”). We therefore propose to merge these two levels in a single “SpW Network Level”. Similarly, the ECSS-E-ST-50-12C Character Level specifies the characters to be used at link layer while the Exchange Level specifies the rules for synchronisation between these characters (finite state machine). We also propose to merge these two levels into a single “SpW Link Level”.

2) Clarification of ambiguous clauses: These merges allow as well easy positioning in the standard specification of clauses that were not placed correctly in ECSS-E-ST-50-12C because it was difficult to assess where they belonged. It even allows removing partial duplication of specification that is one of the causes of ambiguity of ECSS-E-ST-50-12C.

3) Introduction of Service Access Points: The new division of clauses into clearer levels allows introducing fully defined Service Access Points for each of these levels. While this is not necessary for component designers to implement a complete SpaceWire stack nor for the SpaceWire users, it allows diversification of the lower protocol levels, as detailed in section III.C.

The proposed revised internal stack is shown in Figure 1.

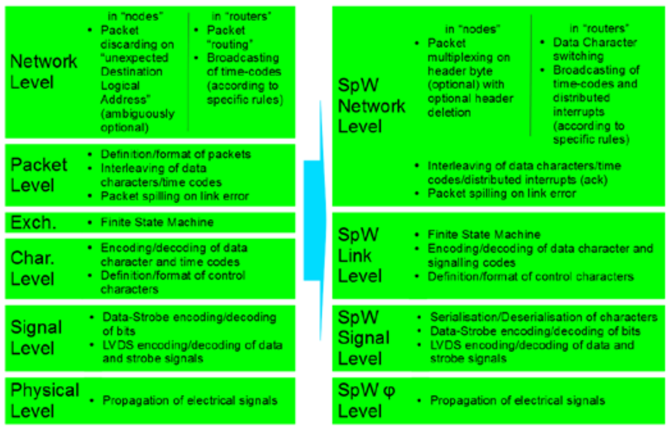

Network Level
in "nodes"
Packet discarding on "unexpected Destination Logical Address" (ambiguously optional)
in "routers"
Packet "routing"
Broadcasting of time-codes (according to specific rules)
Packet Level
Definition/format of packets
Interleaving of data characters/time codes
Packet spilling on link error
Exch.
Finite State Machine
Char. Level
Encoding/decoding of data character and time codes
Definition/format of control characters
Signal Level
Data-Strobe encoding/decoding of bits
LVDS encoding/decoding of data and strobe signals
Physical Level
Propagation of electrical signals
SpW Network Level
in "nodes"
Packet multiplexing on header byte (optional) with optional header deletion
in "routers"
Data Character switching
Broadcasting of time-codes and distributed interrupts (according to specific rules)
Interleaving of data characters/time codes/distributed interrupts (ack)
Packet spilling on link error
SpW Link Level
Finite State Machine
Encoding/decoding of data character and signalling codes
Definition/format of control characters
SpW Signal Level
Serialisation/Deserialisation of characters
Data-Strobe encoding/decoding of bits
LVDS encoding/decoding of data and strobe signals
SpW φ Level
Propagation of electrical signals

Fig. 1. ECSS-E-ST-50-12C stack versus ECSS-E-ST-50-12C Rev.1 stack

B. Clarification of the objects exchanged between protocol levels

As part of the Service Access Points, the new division of clauses into clearer levels allows clear specification of the objects exchanged between levels.

1) *Objects exchanged between the SpW user and SpaceWire (i.e. the SpW Network Level):* In ECSS-E-ST-50-12C, these are mainly packets, with some ambiguity whether time-codes are also exposed from the Network Level or directly from the Exchange Level. For ECSS-E-ST-50-12C Rev.1, we propose that a Packet Service and a Time-code Service are clearly exposed, with the addition of a Distributed Interrupt Service.

a) Packet Service: 8-bit Data Characters are exchanged between the SpW user and SpaceWire. The order in their sequence will determine the packet structure when these Data Characters reach a switch or an end-point.

b) Time-code Service: 6-bit Time-codes are exchanged between the SpW user and SpaceWire. For transmission and broadcasting, they are interleaved with Data Characters as in ECSS-E-ST-50-12C.

c) Distributed Interrupt Service: 5-bit Distributed Interrupts or Distributed Interrupt Acknowledgements are exchanged between the SpW user and SpaceWire. For transmission and broadcasting, they are interleaved with Data Characters as in ECSS-E-ST-50-12C.

2) *Objects exchanged between the SpW Network Level and the SpW Link Level:* In ECSS-E-ST-50-12C, these are not clearly defined. For ECSS-E-ST-50-12C Rev.1, we propose that they are the same objects than the ones exchanged between the SpW user and SpaceWire since the SpW Network Level only manipulates these objects (switching, interleaving, spilling, etc) but does not modify them.

3) *Objects exchanged between the SpW Link Level and the SpW Signal Level:* In ECSS-E-ST-50-12C, these are not clearly defined. For ECSS-E-ST-50-12C Rev.1, we propose that they are the objects exchanged between the SpW Network Level and the SpW Link Level coded as symbols (parity bit and control bit added for Data Characters, plus addition of an ESC symbol prefix for Time-codes and Distributed Interrupts/Acknowledgements) as well as the control characters necessary for the link finite state machine coded as symbols (parity bit and control bit added).

4) *Objects exchanged between the SpW Signal Level and the SpW Physical Level:* In ECSS-E-ST-50-12C, these are not clearly defined. For ECSS-E-ST-50-12C Rev.1, we propose that the SpW Signal Level be clearly divided into (from top to bottom) a Serialisation/Deserialisation sublevel, a Data-Strobe coding sublevel, and an LVDS signalling sublevel. The objects exchanged between the SpW Signal Level and the SpW Physical Level are then obviously four wave forms per bit.

The resulting internal protocol stack in shown in Figure 2.

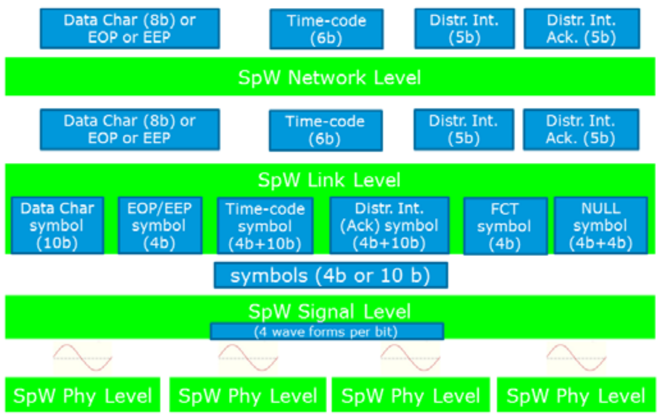


Fig. 2. Layering of the ECSS-E-ST-50-12C Rev.1 internal protocol stack

C. Diversification of the lower protocol levels

The definition of clear Service Access Points for each level of the SpaceWire internal protocol stack allows replacing part of this stack with alternative solutions for the lower protocol levels that provide different properties at link level, signal level, or for the physical media. This flexibility is illustrated in Figure 3.

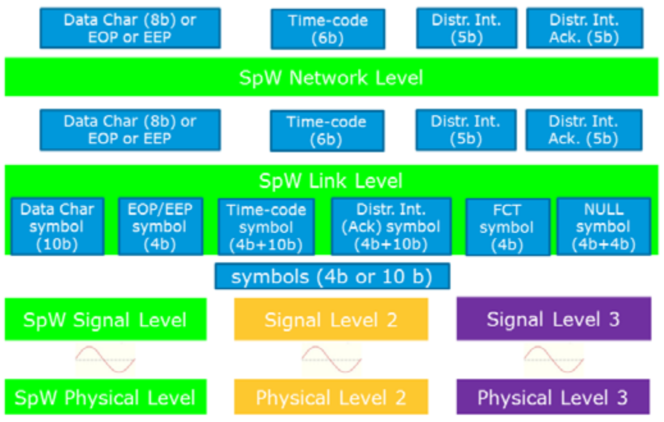


Fig. 3. Flexibility in the ECSS-E-ST-50-12C Rev.1 internal protocol stack

Of course, implementation of these Service Access Points must not be mandatory (this must be clearly expressed in ECSS-E-ST-50-12C Rev.1) since implementing interfaces between levels is not necessary (and most of the time sub-optimal) in a given implementation of the SpaceWire protocol stack. Existing devices or IP cores that do not implement these interfaces are therefore still compliant with ECSS-E-ST-50-12C Rev.1. But the newly introduced modularity allows implementing building blocks (e.g. IP cores) for the different parts of the SpaceWire internal protocol stack and assemble them into different complete stacks, as illustrated in Figure 4.

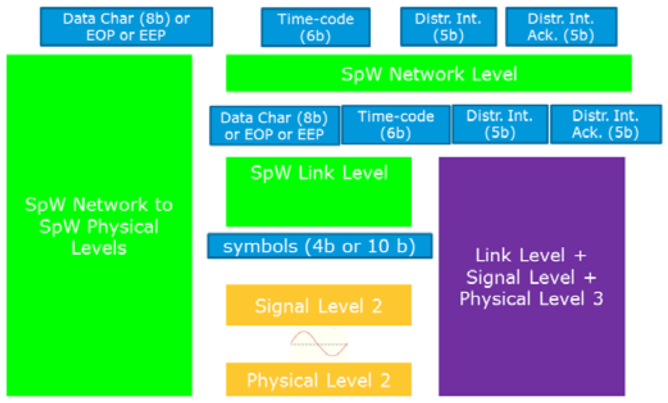


Fig. 4. Modularity of the ECSS-E-ST-50-12C Rev.1 internal protocol stack

This modularity allows benefitting from the recent technology developments such as the GigaSpaceWire (galvanic-isolated gigabit per second SpaceWire) proposed by SUAI [5][6] or the Virtual Channel SpaceWire proposed by 4Links Ltd [8][10] and will allow smooth integration of the SpaceFibre technology into SpaceWire networks, as illustrated in Figure 5.

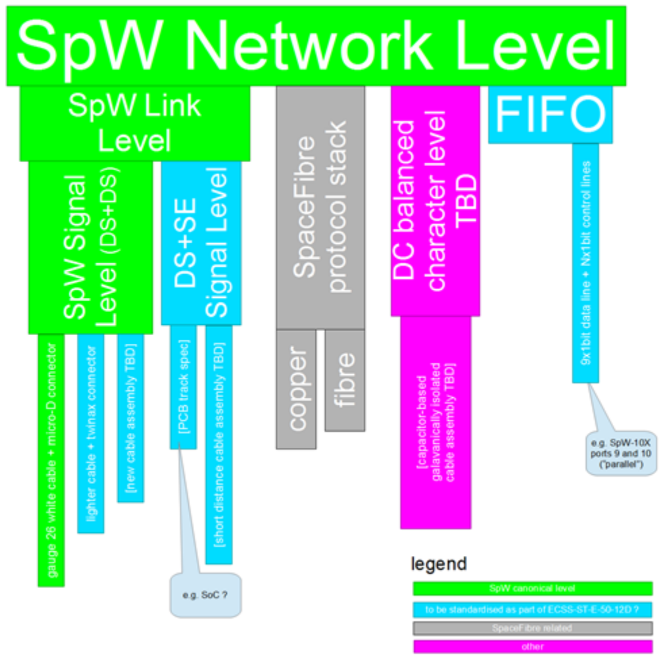

SpW Network Level
SpW Link Level
SpW Signal Level (DS+DS)
DS+SE Signal Level
SpaceFibre protocol stack
DC balanced character level TBD
FIFO
gauge 26 white cable + micro-D connector
lighter cable + twinax connector
[new cable assembly TBD]
[PCB track spec]
[short distance cable assembly TBD]
copper
fibre
[capacitor-based galavanically isolated cable assembly TBD]
9x1bit data line + Nx1bit control lines
e.g. SpW-10X ports 9 and 10 ("parallel")
e.g. SoC ?
legend
SpW canonical level
to be standardised as part of ECSS-ST-E-50-12D ?
SpaceFibre related
other

Fig. 5. Diversity of link, signal and physical solutions for SpaceWire links

D. Terminology – clarification of concepts for SpaceWire

The SpaceWire community identified a number of ambiguities in the current SpaceWire specification (ECSS-E-ST-50-12C) that are mostly due to inconsistency in the definition and usage of a few terms. One of the recommendations was to align the terms used for SpaceWire with international telecommunications core definitions of network items and concepts.

1) "Router" vs "switch": In *ECSS-E-ST-50-12C*, the elements that allow not only point-to-point communication but also SpaceWire networks are called "routers". This is confusing because "routing", i.e. "building a route" is only happening in the source node, when address data characters are assembled in sequence to form the address of a SpW packet (see Figure 6). This is true both for path address building and when the address contains Logical Addresses which are in fact "short-cuts", i.e. labels for path address sequences.

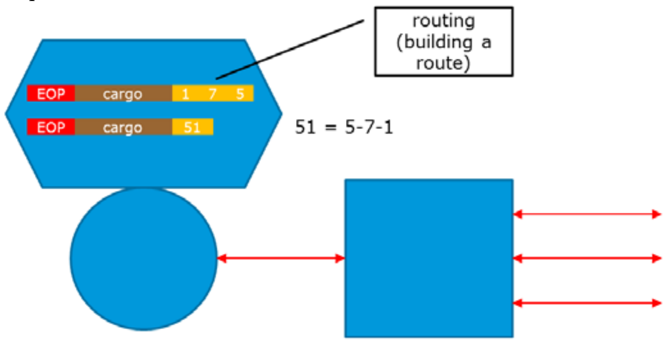


Fig. 6. "Routing", i.e. "route building" in a SpW source node

What happens next in a "router" as illustrated in Figure 7 is not "routing" but "switching": the first data character of a packet configures the switch for the duration/length of the packet (this configuration is done according to the value of this header byte, be it a Path Address or a Logical Address) and the following data characters are switched to the same output port until the end-of-packet marker included.

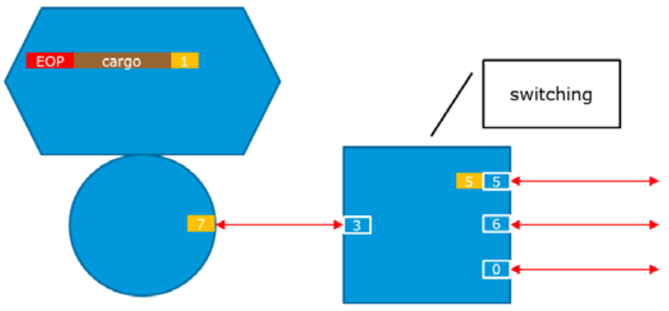


Fig. 7. Wormhole Switching in a SpW "switch"

It is therefore proposed that, in ECSS-E-ST-50-12C Rev.1, the network elements that do Wormhole Switching are called "switches".

2) End-points, nodes, and units: In ECSS-E-ST-50-12C, "nodes" are defined as "source and destination of packets". A "node" would therefore be the application driving one or more SpW protocol stacks (also called "higher layer" because this application can be an "applicative" engine, either software or hardware, but also a "higher" protocol sitting in between the "application" and the SpW protocol stack). However, it is also stated in ECSS-E-ST-50-12C that "nodes" are "SpW interfaces", which is not consistent with their definition. This clearly shows the necessity to introduce a new term to describe the equivalent of "SpW interface" (which is defined only for the SpW Link Level and downwards) at the level of the SpaceWire Service Access Points (i.e. at SpW Network Level). The proposed term for ECSS-E-ST-50-12C Rev.1 is "end-point", a node containing a "higher layer" driving one or more end-points. This allows keeping the ECSS-E-ST-50-12C definition for "nodes": they are source and destination of SpW packets; and they include one or more end-points. the definition of unit can also remain, though somewhat clarified: box, board or subsystem, containing one or more nodes and zero or more switches that may be interconnected through a subnetwork, and that exposes one or more end-points.

End-points, nodes, and units are illustrated in Figures 8 and 9.

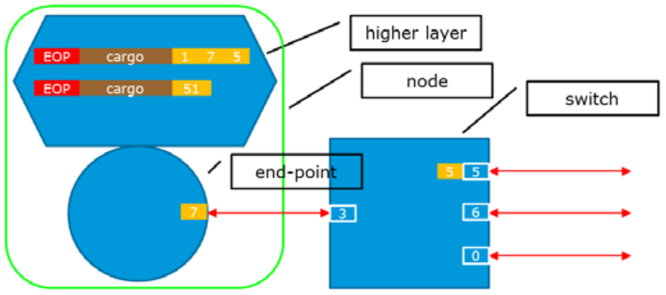


Fig. 8. End-point, node, and switch

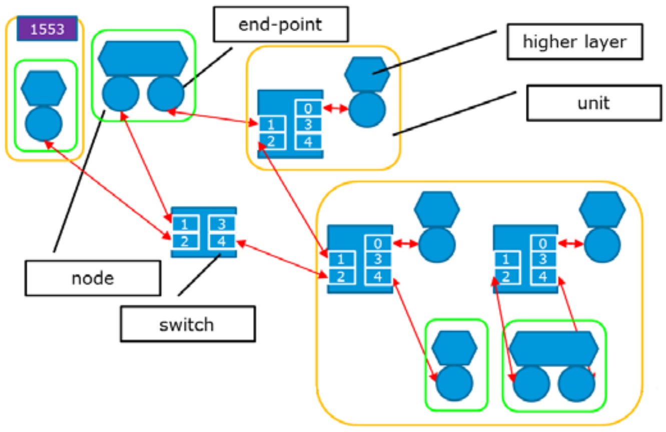


Fig. 9. End-points, nodes, and units

3) Graph representation: Besides consistency and improved clarity of SpaceWire concepts, the main benefit of the introduction of "end-points", as well as of the clarification of the definitions of "nodes" and "units" as objects beyond the scope of SpaceWire per say, is the possibility to represent any set of units connected into a SpaceWire network as a graph containing only leaves (end-points), vertices (switches) and links (see Figure 10). This opens the door to the application to SpaceWire networks of a considerable amount of theoretical results and tools coming from the Graph Theory community, which will ease significantly tackling the main challenge that users of SpaceWire networks are facing, i.e. traffic analysis.

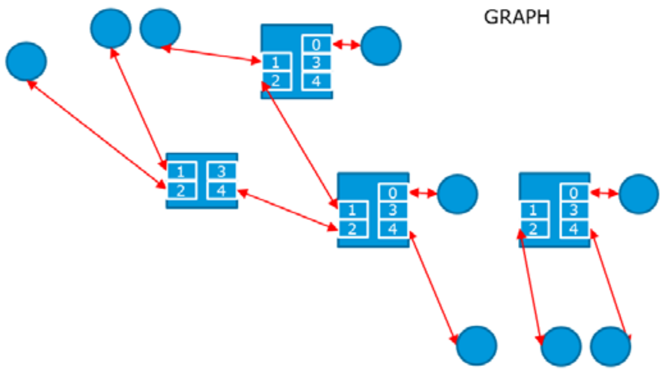
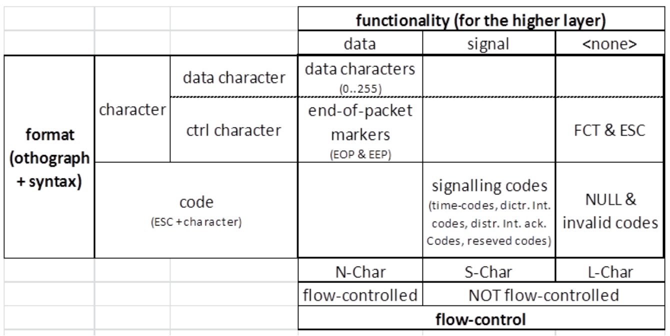


Fig. 10. Any SpaceWire network can be represented as a graph

Additionally, this allows representing any SpaceWire network as an XML formatted text file, as JAXA/ISAS already attempted for their ASTRO-H mission [7].

E. Terminology – clarification of concepts at SpW Link Level

A few ambiguities had to be corrected also in the Character Level of ECSS-E-ST-50-12C, one of them being the status of Time-codes. These control codes are sometimes presented as Link Characters (L-Char) because they are not flow-controlled; but the time-code value is presented to the SpaceWire user interface, which make them Normal Characters (N-Char). As shown in Figure 11, we therefore propose to introduce a third class of characters for Signalling codes (Time-codes and Distributed Interrupts), which are exposed to the SpW user interface but are not flow-controlled: Signalling Characters (S-Char).



			functionality (for the higher layer)		
			data	signal	<none>
format (othograph + syntax)	character	data character	data characters (0..255)		
		ctrl character	end-of-packet markers (EOP & EEP)		FCT & ESC
	code (ESC + character)			signalling codes (time-codes, dictr. Int. codes, distr. Int. ack. Codes, reseved codes)	NULL & invalid codes
			N-Char	S-Char	L-Char
			flow-controlled	NOT flow-controlled	
			flow-control		

Fig. 11. Properties of all characters and codes for the SpW Link Level

F. Terminology – consistent set of definitions for all levels

In order to ensure the consistency between definitions that is lacking in ECSS-E-ST-50-12C, some effort was put into representing the relationship between SpaceWire terms unambiguously using the UML formalism. This effort is still on-going but Figure 12 shows some example of relationships between the notions of "port", "end-point", "switch", "node", "unit", "link-configurable switch", etc. as explained in section III.D; and Figure 13 shows that the effort is targeted to a representation of terms along a matrix structure covering the whole set of SpW internal protocol levels, both in terms of protocol objects and of implementation objects.

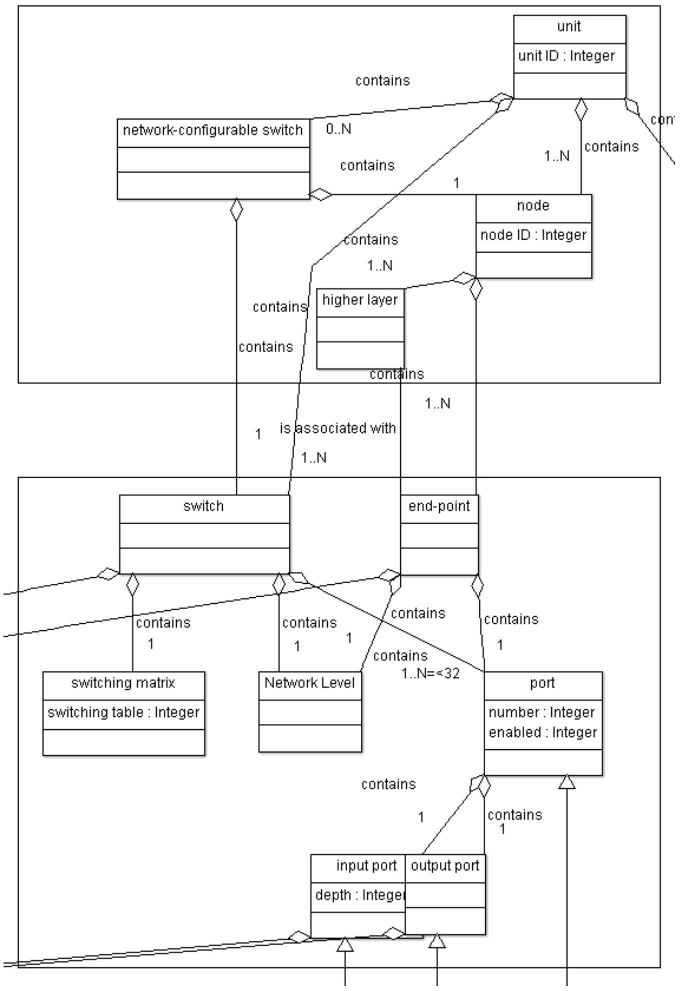


Fig. 12. Detail of UML model showing relationships between terms

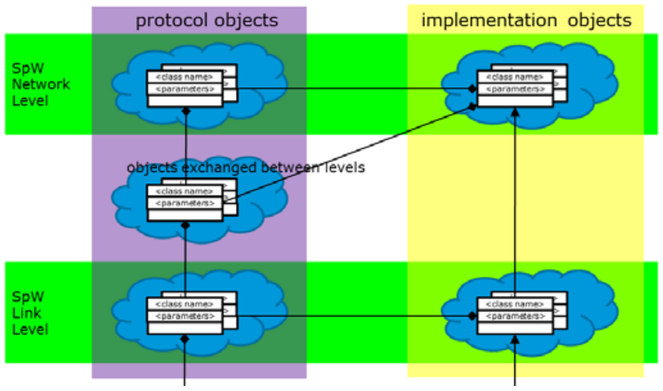


Fig. 13. Matrix structure of SpW term UML representation

IV. Clarification of the Support to SpW-based Protocols

One of the major needs for clarification of the SpaceWire concepts and terms is the impossibility to design a proper management system (protocol + service) for SpaceWire due to the ambiguities in ECSS-E-ST-50-12C.

A. Support to SpW-based Protocols

In particular, the possibility for a higher layer to receive packets with zero as first character and to distinguish these packets (management packets) from the others ("application" packets) needs to be clarified in ECSS-E-ST-50-12C Rev.1. We propose to address this issue by clarifying the optional status of clause 10.5.4.3.a: "If a packet arrives at a node with an unexpected destination address then that packet shall be discarded." and introduce the possibility to either reject or pass to the higher layer a packet based on the value of its header character that can be any from 0 to 255; and to have the option of having this first character deleted before being passed to the higher layer.

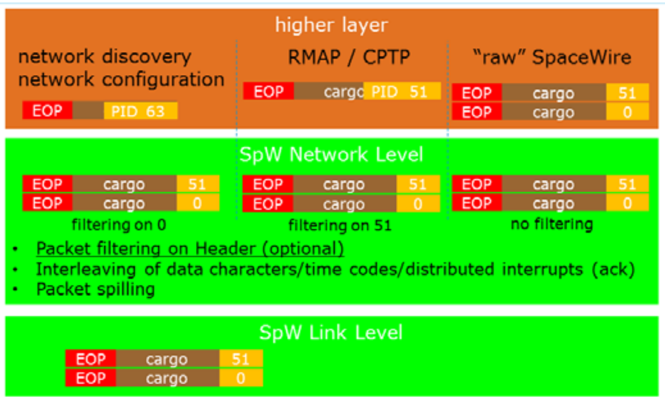


Fig. 14. Packet filtering on Header allows supporting "application" and "management" packets

As illustrated in Figure 14, this feature would allow, when two packets arrive at an end-point, one starting with e.g. data character 51, and the other one with data character 0, to support

- "raw" SpaceWire mode: both packets are passed to the higher layer because the header data character is just the first byte of the SpW cargo
- PID-based "application" protocol (in this example being configured as Destination Address 51): only the packet with expected Destination Address 51 is passed to the higher layer
- management protocol: only the packet with expected leading zero is passed to the higher layer with the zero deleted so that the higher layer can be a standard PID-based protocol (but running in the management space of the node, not its application space).

B. Identification of management parameters

Another area where clarification of the SpaceWire specification (ECSS-E-ST-50-12C Rev.1) and consolidation of the support to management protocols go hand-in-hand is the identification of management parameters. Some parameters such as "link speed" address clearly lower SpW internal protocols levels (in this case the SpW Signal Level) and have therefore little impact on the clarification of SpaceWire concepts. We still propose for ECSS-E-ST-50-12C Rev.1 to clearly identify them as management parameters for the sake of defining management protocols later on. This is also the case of the switching matrix which is a management parameter of a SpW switch (link-configurable or not).

Other features like the ON/OFF state of a link do not appear clearly as management parameters but they are: whether a link is ON/enabled or OFF/disabled at SpW Network level obviously has an impact on the system (a packet to be switched through this link will be spilled) but the SpaceWire user ("higher layer") has no control on the status of this link. It is also clear from the discussions of the ECSS Working Group on the revision of ECSS-E-ST-50-12C that, at least for the sake of backwards compatibility of new features like the Distributed Interrupts, it must be possible to enable an output port for data Packets but not for Time-codes, or for Packets and Time-codes but not for Distributed Interrupts, etc. This shows not only that a SpaceWire output port is in fact a set of four output ports (one for Data Characters, one for Time-codes, one for Distributed Interrupts, and one for Distributed Interrupt Acknowledgements), but also that these character-specific output ports can be enabled one by one. They are therefore management parameters and identifying them clearly as such will allow clarification of the description of the behaviour of the SpW Network Level in ECSS-E-ST-50-12C Rev.1.

The effort of identification of management parameters for each level of the internal SpaceWire protocol stack is on-going. Therefore the complete list to be specified in ECSS-E-ST-50-12C Rev.1 cannot be presented here.

C. "Application protocols" and management protocols

Once we have

- a list of management parameters for each level of the internal SpaceWire protocol stack
- a clear description of any SpaceWire network as a graph linking end-points and switches
- a clear description of a node as hosting a higher layer and one or more end-points
- the possibility to filter packets at SpW Network Level according to the value of the leading byte (with possibility to delete this byte, e.g. if it is zero)

we can clarify the last ambiguity in the concept of "node" that was introduced when the possibility to configure nodes and switches was presented and that ECSS-E-ST-50-12C cannot help clarifying (see section II.A). This ambiguity can be expressed in the two following questions:

a) If only a node can host the configuration space, does a configurable switch contain a node?

b) If only a node can host the configuration space of a switch, is it the same with the configuration space of a node, and does a node contain a node?

Given all what we have presented in this paper, the answer to the first question is "Yes and No": The first clarification is that a switch as such can be configurable but no necessarily through one of its SpW ports. This the case if it contains no node which is the only "source and destination of packets". So "No", switches per-say do not contain nodes. However, a "link-manageable switching unit" will have to include a switch as well as a node connected to one of its ports (e.g. port 0, for the sake of standardisation) that will receive and send the management packets.

For the same reasons, the answer to the second question is clearly "No": Thanks to the segregation between "application" packets and management packets based on the value of the leading byte being zero or not, a node can host in its higher layer an "application" space and a management space. In the case of a link-manageable node, the management space of this node manages the node itself, i.e. the SpW protocol stack running on each of its end-points. This is illustrated in Figure 15. In the case of a link-manageable switching unit, the "application" space of the "management node" (the node receiving and sending switch management packets) is in charge of managing (e.g. configuring) the switch (e.g. the switching matrix, but also the SpW protocol stack running on each of the ports of the switch). This is illustrated in Figure 16.

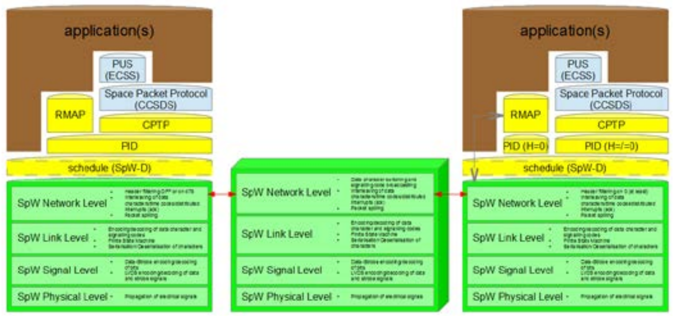


Fig. 15. "Application" space and management space in a node (right node)

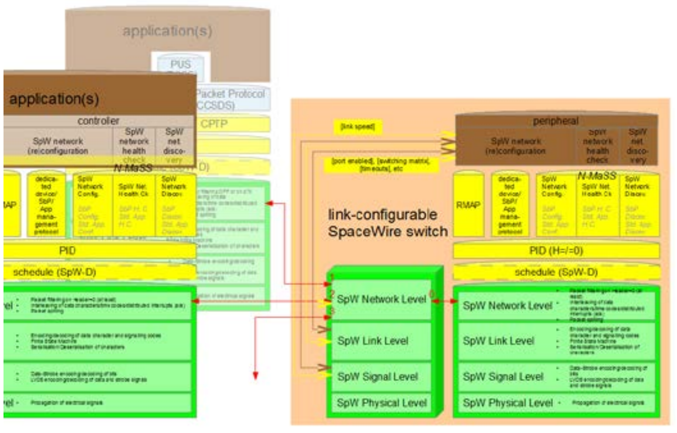


Fig. 16. Link-manageable switching unit: "Application" space of a node managing a switch

V. Conclusion

Through thorough analysis of the SpaceWire protocol stack, both in the perspective of the diversification of the lower protocol levels and in the perspective of the support to management protocols and services, a lot of progress has been achieved in the understanding of the SpaceWire concepts and terms and the way this communication protocol should be more clearly specified. The areas of ambiguity and unknown as shown in Figure 17 are now much clearer, as shown in Figure 18.

Part of the clarifications proposed in this paper still need to be detailed; and the Signal and Physical still need to be addressed. But the re-writing of the SpaceWire specification in to a much clearer ECSS-E-ST-50-12C Rev.1 is on the right track, including smooth introduction of new (but backwards compatible) features such as sideband signalling for interrupt distribution.

Once ECSS-E-ST-50-12C Rev.1 is ready, standardisation of additional SpaceWire-based protocols (SpW-D for time determinism, time synchronisation protocol, network discovery and management protocol, etc.) will be a fairly process because these protocols will be supported by a clear and consistent set of concepts.

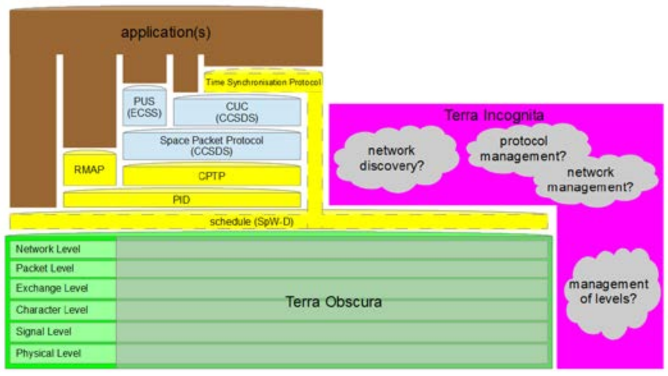


Fig. 17. Map of the SpaceWire world according to ECSS-E-ST-50-12C, ECSS-E-ST-50-51C, ECSS-E-ST-50-52C, and ECSS-E-ST-53-12C

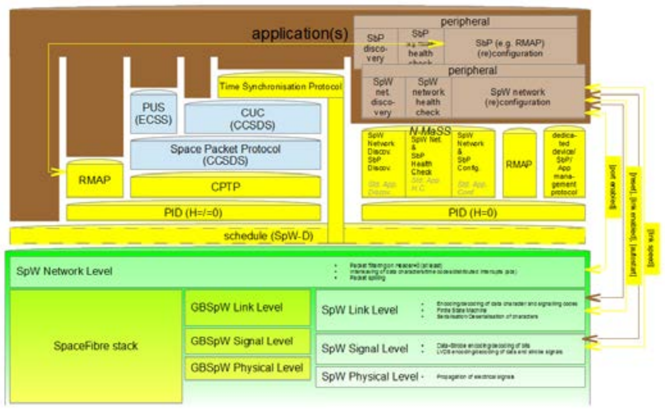


Fig. 18. Possible map of the SpaceWire world based on ECSS-E-ST-50-12C Rev.1

ACKNOWLEDGEMENT

We would like to thank Peter Mendham for his significant contribution to the clarifications proposed in this paper, as part of his work on SpaceWire network management protocols and services.

Most of the clarifications proposed in this paper have been discussed extensively and refined within the frame of the ECSS Working Group on the revision of ECSS-E-ST-50-12C (Ahmed Bouabdallah, Stephen Bury, Barry Cook, Brice Dellandrea, Sandi Habinc, Torbjörn Hult, Marko Isomaki, Stefan Jörg, Jérôme Lachaize, Mathias Nickl, Olivier Notebaert, Steve Parkes, Roger Peel, Paul Rastetter). Some of these clarifications were even born in that context and we would like to thank all the members of this working group for their fruitful contributions.

REFERENCES

[1] ECSS-E-ST-50-12C, "SpaceWire - Links, nodes, routers", 31 July 2008

[2] ECSS-E-ST-50-51C, "SpaceWire protocol identification", 5 February 2010

[3] ECSS-E-ST-50-52C, "SpaceWire - Remote memory access protocol", 5 February 2010

[4] ECSS-E-ST-50-53C, "SpaceWire - CCSDS packet transfer protocol", 5 February 2010

[5] Evgeny Yablokov, "GigaSpaceWire – Gigabit Links for SpaceWire Networks", International SpaceWire Conference, Gothenburg, June 2013

[6] Yuriy Sheynin, "DC-balanced SpaceWire links with galvanic isolation, longer distances and gigabit rates, Yuriy Sheynin, SUAI", 19th SpaceWire Working Group, ESTEC, April 2013

[7] Takayuki Yuasa, "Design guideline and software tools for deterministic SpaceWire network using SpaceWire-D", 19th SpaceWire Working Group, ESTEC, April 2013

[8] Barry Cook, "Low Latency Packet Delivery in SpaceWire Networks", International SpaceWire Conference, San Antonio, November 2011

[9] Yuriy Sheynin, "Next release of the SpaceWire standard - some requests for change", 14th SpaceWire Working Group, ESTEC, February 2010

[10] 4Links Ltd, White paper on Virtual SpaceWire Networks, September 2009

[11] Peter Mendham, SpaceWire-PnP Protocol Definition, Draft A Issue 2.1, 16th September 2009

[12] Barry Cook, "Half Duplex SpW", 13th SpaceWire Working Group, ESTEC, September 2009

[13] Yuriy Sheynin, "SpaceWire evolution", 10th SpaceWire Working Group, ESTEC, February 2009

[14] Barry Cook, "Half-Duplex SpaceWire", 10th SpaceWire Working Group, ESTEC, February 2009

[15] Prof. Yuriy Sheynin, "Distributed Interrupts in SpaceWire Interconnections", International SpaceWire Conference, Nara, November 2008

[16] Liudmila Onishchenko, Artur Eganyan, Irina Lavrovskaya, "Distributed Interrupts Mechanism Verification and Investigation by Modeling on SDL and SystemC", International SpaceWire Conference, Nara, November 2008

[17] Barry Cook, Paul Walker, "Half-duplex SpaceWire: Reducing harness mass while retaining full compatibility with SpaceWire's modularity, configurability and adaptability", IAC-08, September 2008

[18] David Jameux, "SpaceWire for Command & Control", 10th SpaceWire Working Group, ESTEC, February 2008

[19] David Jameux, Albert Florit Ferrer, "Towards the definition of Quality of Service classes for SpaceWire-based message passing", October 2007

[20] Eugenue Yablokov, "Simplex Mode in SpW Technology", International SpaceWire Conference, Dundee, September 2007

[21] ESA & NASA, "ESA and NASA requirements on SpaceWire PnP", March 2007,

[22] Barry Cook, Paul Walker, "PnP aspects, 4Links contribution", 8th SpaceWire Working Group, ESTEC, January 2007

[23] Albert Ferrer Florit, "PnP aspects, ESA contribution", 8th SpaceWire Working Group, ESTEC, January 2007

[24] Prof. Yuriy Sheynin, "Distributed Interrupts in SpaceWire Networks", December 2006

[25] Steve Parkes, "The Operation and Uses of the SpaceWire Time-Code", International SpaceWire Seminar, ESTEC, 2003

Developing SpaceWire Devices with STAR-Dundee Test and Development Equipment

Poster Paper

Stuart Mills, Alex Mason, Chris McClements, David Paterson, Iain Martin
STAR-Dundee Ltd.
Dundee, Scotland, UK
stuart.mills@star-dundee.com, alex.mason@star-dundee.com, chris.mcclements@star-dundee.com, david.paterson@star-dundee.com, iain.martin@star-dundee.com

Steve Parkes
School of Computing
University of Dundee
Dundee, Scotland, UK
sparkes@computing.dundee.ac.uk

***Abstract*—STAR-Dundee has recently released a new range of interface and router devices to support SpaceWire-related test and development. Each of these devices is provided with a comprehensive software suite, which further simplifies the test and development stages. This paper explores the many benefits of using these devices to develop a new SpaceWire device, protocol or application. Typical scenarios in each stage of development are considered, and information on how the software and devices can be used in these situations is provided.**

***Index Terms*—SpaceWire, Networking, STAR-Dundee, Spacecraft Test and Development Equipment, STAR-System, USB, PCI, cPCI, PCIe, Brick Mk2, Router Mk2S, EGSE, Link Analyser Mk2, Conformance Tester, SpaceWire Physical Layer Tester.**

I. Introduction

STAR-Dundee has recently introduced a number of new products to the SpaceWire test and development market, including a range of router and interface devices supported by a comprehensive software suite.

This paper introduces the various products available from STAR-Dundee to assist in test and development of SpaceWire devices, protocols and applications, concentrating primarily on the products recently added to the range. The paper's aim is to demonstrate how these products can be used at each stage of development not only to make that stage easier to complete, but also with better results than previously possible.

II. Learning To Use SpaceWire

For over eight years, STAR-Dundee's SpaceWire-USB Brick has been many people's first introduction to SpaceWire. The device provides two SpaceWire ports and a connection to a PC via a USB cable, with power to the device provided by USB.

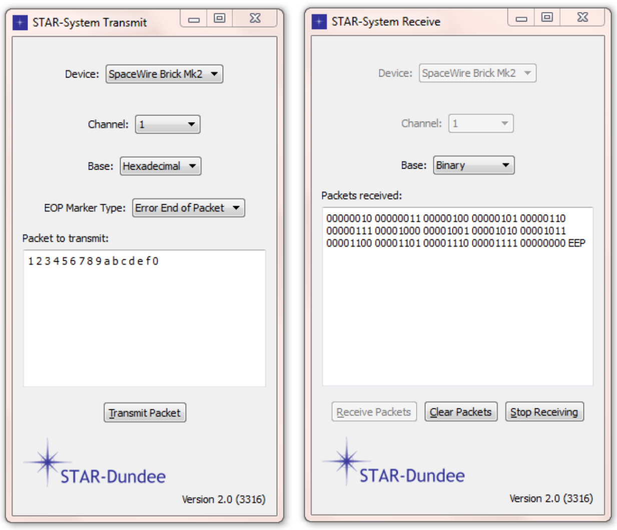
The Brick has now been replaced by a new device, the SpaceWire-USB Brick Mk2 [1], shown in Figure 1, which has all the functionality of the original, plus a lot more. The Brick Mk2 is supplied with STAR-System, STAR-Dundee's software suite which includes drivers, APIs, documentation, test and example programs and powerful graphical applications.

Figure 1 SpaceWire-USB Brick Mk2

Using the Brick Mk2 is very easy, which makes it an ideal device when getting started with SpaceWire. To get up and running, install the STAR-System software on a Windows or Linux PC and connect a USB cable between the PC and the Brick Mk2. The STAR-System applications are available from the Start menu or equivalent in Linux.

As a first stage of transmitting and receiving packets over SpaceWire, a SpaceWire Lab Cable (also available from STAR-Dundee) can be connected between the two SpaceWire ports in a loopback configuration. The STAR-System Transmit application shown on the left in Figure 2 can then be used to transmit packets. The packets are typed in to the text box, the end of packet marker type selected, and the Transmit Packet button pressed to transmit the packet.

The STAR-System Receive application, shown on the right in Figure 2, can be used to receive the packets transmitted. Clicking the Receive Packets button will result in the contents of all received packets being displayed, along with each packet's end of packet marker. The base used when entering and displaying packets can be specified, and in this example the packets were entered in hexadecimal and the received packets are displayed in binary.


STAR-System Transmit

Device: SpaceWire Brick Mk2

Channel: 1

Base: Hexadecimal

EOP Marker Type: Error End of Packet

Packet to transmit:

```
1 2 3 4 5 6 7 8 9 a b c d e f 0
```

STAR-System Receive

Device: SpaceWire Brick Mk2

Channel: 1

Base: Binary

Packets received:

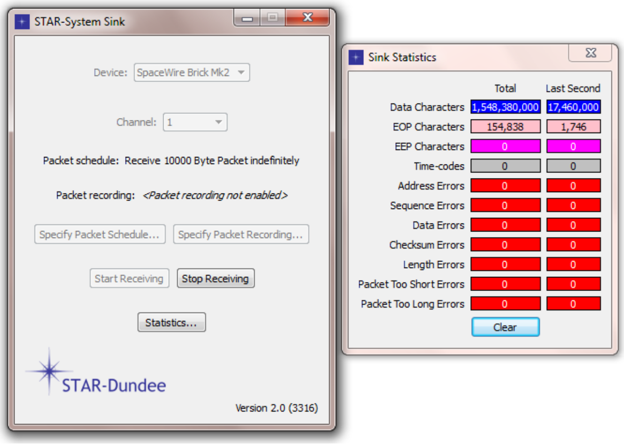
```
00000010 00000011 00000100 00000101 00000110
00000111 00001000 00001001 00001010 00001011
00001100 00001101 00001110 00001111 00000000 EEP
```

Figure 2 STAR-System Transmit and Receive Applications

III. Interfacing and Routing

The Brick Mk2 device is an interface device, and can be used (along with STAR-Dundee's other interface devices) to transmit packets over a SpaceWire network, and receive packets from the network. This functionality can be used during device development for a number of purposes: transmitting commands to the device, receiving data from the device, etc.

In addition to the basic STAR-System Transmit and Receive applications, more powerful applications are also provided. The STAR-System Source application allows complex packet formats to be constructed and transmitted at high speeds, with transmit statistics displayed. The STAR-System Sink application shown in Figure 4 can receive packets at high speeds, and can optionally write these received packets to file. The received packets can be checked for errors, using the same packet formatting information as is used in the Source application. Statistics are available, including the number of errors in the received packets being checked. In the figure, the


STAR-System Sink

Device: SpaceWire Brick Mk2

Channel: 1

Packet schedule: Receive 10000 Byte Packet indefinitely

Packet recording: <Packet recording not enabled>

Sink Statistics

	Total	Last Second
Data Characters	1,548,380,000	17,460,000
EOP Characters	154,838	1,746
EEP Characters	0	0
Time-codes	0	0
Address Errors	0	0
Sequence Errors	0	0
Data Errors	0	0
Checksum Errors	0	0
Length Errors	0	0
Packet Too Short Errors	0	0
Packet Too Long Errors	0	0

Figure 4 STAR-System Sink Application

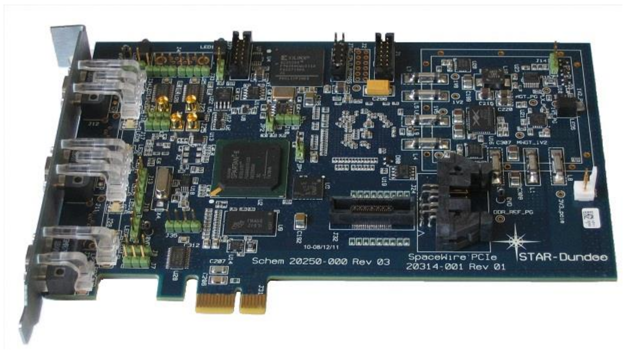


Figure 3 SpaceWire PCIe

application is indefinitely receiving a previously defined packet format (named "10000 Byte Packet") and checking the received packets for correctness. The statistics indicate that no error has been detected so far, and that data is being received at approximately 17 Mbytes/s.

The other STAR-Dundee interface devices are the SpaceWire PCI Mk2 [2], the SpaceWire cPCI Mk2 [3] and the SpaceWire PCIe [4], shown in Figure 3. All are provided with the same STAR-System software suite, have three SpaceWire ports, and each can be accessed using the same STAR-System APIs. The functionality provided by each unit is similar – the main differences are in their bus interfaces. A USB device can be easily connected and removed from a PC, and there are multiple USB ports on most modern PCs. The PCI bus provides better throughput and latency than the USB bus, but devices must be fitted in the PC when it is switched off. cPCI devices are similar to PCI devices, but can be fitted in a rack. Finally, the PCIe bus offers better throughput than the PCI bus.

All of STAR-Dundee's interface devices also provide a routing mode, which allows SpaceWire routing to be investigated. For a standalone router, the recently updated SpaceWire Router Mk2S [5] is a router with eight SpaceWire ports. It is functionally equivalent to the STAR-Dundee Router IP and ESA SpW-10X Router ASIC, the Atmel AT791. It also has a USB port, so can optionally be connected to a PC and used with the same STAR-System software as the other interface and router devices.

In addition to transmitting and receiving packets, the interface and router devices can also be used to configure other devices on the network. The STAR-System Device Configuration application (see Figure 5) allows supported devices to be configured either locally or over a SpaceWire network. This makes it easy to set timeouts, change link speeds and get the error status of devices on the network. Routing tables of routing devices can also be configured, so the routing table of a Router Mk2S can be configured using a PCI Mk2, for example, over the SpaceWire network.

IV. Simulating Instruments

Simulation of instruments and other devices can be performed using the interface and router devices previously mentioned. For example, the STAR-System Source application can be used to transmit packets containing images in order to simulate a camera. Other fields that can be included in the transmitted packets include sequence numbers, CRCs, checksums, field lengths, data patterns and address and data bytes. This makes it possible to transmit packets in the format used by existing protocols or to test out new protocols. After creating the required packet format, these are stored to hard disk for future use.

The STAR-System Sink application can be used to receive the images, and simulate a receiving device such as a mass memory unit. The received packets can be written to file for further analysis, and their format can be checked for errors using the same packet format information used by the Transmit application.

For more complex simulations, the STAR-System APIs can be used to develop applications to perform specific tasks. The main API provides functions which make it simple to write applications to transmit and receive packets at high rates and with low latency. Additional APIs are provided for device configuration and for building and interpreting RMAP packets. Example code is provided, and linked to in the comprehensive documentation, to demonstrate how each function can be used.

For development of graphical device simulations, the STAR-System LabVIEW API [6] allows applications to be developed using National Instrument's LabVIEW development environment. The STAR-System LabVIEW wrapper has been designed to be intuitive to LabVIEW users, and includes a number of example applications. These cover common SpaceWire development tasks, and so can be dropped in to a LabVIEW application, greatly simplifying development of that application.

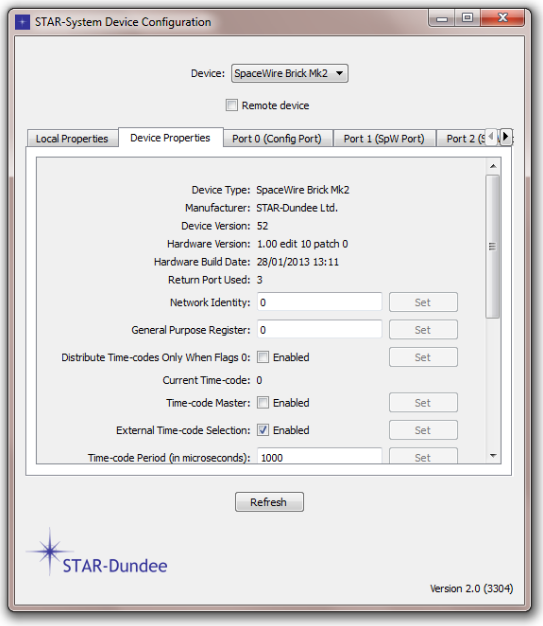
If deterministic behaviour is required in the device simulation, STAR-System releases are available for both VxWorks [7] and QNX [8] real-time operating systems. These releases currently work with the SpaceWire PCI Mk2, cPCI Mk2 and PCIe and provide exactly the same API as on Windows and Linux. This means that software can be developed and tested on a Windows or Linux desktop machine, before being recompiled for the final target.

An alternative device, the SpaceWire EGSE [9] [10], can be used to perform real-time device simulation entirely in hardware, and so provide deterministic behaviour. The EGSE is programmed using a simple yet powerful scripting language using a connected PC, and then the script is executed on the device without further interaction with the hosting PC. The PC can be notified of events, if required, and can also send software triggers. The EGSE's trigger output ports can be used to notify external equipment when a specified event occurs, while the trigger input ports can be used to alter the state of the script based on an external event. For example, an external clock could be used to indicate that a packet should be sent at regular intervals.

V. Developing For Flight

The devices discussed above can be used to test and develop new flight equipment. To work with existing flight-rated devices, STAR-Dundee also offers a number of solutions.

While the SpaceWire Router Mk2S is functionally equivalent to the ESA SpW-10X, the STAR-Dundee SpW-10X



STAR-System Device Configuration

Device: SpaceWire Brick Mk2

☐ Remote device

Local Properties | Device Properties | Port 0 (Config Port) | Port 1 (SpW Port) | Port 2 (S

Device Type: SpaceWire Brick Mk2
Manufacturer: STAR-Dundee Ltd.
Device Version: 52
Hardware Version: 1.00 edit 10 patch 0
Hardware Build Date: 28/01/2013 13:11
Return Port Used: 3
Network Identity: 0 [Set]
General Purpose Register: 0 [Set]
Distribute Time-codes Only When Flags 0: ☐ Enabled [Set]
Current Time-code: 0
Time-code Master: ☐ Enabled [Set]
External Time-code Selection: ☑ Enabled [Set]
Time-code Period (in microseconds): 1000 [Set]

Refresh

STAR-Dundee

Version 2.0 (3304)

Figure 5 STAR-System Device Configuration Application

Router ASIC (AT7910E) Evaluation Kit [11] includes an actual SpW-10X ASIC in a rack-mountable unit which simplifies interfacing with this device during evaluation and development. The SpW-10X, and any other devices which make use of STAR-Dundee's Router IP, can be configured using the STAR-System Device Configuration application shown in Figure 5. In the figure, the properties of a Brick Mk2 (which is functionally similar to a SpW-10X) are currently being configured. From this application, the properties of each port can be configured, the error status viewed, and the routing table set.

Another product which provides support for flight-rated devices is the SpaceWire RTC (AT7913E) Development Kit. It has external connectors exposing the various interfaces of the Atmel AT7913E device, the SpaceWire Remote Terminal Controller (RTC). The STAR-Dundee SPARCv8 Software Development Environment (or STAR SDE) is provided with the RTC Development Kit, to assist software development, debugging and testing on the RTC's SPARCv8 chip. It can also be used with other flight-rated SPARCv8 processors including the Atmel AT697E/F and AT697FF devices.

While debugging, the SDE includes graphical views of the internal registers on the device, the trace buffer and the processor cache. Source code, assembly language and graphical representations of the code using Code Rocket are available while stepping through the code. Debugging can be performed interactively over USB or UART.

When developing a new flight device, STAR-Dundee offers a range of SpaceWire IP cores provided as VHDL source code, including a CODEC, a Router [14], an RMAP Initiator and an RMAP Target [15]. These cores have been extensively tested and proven, are used in STAR-Dundee's test and development equipment and are also incorporated in the Atmel AT7910E (SpW-10X) and AT7913E (SpaceWire RTC) ASICs.

VI. Testing and Debugging

The interface and router devices described earlier all provide numerous features to assist in testing and debugging. Using the STAR-System software or APIs, they can of course be used to transmit traffic to a device under test, or receive traffic from the device under test. They can also be used to transmit and receive time-codes, and errors detected on the link are latched so that any unexpected behaviour can be detected. All of these devices also offer mechanisms to introduce errors in to traffic including credit errors, escape errors and parity errors, meaning that it's possible to put together a full test suite, which can be controlled through software, to test a device under development.

One of the most important tools to have when debugging a problem in a SpaceWire network or device is the SpaceWire Link Analyser Mk2 (see Figure 6) [16]. The Link Analyser Mk2 can be used to record the traffic crossing a link, and can trigger on events such as errors, time-codes, specific data characters, etc. The traffic recorded can be viewed as a link level trace (individual n-chars and l-chars), a packet level trace, or a signal level trace (Data and Strobe waveforms). The packet level trace can also show protocol information, such as the fields of an RMAP packet. The Link Analyser Mk2's accurate statistics also make it simple to confirm the throughput performance of a device is as expected, while error injection is provided to monitor how a device performs when errors are introduced.

When developing a new device, it's important to test the device at all layers of the SpaceWire standard. The STAR-Dundee SpaceWire Conformance Tester [17] tests conformance of a device to numerous clauses in the SpaceWire standard, concentrating primarily above the physical layer.

The SpaceWire Physical Layer Tester [18] builds on this by offering testing of the device at the physical layer. It can apply a variety of different aberrations including offset, drive strength, slew, skew and jitter to the electrical SpaceWire LVDS signals. A powerful software suite makes it very easy to control the amount of signal degradation that the unit under test can cope with by progressively degrading combinations of these aberrations until the connection is broken. The SPLT also features analogue buffers on its termination resistors to allow the signal received from the device under test to be buffered and viewed on a scope.

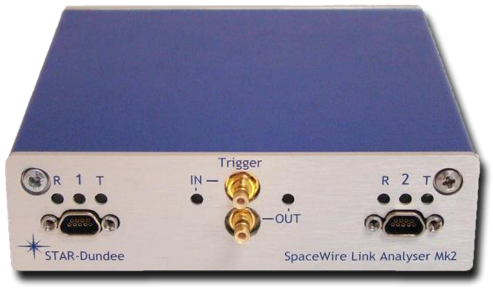


Figure 6 SpaceWire Link Analyser Mk2

VII. Summary

This paper has introduced a number of products to assist in the development and testing of SpaceWire devices, applications and protocols. These devices are used by STAR-Dundee when developing our own products, so include not only features requested by customers, but also features required by STAR-Dundee engineers. Our products are always being improved as we receive requests from customers, or encounter a situation where an existing solution does not exist, and we always welcome comments and suggestions.

References

[1] STAR-Dundee, "SpaceWire-USB Brick Mk2", http://www.star-dundee.com/products/spacewire-usb-brick-mk2, 2013.

[2] STAR-Dundee, "SpaceWire PCI Mk2", http://www.star-dundee.com/products/spacewire-pci-mk2, 2013.

[3] STAR-Dundee, "SpaceWire cPCI Mk2", http://www.star-dundee.com/products/spacewire-cpci-mk2, 2013.

[4] STAR-Dundee, "SpaceWire PCIe", http://www.star-dundee.com/products/spacewire-pcie, 2013.

[5] STAR-Dundee, "SpaceWire Router Mk2S", http://www.star-dundee.com/products/spacewire-router-mk2s, 2013.

[6] STAR-Dundee, "SpaceWire LabVIEW Driver", http://www.star-dundee.com/products/spacewire-labview-driver, 2013.

[7] STAR-Dundee, "SpaceWire VxWorks Driver", http://www.star-dundee.com/products/spacewire-vxworks-driver, 2013.

[8] STAR-Dundee, "SpaceWire QNX Driver", http://www.star-dundee.com/products/spacewire-qnx-driver-0, 2013.

[9] STAR-Dundee, "SpaceWire EGSE", http://www.star-dundee.com/products/spacewire-egse, 2013.

[10] S. Mudie, M. Dunstan, S. Parkes; "SpaceWire EGSE: Real-Time Instrument Simulation in a Day", International SpaceWire Conference 2013, June 2013, Sweden.

[11] STAR-Dundee, "SpW-10X Router ASIC (AT7910E) Evaluation Kit", http://www.star-dundee.com/products/spw-10x-router-asic-at7910e-evaluation-kit, 2013.

[12] STAR-Dundee, "SpaceWire RTC (AT7913E) Development Kit", http://www.star-dundee.com/products/spacewire-rtc-at7913e-development-kit, 2013.

[13] STAR-Dundee, "SPARCv8 Software Development Environment", http://star-dundee.com/products/sparcv8-software-development-environment, 2013.

[14] STAR-Dundee, "SpaceWire IP Cores", http://www.star-dundee.com/products/spacewire-ip-cores, 2013.

[15] STAR-Dundee, "SpaceWire RMAP IP Cores", http://www.star-dundee.com/products/spacewire-rmap-ip-cores, 2013.

[16] STAR-Dundee, "SpaceWire Link Analyser Mk2", http://www.star-dundee.com/products/spacewire-link-analyser-mk2, 2013.

[17] STAR-Dundee, "SpaceWire Conformance Tester", http://www.star-dundee.com/products/spacewire-conformance-tester, 2013.

[18] A. Spark, P. Scott, P. Crawford, S. Parkes; "Margin Testing of SpaceWire Devices", International SpaceWire Conference 2013, June 2013, Sweden.

Thursday 13 June

Standardisation (Short)

Towards High-Speed SpaceWire Links

SpaceWire Standardisation, Short Paper

Mathias Nickl, Stefan Jörg, Thomas Bahls,
Robotics and Mechatronics Center
German Aerospace Center (DLR)
Oberpfaffenhofen, Germany
mathias.nickl@dlr.de

Barry M. Cook
4Links Limited
Milton Keynes, England
barry@4links.co.uk

***Abstract*—Various applications, such as telecommunication and robotics, ask for communication bandwidth beyond 1Gb/sec. However, SpaceWire is still limited to lower rates. The IEEE1355 standard proposes a high-speed *exchange level*, which is optimized for 8b12b-encoding (HS-SE-10 and HS-FO-10). To enable high-speed communication with SpaceWire, the authors resurrect the IEEE1355-HS-SE concept and adapt it to the requirements for SpaceWire links. Therefore, time-characters are integrated and the encoding is changed to 8b10b-encoding to enable the usage of common physical layer circuits. This paper presents the resulting specification, an exemplary implementation, and a first experimental result.**

***Index Terms*—SpaceWire, IEEE1355, SpaceWire-HS, robotics**

I. Introduction

Various applications, such as telecommunication and robotics, still ask for the advantages of SpaceWire, i.e. a simple packet protocol with small footprint. However, due to rising complexity, modern systems require communication bandwidth beyond 1 Gbit/s per link, which is not supported by ECSS-E-ST-50-12C.

IEEE1355-1995, an earlier version of what we call SpaceWire today, proposes high-speed links named HS-SE-10 for single ended copper and HS-FO-10 for fiber optic links. But neither one has been transferred to ECSS-E-ST-50-12C.

The SpaceFibre standard proposes communication bandwidth beyond 1 Gbit/s and the interfaces of SpaceFibre links are intended to be compatible to SpaceWire links. However, SpaceFibre links are more complex than SpaceWire links, since reliability aspects are moved from application layer into the links.

Nickl et al. [1] show that the SpaceWire *exchange level* can be combined with common *character level* circuits that run with 8b10b encoding (see [4]). This straight forward approach is not efficient, since bit-stream synchronization is implemented by 8b10b comma characters and is also regarded by the *exchange level* state machine[1].

To enable an efficient high-speed SpaceWire link with small footprint, the authors resurrect the IEEE1355-HS-SE and adapt it to the requirements for SpaceWire links. Therefore,

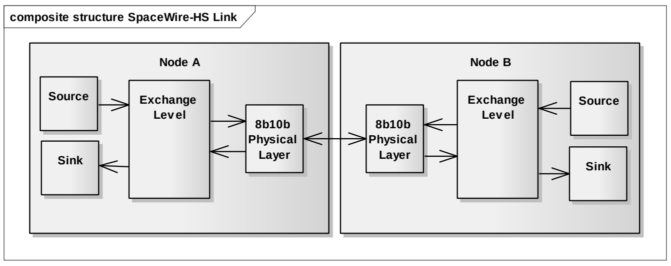


Figure 1: Structure of a SpaceWire-HS node with *exchange level* interfacing a common 8b10b physical layer circuit

time-codes are integrated and the encoding is changed to 8b10b to enable the usage of common physical layer circuits. Hereinafter this concept is referred to as SpaceWire-HS.

This paper presents the protocol specification of SpaceWire-HS in section II, an exemplary implementation in section III, and first experimental result in section IV.

II. Protocol Specification

SpaceWire-HS defines an *exchange layer* that enables SpaceWire communication by using common physical layer circuits with 8b10b encoding as used for Gigabit Ethernet (IEEE 802.3). The protocol is a derivation of the IEEE-1355-HS *exchange level* protocol. Minor modifications such as *time-code* support, changed initialization sequence, and 8b10b instead of 8b12b bring IEEE-1355-HS into line with SpaceWire and 8b10b encoding.

Fig. 2 shows the structure of an *exchange level* implementation with bidirectional interfaces for *time-codes* and *nchars* (left interface) as well as a duplex 8b10b-interface (right interface). The protocol automata (**RxFsm** and **TxFsm**) implement start, stop, flow control and error handling. The **Encoder** converts SpaceWire symbols (i.e. *nchars* and *lchars*) to 8b10b-characters (i.e. K.x.y and D.x.y) and the **Decoder** vice versa. The internal channels are named alphabetically, to avoid ambiguities. **Mux** and **Dispatcher** decouple message routing and protocol implementation. Due to that, the implementation provides four channels, a forward channel C_{fw} = (a, g, h, b), a feedback channel C_{bw}=(c, g, h, d), a channel for *time-codes* C_{tc}=(e, g, h, f), and a handshake channel C_{hs}=(a, g, h, d).

[1] To distinguish different modifications of SpaceWire, this implementation is called 'SpaceWire-1Gb'

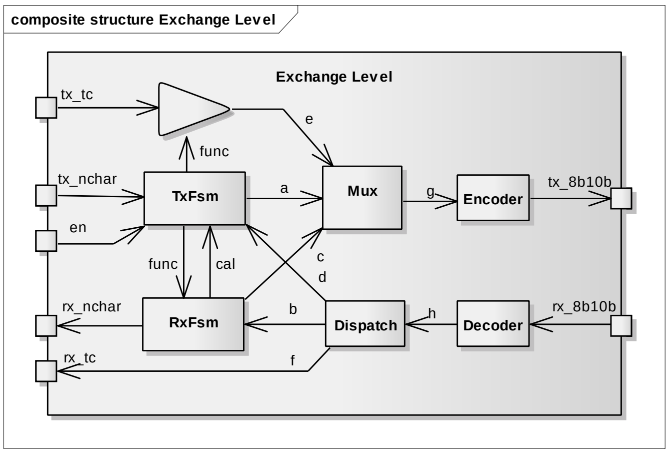


Figure 2: SpaceWire-HS *exchange level*

The grammar in Fig. 6 specifies all symbols, which are carried along the channels shown in Fig. 2 and defines the encoding, i.e. the mapping of SpaceWire-characters to 8b10b-characters.

All symbols are encoded as simple *kchars* (K.*x.y*) *extended kchars* (K.*x.y* followed by D.*x.y*), or data characters D.*x*.y. IDLE is mapped to K.28.0, which has balanced disparity, to reduce power consumption while sending IDLEs. *Time-codes* are mapped to (K.28.1, D.*x.y*), where D.*x.y* carries the *time-code*, i.e. *tc* and *ctrl* (see [3]). START, STOP, and RESET symbols are *escape characters* (K.28.6, D.*x*.0), where *x* is the *identifier* of the *escape character*. INVALID is an extra symbol that covers *character level* errors.

In contrast to the data strobe link (DS), which has a silent initialization phase to synchronize the bit-stream, SpaceWire-HS is optimized for 8B10B character layers, which allow bit stream synchronization by a dedicated comma character. Therefore, the transmitter sends a startup sequence (INIT), which consists of a COMMA followed by IDLEs. The number of IDLEs can be configured, e.g. currently one COMMA followed by 32 IDLEs.

IEEE-1355-1995 (Annex G) proposes to insert an error checking code to get a higher level of fault-sensitivity. This is regarded by an extra byte for cyclic-redundancy-check (CRC) at the end of packet data, previous to the EOP. (An EEP packet has no CRC.)

The *exchange level* protocol is sliced in TxFsm and RxFsm (see Fig. 7). To avoid deadlock, TxFsm has two startup paths, one for *unidirectional* and one for *bidirectional startup*.

In the case of *unidirectional startup* only one (A) of the two connected nodes gets a request to transmit (en='1'). Hence, only A changes to state TX_CAL_1. After A has calibrated its transmitter it changes to TX_WORKING_1 and starts to send INIT characters. The incoming INITs calibrate the receiver of B (cal='1'). Hence, B changes to state TX_CAL_2. After calibrating the transmitter B changes to TX_WORKING_2 and also starts to send the INIT sequence. Thus, A calibrates its receiver and changes to TX_WORKING_3. At this point of time, both nodes have calibrated transmitters and receivers and send the startup handshake, i.e. START_REQ and START_ACK.

In case of *bidirectional startup* both nodes get a request to transmit (en='1'). Therefore, both change from TX_NOT_WORKING to TX_CAL1 and after calibration of the transmitter to TX_WORKING_1. Then A and B send the INIT sequence, which calibrates the peer receiver. After receiver calibration the transmitter sends the startup handshake. Hence, *bidirectional startup* allows sending START_REQs before the opposite node is calibrated [ieee1355]. Therefore, the timeout TO1 triggers resending START_REQs. The global timeout TO2 initiates a reset in case of failure during startup (e.g. disconnect, etc.).

After startup the TxFsm reaches TX_FUNCTIONAL. According to the credit counter, which represent the free space in the peer receiver FIFO, *nchars* can be sent.

The shut-down mechanism also distinguishes unidirectional and bidirectional case. Only if both nodes have requested a shut-down the link shall completely shut-down. As long as the link isn't completely shut down the receivers are still active and able to receive data.

The receiver state machine handles the calibration and implements the credit counter for the incoming *nchars*. Furthermore, it checks the CRC, i.e. converts an EOP to EEP in case of invalid check-sum, and adds an EEP in case of broken link.

III. Implementation and Experimental Results

For testing two circuit boards (see Fig. 3) with Xilinx Virtex5 (5VLX50) and TLK1221 from Texas Instruments, which is an IEEE802.3 Gigabit Ethernet compliant physical layer circuit, are used. The boards are linked by an impedance controlled cable with two crossed differential pairs and high-speed connectors (COMTRONIC’s CMRM, see Fig. 5). The differential serial outputs are PECL-compliant and the serial inputs are AC-coupled. The boards can be coupled by copper or fiber (the latter by applying additional copper-to-fiber adapters). Both FPGAs host a *testNode* which generates stimulation-patterns and observes the behavior of the links. Furthermore, the tested link implementation has additional instrumentation capabilities to trace internal link errors (debug interface).

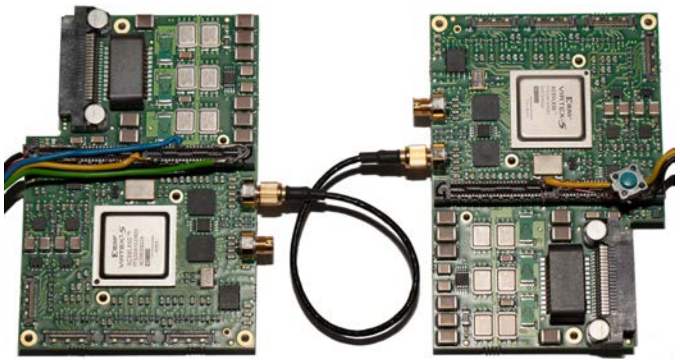


Figure 3: FPGA Testboards for Gigabit Communication with Comtronics CMRM cable

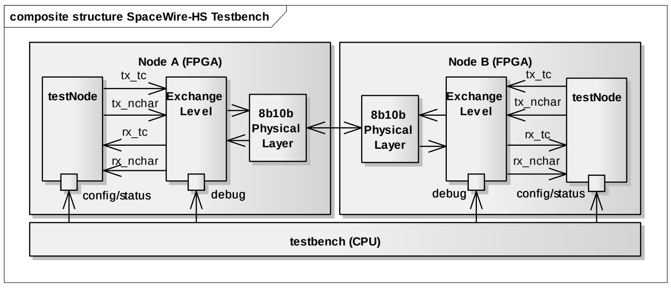


Figure 4: Testbench for SpaceWire-HS link

Transmitter and receiver of the TLK1221 run on different unsynchronized clocks (**clk_tx, clk_rx**). The *testNode* has a third clock (**clk_node**). As a consequence the data flow of the transmitter path (**TxFsm**, **Mux**, **Encoder**) and the receiver path (**RxFsm**, **Dispatcher**, **Decoder**) as well as the application interface has to be decoupled with deterministic rate-transitions. Therefore, the channels **tx_nchar**, **rx_nchar**, **tx_tc**, **rx_tc**, **c**, and **d** (Fig. 2) are implemented as dual-clock FIFOs, **cal** and **func** as synchronizers [5].

The SpaceWire-HS link-level is implemented in VHDL. Table 1 shows the synthesis results of a design consisting of *exchange level* and 8b10b encoding/decoding and compares it to a design of SpaceWire-1Gb (see [1]). For synthesis Mentor Graphics Precision RTL Synthesis 2010a_Update2.254 is used, for place and route XILINX ISE 13.1.

As one can see, both implementations need approximately the same amount of logical resources and have similar timing constraints. The initial expectation of having a smaller footprint could not be proven. While the strict separation of channels increases modularity it also requires additional resources for rate transition logic.

For the tests an *exchange level* with dedicated instrumentation interfaces was connected to a special test node which is able to generate various test patterns (Fig. 4). The configuration interface allows to read status information and to configure the test patterns. Disconnection errors, 8b10b encoding/decoding errors (wrong 10b character, running disparity error), CRC errors as well as FIFO states are monitored.

Three performance tests are performed to show that SpaceWire-HS is a valuable alternative to SpaceWire-1Gb:

1. SpaceWire packet transmission with variable packet lengths and variable packet rate synchronous and asynchronous to *time-codes* transmission (24 hours nonstop)
2. SpaceWire packet transmission with fixed packet length and maximum packet rate (10 days nonstop).
3. Loss of connection and resynchronization (100 times).

All tests passed successfully. No error occurred during the test phases.

Figure 5: Gigabit Connector (Comtronics CMRM)

	SpaceWire-1Gb [1]		SpaceWire-HS	
	5VLX50	6SLX16	5VLX50	6SLX16
Global Buffers	3	3	3	3
LUTs	493	585	651	707
CLB Slices	124	147	163	177
Dffs or Latches	399	402	543	526
RAMB18	4		4	
RAMB8BWER		2		2
clk_rx	266 MHz	223 MHz	265 MHz	200 MHz
clk_tx	239 MHz	163 MHz	183 MHz	139 MHz
clk_node	264 MHz	181 MHz	249 MHz	154 MHz

Table 1: Synthesis Results for SpaceWire-1Gb and Spacewire-HS for Xilinx Virtex-5 and Spartan-6

IV. Conclusion

This publication shows that the concepts of IEEE1355 high-speed links can be adapted to the SpaceWire requirements. A specification is presented, which is optimized for 8b10b encoding. It defines a dedicated mapping of SpaceWire symbols to 8b10b-characters, which considers *time-codes* as well as power dissipation aspects.

The experimental results show that SpaceWire-HS is a valuable alternative to the implementation of SpaceWire-1Gb[1], which is complex due to the specific timing of 8b10b encoding, physical layer circuits and *exchange level*. Therefore, SpaceWire-1Gb needs extensive timeout parameter tuning. Hence, adaption to various bandwidths is cumbersome and error prone. SpaceWire-HS gets along without parameter tuning, since the timing of *character level* and *physical layer* does not influence the *exchange level*.

IEEE1355-HS has a dedicated reset, which is neglected for this evaluation. Further discussions should clarify, if a reset is really necessary.

The current implementation lacks a timeout at the **rx_nchar** interface, which detects open packets in case the link falls down. This failure is neglected in IEEE1355. However, open packets can clog the residual network and should be closed in case of broken link.

As recommended by the authors of IEEE1355 a CRC is added to extend the error detection capabilities. Currently 8-Bit CRC is chosen, but this is not enough for most applications, which have stricter requirements on probability of failure per hour and packet size. Furthermore, the polynomial for the CRC is still an open issue and should also be discussed.

In a next step a more optimized implementation should be evaluated. The footprint of the modular approach presented in this publication is not comparable to the optimized implementation presented in [1]. However, the modular approach enables to add additional channels for future analysis, e.g. multiple *nchar*-channels with dedicated bandwidth via one shared 8b10b link.

This presentation is a first proof of concept. In a next step SpaceWire-HS will be integrated with the DLRs Hand-Arm-System, a robotic system with 52 motors and more than 430 sensors. Then, the *exchange level* implementation will be tested intensively.

REFERENCES

[1] M. Nickl and S. Jörg, T. Bahls, A. Nothhelfer, S. Strasser (2011) SpaceWire, A Backbone For Humanoid Robotic Systems, In Proceedings of the International Spacewire Conference 2011, pp. 356-359

[2] IEEE Std 1355-1995, IEEE Standard for Heterogeneous InterConnect (HIC).

[3] ECSS-E-ST-50-12C, The SpaceWire Standard.

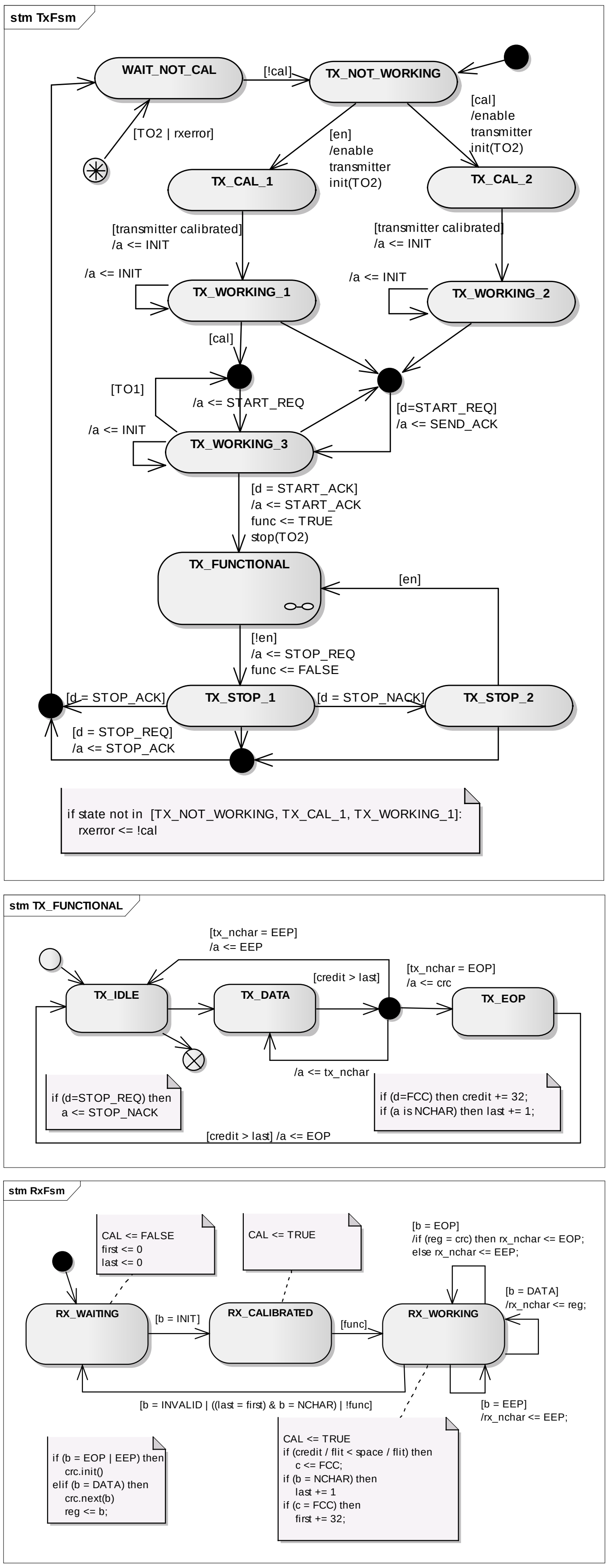
[4] A. X. Widmer and P. A. Franaszek, A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code, IBM Journal of Research and Development 27 (5): pp. 440.

[5] Ginosar, R., "Fourteen ways to fool your synchronizer," Asynchronous Circuits and Systems, 2003. Proceedings. Ninth International Symposium on , vol., no., pp.89,96, 12-15 May 2003, doi: 10.1109/ASYNC.2003.1199169

# channel definition
#
C_{fw} = DATA(x) | EOP | EEP
C_{bw} = FCC
C_{tc} = TIMECODE(tc, ctrl)
C_{hs} = RESET | START_REQ | START_ACK | STOP_REQ | STOP_ACK | STOP_NACK
# interfaces
tx_tc = C_{tc}
rx_tc = C_{tc}
tx_nchar = C_{fw}
rx_nchar = C_{fw}
en=TRUE | FALSE
tx_8b10b = 8B10B;
rx_8b10b = 8B10B;
# internal channels
a = C_{fw} | C_{hs}
b = C_{fw} | INVALID
c = C_{bw}
d = C_{hs}
e = C_{tc}
f = C_{tc}
g= C_{fw} | C_{bw} | C_{hs} | C_{tc}
h= C_{fw} | C_{bw} | C_{hs} | C_{tc} | INVALID
cal = TRUE | FALSE
func = TRUE | FALSE

# 8B10B encoding
#
INIT= COMMA 2^n*(IDLE)
TIMECODE(tc, ctrl) = K.28.1 D.$tc_{4:0}$.($ctrl_{1:0}$ & tc_5)
COMMA=K.28.5
IDLE= K.28.0
FCT=K.28.2
START_ACK= K.28.6 D.1.0
STOP_REQ = K.28.6 D.2.0
STOP_ACK = K.28.6 D.3.0
STOP_NACK = K.28.6 D.4.0
START_REQ = K.28.6 D.5.0
RESET = K.28.6 D.6.0
DATA($x_{7:0}$) = D.$x_{4:0}$.$x_{7:5}$
EOP= K.28.3
EEP= K.28.4
8B10B = K.28.*y* | D.*x*.*y*
INVALID=<received invalid character or invalid running disparity>
flit = 32

Figure 6: Encoding for 8b10b



stm TxFsm
WAIT_NOT_CAL
[!cal]
TX_NOT_WORKING
[TO2 | rxerror]
[en] /enable transmitter init(TO2)
[cal] /enable transmitter init(TO2)
TX_CAL_1
TX_CAL_2
[transmitter calibrated] /a <= INIT
[transmitter calibrated] /a <= INIT
/a <= INIT
TX_WORKING_1
/a <= INIT
TX_WORKING_2
[cal]
[TO1]
/a <= START_REQ
[d=START_REQ] /a <= SEND_ACK
/a <= INIT
TX_WORKING_3
[d = START_ACK] /a <= START_ACK func <= TRUE stop(TO2)
TX_FUNCTIONAL
[en]
[!en] /a <= STOP_REQ func <= FALSE
[d = STOP_ACK]
TX_STOP_1
[d = STOP_NACK]
TX_STOP_2
[d = STOP_REQ] /a <= STOP_ACK
if state not in [TX_NOT_WORKING, TX_CAL_1, TX_WORKING_1]: rxerror <= !cal
stm TX_FUNCTIONAL
[tx_nchar = EEP] /a <= EEP
TX_IDLE
TX_DATA
[credit > last]
[tx_nchar = EOP] /a <= crc
TX_EOP
/a <= tx_nchar
if (d=STOP_REQ) then a <= STOP_NACK
if (d=FCC) then credit += 32; if (a is NCHAR) then last += 1;
[credit > last] /a <= EOP
stm RxFsm
CAL <= FALSE first <= 0 last <= 0
CAL <= TRUE
[b = EOP] /if (reg = crc) then rx_nchar <= EOP; else rx_nchar <= EEP;
RX_WAITING
[b = INIT]
RX_CALIBRATED
[func]
RX_WORKING
[b = DATA] /rx_nchar <= reg;
[b = INVALID | ((last = first) & b = NCHAR) | !func]
[b = EEP] /rx_nchar <= EEP;
if (b = EOP | EEP) then crc.init() elif (b = DATA) then crc.next(b) reg <= b;
CAL <= TRUE if (credit / flit < space / flit) then c <= FCC; if (b = NCHAR) then last += 1 if (c = FCC) then first += 32;

Figure 7: Protocol automata TxFsm and RxFsm

Implementation and Interoperability Tests of SpaceFibre

Session: SpaceWire standardisation, Short Paper

Takahiko Masuzaki, Minoru Nakamura, Tetsuro Kato, and Yasunori Ido
Information Technology R&D Center,
Mitsubishi Electric Corporation,
5-1-1, Ofuna, Kamakura, Kanagawa, 247-8501, Japan
Masuzaki.Takahiko@dc.MitsubishiElectric.co.jp,
Nakamura.Minoru@ea.MitsubishiElectric.co.jp,
Kato.Tetsuro@dr.MitsubishiElectric.co.jp,
Ido.Yasunori@eb.MitsubishiElectric.co.jp

Toru Sasaki
Advanced Technology R&D Center,
Mitsubishi Electric Corporation,
8-1-1, Tsukaguchi-Honmachi, Amagasaki, Hyogo, 661-8661, Japan
Sasaki.Toru@eb. MitsubishiElectric.co.jp

***Abstract*—SpaceFibre is a next-generation interconnect standard for connecting components in a spacecraft. Standardization of SpaceFibre is led by the European Space Agency (ESA). Although some features (e.g., multi-lane) are under consideration, a Draft E1 version of the standard was released at the end of September 2012. To use SpaceFibre, it is necessary to be easy to use by eliminating the ambiguity of the standard and by increasing the perfection level of the standard. In order to check whether or not there are some points that can be improved in the draft standard such as ambiguity, we made a prototype implementation of SpaceFibre Draft E1 version on an FPGA. We used a SerDes-IC TLK2711 (Texas Instruments) for functions of the physical layer and a part of the encoding layer. We also used an FPGA (XILINX® Virtex®-5) for the upper layers. In this paper, we report the results of confirming the behavior by the prototype implementation, and the evaluation of the Draft E1 based on the results. In addition, we ran interoperability tests with a Japanese company and the University of Dundee in order to verify the interoperability of the draft standard. There, timeout occurred in the lane initialization that may be improved.**

***Index Terms*—SpaceWire, SpaceFibre, interoperability test**

I. Introduction

SpaceFibre[1] is a very high-speed serial interconnect standard for spacecraft and is led by the European Space Agency (ESA). The background in which SpaceFibre was proposed is a demand for high speed and scalability. The amount of the sensor data has increased in every mission. However, before SpaceFibre, there was no standard that is high-speed, scalable and easy-to-use. Therefore, developing a new technology for data transmission has been needed for every mission. As a result, the problems of the high cost and increase development time has occurred. On the other hand, SpaceFibre can be reused because this standard satisfies the demand for high data rate over a long period of time. As a result, we were able to improve reliability and reduce development costs. Also, SpaceFibre has the potential to unify the interface in a satellite with SpaceWire[2] because SpaceFibre has affinity with SpaceWire. Thus, a backplane prototype with SpaceWire and SpaceFibre links was proposed[3].

Based on the above, we have evaluated SpaceFibre Standard to make effective use of the standard. In this paper, we describe our SpaceFibre CODEC prototype implementation, interoperability test results and consideration about lane initialization.

II. SpaceFibre

SpaceFibre Standard Draft E1 was released at the end of September 2012. Although some features (e.g., multi-lane) are under consideration, the standard is nearing completion.

Fig. 1 shows SpaceFibre's layer structure. A description of the layers is provided below.

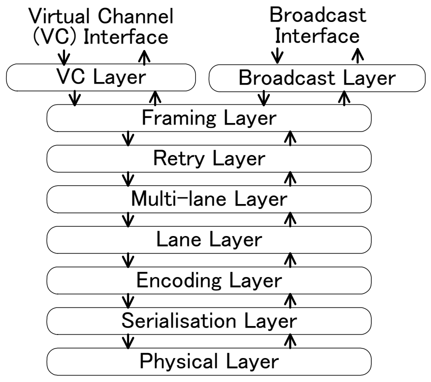


Fig. 1. Layers of SpaceFibre

(1) Virtual channel layer has an interface to user application and handles QoS control.
(2) Broadcast layer broadcasts short messages.
(3) Framing layer is responsible for framing data.
(4) Retry layer resends data to recover from errors.
(5) Multi-lane layer operates serveral lanes in parallel.
(6) Lane layer initializes lane.
(7) Encoding layer encodes/decodes 8b/10b code and does word synchronisation.
(8) Serialisation layer converts parallel data and serial data.
(9) Physical layer defines electrical signals.

III. Implementation of SpaceFibre

We made a prototype implementation based on Draft E1. Fig. 2 shows the parts of implementation. We implemented the broadcast layer, framing layer, lane layer, encoding layer, and partially implemented the virtual channel layer, retry layer in FPGA. We implemented SpaceFibre CODEC limited to the minimum functions. We have excluded the multi-lane layer from the prototype because the layer is incomplete in Draft E1. We have not supported the multi-virtual channels and retry function as yet. We will implement these functions in the near future.

Fig. 3 shows an outline of the SpaceFibre Evaluation Board, and Fig. 4 shows its photo. We used a SerDes-IC TLK2711 (Texas Instruments) for the functions of the physical layer and 8b/10b encoding/decoding of the encoding layer. We also used an FPGA (XILINX Virtex-5) for the upper layers. In order to control the FPGA, we also mounted a CPU on the same board. SpaceFibre CODEC can be controlled by a PC via this CPU.

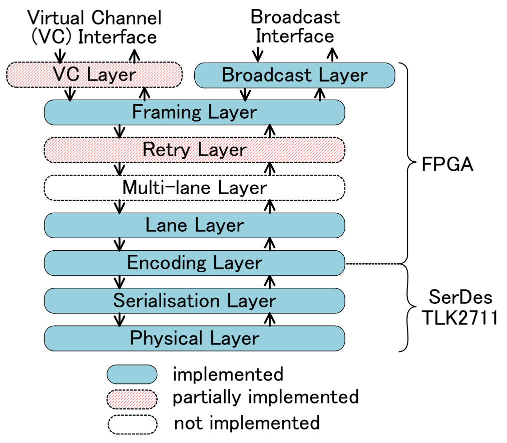


Fig. 2. Parts of Implementation

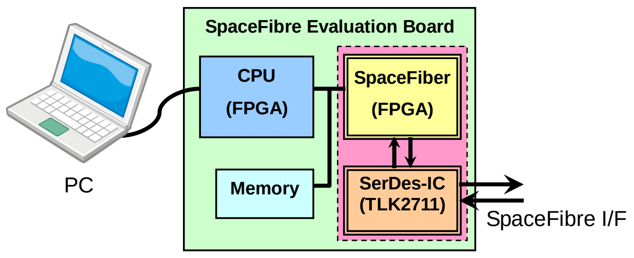


Fig. 3. Outline of SpaceFibre Evaluation Board

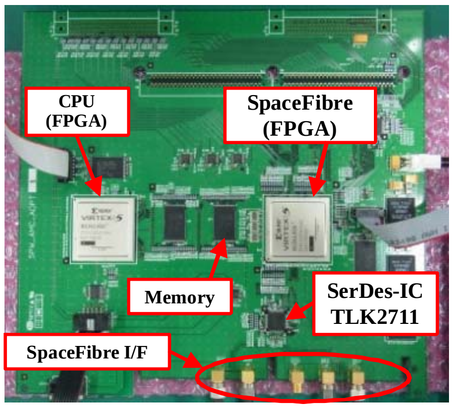


Fig. 4. SpaceFibre Evaluation Board

IV. Interoperability Tests

A. Outline

We joined an interoperability test meeting arranged by the Japan SpaceWire User Group, to confirm the basic interoperability of the transmission. In the beginning, we ran an interoperability test in Japan. After that, we ran a test with the University of Dundee / STAR-Dundee Ltd (Dundee). The aim of these interoperability tests is to confirm whether or not lane initialization and basic transmission of frames are performed with no issues. The implementation was based on the Draft E1.

Prior to the interoperability test with Dundee, we had tests in Japan. The tests in Japan are lane initialization test, single-shot frame test, and sequential frames test. Because there was an issue in the lane initialization test, we recognized that 20us timeout period defined in Draft E1 may have to be longer.

Based on the results of the tests in Japan, Dundee and our team implemented a function to change the timeout period. Figure 5 shows the test environment with Dundee, and Fig. 6 shows its photo. The Dundee prototype's name is STAR Fire. The rate of connection was 2.5Gbps. The SMA cable in our prototype and eSATA cable were one meter each. Our prototype and STAR Fire are connected to each PC. PCs can send various commands to prototypes, and can monitor received data, and so on. TABLE I shows the summary of the test items and results of this interoperability test. We tested three major items as shown in TABLE I. Because these items are essential to basic communication, these items should be passed. Though some issues occurred, very basic parts of the lane initialization and the data transmission worked properly. The details are described in the next section.

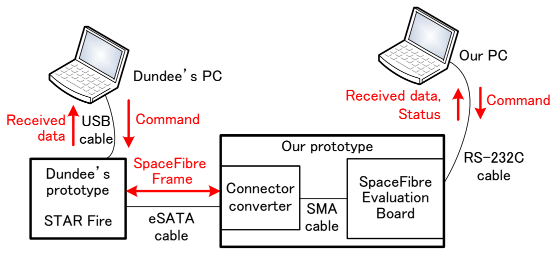


Fig. 5. Test Environment

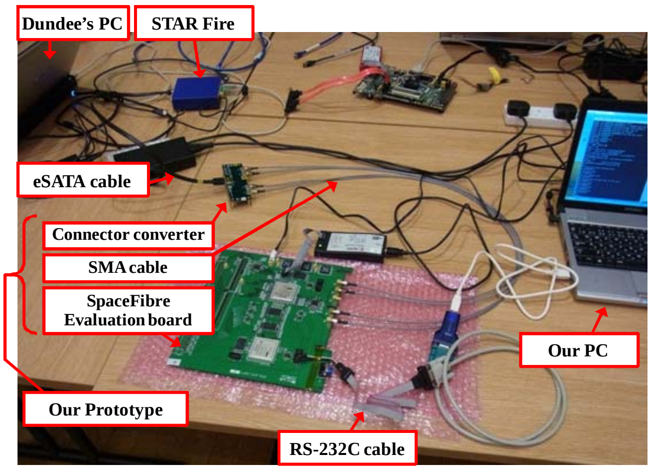


Fig. 6. Test Environment

TABLE I. OUTLINE OF TEST ITEMS AND RESULTS

No.	Test Item	Result
1	Lane initialization	Timeout occurred.
2	Transmission of single-shot frames	OK.
3	Loopback transmission of sequential frames	STAR Fire detected received data error.

B. Test Items and Results

In the interoperability test with Dundee, the following three tests were run.

1) Lane initialization

a) Test Item

This test is a lane initialization test. Confirmation items are that both transceivers succeed in link-up, that both transceivers send and receive flow control token (FCT) control words, and whether or not timeout occurs in the process of lane initialization.

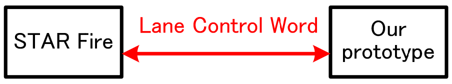


Fig. 7. Lane initialization test

b) Result

Link-up and sending/receiving FCTs succeeded, but timeout occurred. TABLE II shows the number of timeout counts. In this test, timeout was counted after both transceivers started the lane initialization. If the timeout period is sufficient, timeout seems not to occur because of the structure of the state machine. However, even though the timeout period is extended to 1ms, the number of timeout counts seems not to decrease.

When "No signal" is detected, the lane initialization state machine returns to the initial state. Thus, if one side is not sending words temporarily, both sides can restart in the same timing. As a result, the detection "No signal" prevents timeout. We tried both on and off of the "No signal" detection in our prototype. However, even if the setting was on, "No signal" could not be detected in the last three trials (number of timeout counts: 9, 4, 7) in our prototype (other trials could not be checked).

The consideration about this issue is described in chapter V.

TABLE II. NUMBER OF TIMEOUT COUNTS

No.	No signal detect setting (Our prototype)	Timeout period setting (both)	Number of trials	Result: number of timeout counts
1	Off	300us	2	1, 12
2	Off	500us	2	11, 3
3	Off	1ms	4	2, 4, 24, 13
4	On	1ms	23	8, 8, 4, 15, 1, 1, 12, 7, 8, 3, 2, 1, 1, 3, 1, 14, 6, 1, 17, 21, 9, 4, 7

2) Transmission of single-shot frames

a) Test Item

Our prototype and STAR Fire send single-shot frames (data frames and broadcast frames) to each other. Confirmation item is whether the receiving side receives the frames and the receiver detects no error. For example, our prototype sends one-, two-, and 64-word single-shot data frames.

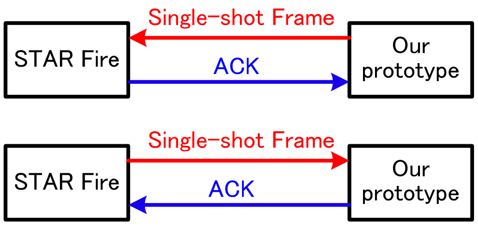


Fig. 8. Transmission of single-shot frame test

b) Result

Our prototype and STAR Fire successfully received the data frames and the broadcast frames with no error.

3) Loopback transmission of sequential frames

a) Test Item

STAR Fire sends sequential data frames. The data frames are sent back by our prototype in the user application layer. STAR Fire receives the data frames.

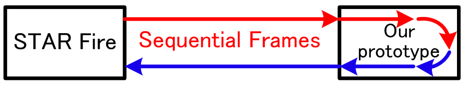


Fig. 9. Loopback transmission of sequential frames test

b) Result

Our prototype received all frames with no error. STAR Fire received the first several frames with no error. However, our prototype detected NACK control words after several frames or several hundreds frames. NACK indicates an error occurred. This issue is under investigation.

V. Consideration about Lane Initialization

In the link initialization test, timeout occurred. Although we are currently investigating this issue, we have described this issue in this chapter.

First of all, there is inconsistency in the device specifications and timeout period defined by the standard. Our prototype uses a SerDes-IC TLK2711, which is assumed as a device to be used for SpaceFibre. According to TABLE III, this SerDes-IC's maximum PLL startup lock time is 0.4ms [3]. STAR Fire uses Spartan®-6 transceiver. According to TABLE IV, the transceiver's maximum PLL lock time is 1ms, and maximum lock time to data is 200us[4]. Thus, at most 1.2ms is needed to lock to data after starting PLL. On the other hand, the timeout period in Draft E1 is 20us. Therefore, timeout occurs before the transceiver PLLs lock to data. Accordingly, the timeout period is too short to link to data; or, it may be better not to disable the PLL on every timeout.

TABLE III. Specification of TLK2711

PARAMETER	NOM	Max	Units
PLL startup lock time	0.1	0.4	ms

TABLE IV. Specification of Spartan-6's Transceiver

Description	Conditions	Max	Units
Clock recovery frequency acquisition time	Initial PLL lock	1	ms
Clock recovery phase acquisition time	Lock to data after PLL has locked to the reference clock	200	µs

Secondly, sending and receiving of INIT1 control words are started immediately after transceiver PLL is enabled. If INIT1 is detected by some chance, the initialization state machine goes to the next state even if PLL is unstable. In addition, when unexpected words are received, the state machine does not return to initial state. This may cause unexpected behavior. We believe that a mechanism to promote stable state is needed. An example of the mechanism is where the state machine moves to the next state after PLL locks to data, and when unexpected data is received, the state machine returns to initial state.

Finally, there is a need to clarify the definition of "No signal." The draft standard says "No Signal means no signal detected at receiver inputs." However, the meaning of "No signal" is not clear. TLK2711 has a loss of signal detection function, and our prototype uses this function. However, Xilinx FPGAs do not have "No signal" detection function. In this way, functions depend on the devices. We believe that a device-independent definition is needed, or another possible option is to not use "No signal".

Because there are unclear points, we could not determine the cause of timeout. The points to clarify are the details of state transition and sent/received control words' accuracy during lane initialization. In order to investigate the cause of timeout, we have to add tests and analysis.

VI. Conclusion

We made a prototype implementing SpaceFibre CODEC in FPGA with SerDes-IC TLK2711, and confirmed its behavior. First, we determined an issue in initialization through testing in Japan. Second, we checked lane initialization and basic transmission of frames, by interoperability testing with the University of Dundee. As a result of the test, we extracted an issue that timeout occurred during the lane initialization. The lane initialization may have to be improved, though the timeout mechanism should be clarified with additional tests and analysis.

References

[1] S. Parkes, A. Ferrer, A. Gonzalez & C. McClements, "SpaceFibre Standard Draft E1," University of Dundee, September 2012.

[2] ECSS-E-ST-50-12C, "SpaceWire – Links, nodes, routers and networks, " July 2012.

[3] M. Nakamura, T. Ito, Y. Takeda, I. Odagi, S. Hirakuri, K. Yamagishi, K.Shibuya, and M. Nomachi, "SpaceWire Backplane With High-Speed SpaceFibre Link, " June 2010, pp 163-166.

[4] Texas Instruments, Inc., "TLK2711A 1.6 to 2.7 GBPS TRANSCEIVER. (Rev. B)," October 2012.

[5] Xilinx, Inc., "Spartan-6 FPGA Data Sheet: DC and Switching Characteristics," October 2011.

Performance evaluations and proposal to improve next-generation SpaceFibre protocol

Standardisation, Short Paper

Yu Otake[1], Kohei Hosokawa[1], Yasuhiro Sota[1], Takahiko Tanaka[1], and Hiroki Hihara[2]
[1]NEC Corporation Tokyo, Japan
[2]NEC TOSHIBA Space Systems, Ltd. Tokyo, Japan
y-otake@bp.jp.nec.com

***Abstract*— The SpaceFibre protocol, which introduces the high-speed serial data-links widely adopted for peripheral component interconnect (PCI) express and serial advanced technology attachment (ATA), was developed by the European Space Agency as a standard protocol to communicate between payloads in satellite networks. It is important to implement and evaluate the present SpaceFibre protocol draft to improve the protocol. Therefore, we implemented the SpaceFibre Draft-E1 protocol on a field-programmable gate array (FPGA) and evaluated its performance in five points by: (A) evaluating the effective data-throughput and latency of packets when the bit error rate (BER) on SpaceFibre links varied, (B) analyzing latency of packets in each SpaceFibre protocol layer in detail, (C) evaluating latency in data-transmission through broadcast channels, (D) evaluating latency of virtual channels set to bandwidth reservation QoS, and (E) evaluating the implementation results on an FPGA, such as the number of logics, memory usage, and operating speed.**

We report the results obtained from these evaluations and a method to reduce the latency of packets by analyzing these results. We also report results from evaluating the method and compare it with the SpaceFibre Draft-E1 protocol.

***Index Terms*— SpaceFibre, Latency, Throughput, Virtual Channel.**

I. Introduction

The SpaceFibre protocol newly adapted virtual channels (VCs) that play the role of sending and receiving SpaceWire packets from several sources to corresponding destinations over SpaceFibre links [1, 2]. The VC layer also provides a quality of service (QoS), i.e., best effort, priority, bandwidth reservation, and time-slot to accommodate difference communication demands. Furthermore, the SpaceFibre also adapted error-detection and recovery techniques at the link level accomplished by adding frame sequence numbers and cyclic redundancy check (CRC) checksums to data frames.

It is important to implement and evaluate the present SpaceFibre protocol draft to improve the protocol, especially VC and retry layers. Therefore, we implemented the SpaceFibre Draft-E1 protocol on an FPGA. We utilized a ML507 FPGA evaluation board including one Xilinx Virtex-5 FX70 and constructed an environment to assess the SpaceFibre protocol by using STAR-Fire and SpaceWire USB Brick in Figure 1 [3]. In some evaluations, we only use the ML507 board in back-to-back mode as shown in Figure 2. Our implementation has four VCs and an error injector to evaluate the behavior of QoS and error recovery in the FPGA. Both the encoding and serialization layers were implemented by the Rocket I/O provided by Xilinx FPGA.

Here, we report the results from evaluating effective data throughput and latency of packets in section II and a method to reduce latency of packets in section III.

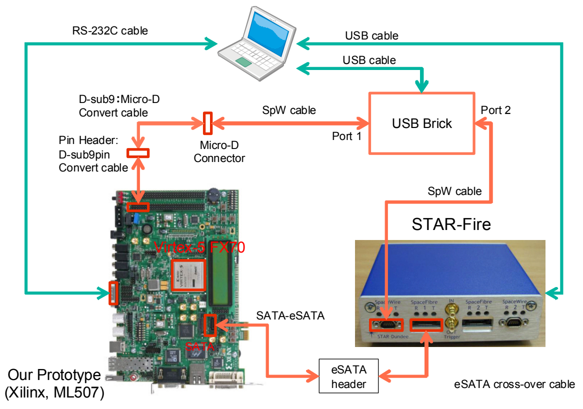


Figure 1 Evaluation environnement for SpaceFibre protocol

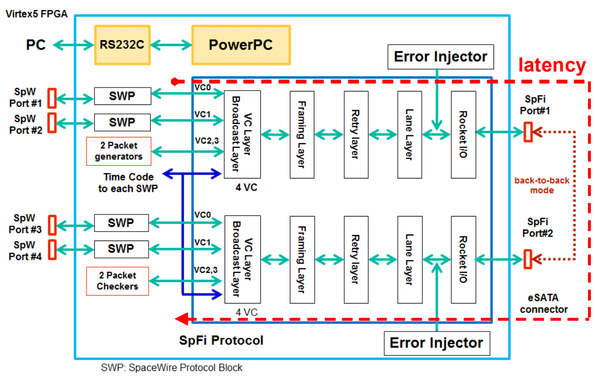


Figure 2 Architecture on Virtex5 FPGA

II. Evaluation of SpaceFibre Draft-E1 Protocol

We evaluated the present SpaceFibre Draft-E1 protocol in five points by:

A) Evaluating effective data throughput and latency of packets when the BER on SpaceFibre links varies,

B) Analyzing the latency of packets in each SpaceFibre protocol layer in detail,

C) Evaluating the latency in data-transmission through broadcast channels,

D) Evaluating the latencies of VCs set to bandwidth reservation QoS, and

E) Evaluating the results from implementation on an FPGA, such as the number of logics, memory usage, and operating speed.

The packet length in these evaluations was fixed to 64 words (1 word = 4 bytes) and the signaling rate on the SpaceFibre link was 2.5 Gbit/s. Therefore, the maximum throughput was 2.0 Gbit/s since the SpaceFibre protocol adapts 8B/10B encoding. The length of data transmission link (eSATA crossover cable) is about 50cm, and multi-lane layer is not implemented. All latencies in this paper are sum of a packet transmitting time through SpaceFibre from the output VC buffer (VCB) in the transmit side to the input VCB in the receirve side shown in Figure 2 and a waiting time due to full of the output VCB.

A. Evaluation of effective data throughput and latency of packets

We first evaluated what influence resending packets due to bit error had on data throughput and latency of packets. Figure 3 shows the average latency of packets when the BER on the SpaceFibre link varies. As shown in Figure 3, the average latency is equal to 2.72 μs if BER is less than $5*10^{-6}$.In the case input data rate is 95%, the latency rapidly increases when BER is more than $5*10^{-6}$. In the case input data rate is 65% the latency increases when BER is more than $5*10^{-5}$. Therefore we can see the latency depends on input data rate. If BER is higher than $4*10^{-4}$, the link cannot be established because a bit error is always occurred when data frame size is 64 words (2048 bit = 2560 bit at the SpaceFibre link) in any input data rates. If multiple VCs are operated, another latency by QoS is added to this latency. The detail analysis of average latency 2.72 μs is described at the next subsection.

We also evaluated effective data throughput in the same case. As shown in Figure 4, If BER is less than 10^{-6}, the data throughput is about 1.9 Gbps, which is not equal to 2.0Gbps because some control commands (start-of-frame, end-of-frame, ACK/NACK, and FCT) are required to transfer data. If BER is higher than 10^{-6}, the effective data throughput is gently decreased. Finally, the effective throughput is 1.34Gpbs, which is only 70% of the maximum throughput, if BER is 10^{-4}. From the above evaluations, not only SpaceFibre link usage but also BER should be considered to build a satellite system.

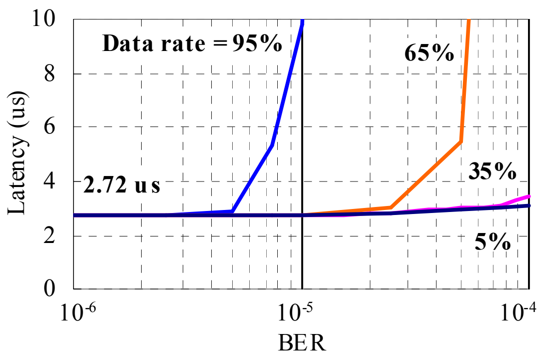


Figure 3 Relation between packet latency and bit error rate

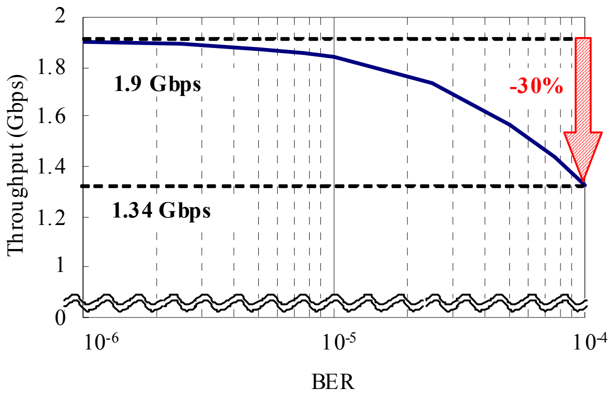


Figure 4 Relation between data throughput and bit error rate

B. Detailed analysis of latency of packets in each SpaceFibre protocol layer

The details on latency of packet in each layer of the SpaceFibre protocol on both the transmit (TX) and receive (RX) sides are summarized in Table 1.

Table 1 Details on packet latency in SpaceFibre layers

	TX	RX
Virtual channel layer	1.11 μs	0.08 μs
Framing layer	0.03 μs	0.01 μs
Retry layer	0.03 μs / *1.95 μs	1.11 μs
Lane layer	0.03 μs	0.03 μs
Physical layer	0.32 μs	

*When one resend is occurred

In this evaluation, the latency of physical layer includes cable transmission delay and heavily depends on the specificaion of Rocket I/O provided by Xilinx FPGA, especially, elastic buffer size. This value may be changed if another physical layer, like TLK-2711-SP, is applied.

As shown in Table 1, the total latency is 2.72 μs when there is no retry, and the latency increases 1.95 μs every resending, which includes the transmission delay of NACK command from RX to TX and that of resending data. This time

1.95 μs is added to the TX retry layer in this paper since ACK/NACK and resending data is generated in the retry layer. The only two layers, which are VC layer at TX side and retry layer at RX side, occupy 80% of the total latency without resending. The time at TX side in VC layer mostly spends to store 64 words into the output VCB because the data in the VCB are sent to the medium access controller if the data size is euqal to or more than 64 words or the data has an end of packet (EOP) marker. On the other hand, the time at RX side in the retry layer spends to calculate CRC. We will report a method to reduce the total latency by making the frame length short at section III because these latencies are depend on the data frame size.

C. Evaluation of transmission through broadcast channels

We evaluated latency in data transmission that was sent through the broadcast channel (BC) layer. It is important to evaluate the influence on latency since a time-code packet to synchronize all equipments on the satellite system sends through BC. The length of the broadcast frame is 2 words which is shorter than that of the data frame that is up to 64 words in almost all cases. Therefore, the broadcast frame had less influence on BER than the data frame. Table 2 summarizes the latency and possibility for the broadcast frames when BER is 10^{-4}.

Table 2 Latency in broadcast frames in retry cases

Retries	Latency (μs)	Possibility (%)
0	0.512	99.2
1	1.440	0.7936
2	2.368	0.0063
3	3.296	0.0001

Few broadcast frames are resent in almost all cases as Table 2 shows. However, the latency in broadcast packets increases three or more times when they are resent. It will therefore be necessary to take this latency into consideration when constructing a time-synchronized system using the SpaceFibre protocol. These evaluation results are for cases where no data frames are transmitted. Even if data frames are transmitted, the latency in the broadcast frames is considered to be the same as that in Table 2, since the broadcast frames have top priority for sending and can be inserted into data frames.

D. Evaluation of bandwidth reservation QoS

There are three parameters, time-slot, priority, and expected bandwidth, for each VC as QoS in the SpaceFibre protocol (the best effor QoS is considerd as lowest priotiry QoS). The time-slot parameter controls whether or not the data in the VC is transfered by time. The parameter can support time-division transmition used in deterministic networks. The second parameter is a priority. This is a simple parameter because data in VC with high priority is transfered at first. Since the precedence of these two parameters are not changed with the usage rate of the SpaceFibre link, we don't evaluate these parameters in this paper. Therefore, we evaluate the data latency when the expected bandwidth parameter is changed, when the time-slot and the priority parameters are set to same values in all VCs, and when two packet generators, that can periodically generate specified data frame rate, are connected to VC0 and VC1. In this evaluation, each value of expected bandwidth is set to the actual bandwidth which is the ratio between data signaling rate of the SpaceFibre link and each input data rate. Table 3 shows the latency of packets sent from VC0 when input data rate of VC0 and that of VC1 are varied.

Table 3 Packet latency of VC0 using bandwidth reservation QoS

		Input data rate (VC1)				
		5%	25%	45%	65%	85%
Input data rate (VC0)	5%	2.72 us	2.72 us	2.72 us	2.72 us	2.72 us
	25%	2.93 us	2.72 us	2.88 us	3.00us	
	45%	2.85 us	2.94 us	2.75 us		
	65%	2.86 us	3.10 us			
	85%	3.02 us				

When the total input rate of VC0 and VC1 is less than 100% of the SpaceFibre link and the expected bandwidth is set to the actual bandwidth, the packet latency including the latency added by QoS is around 3 μs as summarized in Table 3. Latency of VC0 becomes smaller in this region when the input data rate of VC0 is smaller than that of VC1. The reason why the less expected bandwidth have higher priority when VC are ready to send is because data generator produces data periodically in this evaluation.

E. Evaluation of implementation results on FPGA

The number of primitive circuits used to implement the SpaceFibre Draft-E1 protocol are listed in Table 4, where LUT is a look-up table with six inputs and one output, FF is a one bit flip-flop, and BlockRAM is an 18-Kbit dual-port random access memory. All primitive circuits are based on the Virtex-5 architecture. The size of the virtual channel layer in Table 4 is where four VCs are mounted. Currently, the data bus on the FPGA was 32 bits wide and the operating frequency was 62.5 MHz for the 2.5Gbps SpaceFibre link.

Table 4 Details on SpaceFibre circuit size

	LUT	FF	BlockRAM
Virtual channel layer (4 VCs)	2510	1840	8
Framing layer	99	29	2
Retry layer	2198	1596	11
Lane layer	389	357	0
Physical layer	427	488	0
Total	5623	4310	21

III. Method to Reduce Packet Latency

A. Algorithm

As described in Section II.B, packet latency in the SpaceFibre protocol is mostly occupied by both the VC layer on the TX side and the retry layer on the RX side to buffer at least one frame. Data buffering in the retry layer on the RX side is necessary to determine if there are any errors in the received data by checking CRC and the frame sequence number which are located at the end of the frame. However, data buffering in the virtual channel layer is not necessary when there is sufficient space to send data at the SpaceFibre link. We therefore evaluate a method that decides to buffer data to the output VCB depending on the condition of the SpaceFibre link in two ways;

1. When there is at least one word in the output VCB regardless of EOP, the output VCB should notify the medium access controller that it has data ready to form a data frame.
2. When the output VCB is permitted to send data, the frame length is set to the same value as the number of data in the output VCB. However, if there are more than 64 words of data in the output VCB, the frame length is set to 64 words.

If there is room to send data at the SpaceFibre link, the packet latency can be decreased since the permission for the output VCB to send data is quickly provided in the method given above. On the other hand, if there is no room at the SpaceFibre link, the latency for the method will be same as that for the Draft-E1 protocol since the almost all frame lengths to buffer data in the method are set to the same value as those in the Draft-E1 protocol. The packet latency of the method is evaluated at Section III.B.

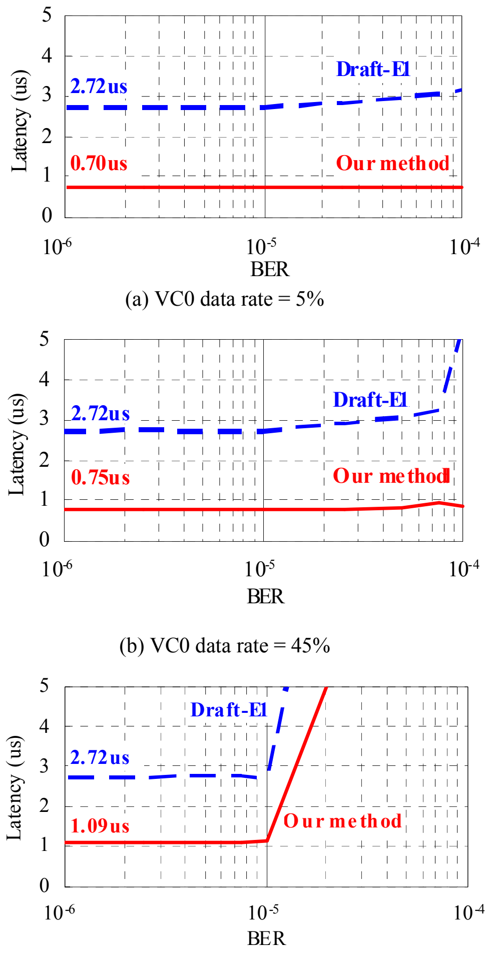
However, there is concern about the decrease in data throughput caused by increasing in control commands to form data frames due to divide a packet into some short frames. Therefore, the data throughput with the method and that with the Draft-E1 protocol are compared in Section III.C.

The implementation also can be slightly reduced because the method does not take into account the number of EOPs in the output VCB. On the other hand, the current specification needs EOP detectors at input and output in the output VCB to manage a number of EOPs.

B. Evaluation of latency with this method

We implemented the method into the evaluation board to evaluate the reduction in latency. Latency with the method and that with the present Draft-E1 protocol are compared in Figure 5 when only VC0 send a data. Figure 5(a), (b), and (c) plot the latencies of packets through VC0 where each input data rate is 5%, 45%, and 85%, respectively.

As Figure 5(a) and (b) show, the latency with the method is about 0.7 µs, which can be decreased 74% from that of the Draft-E1 protocol. The method achieves a maximum reduction in the latency in these cases because there is space for the increase in control commands at the SpaceFibre link.



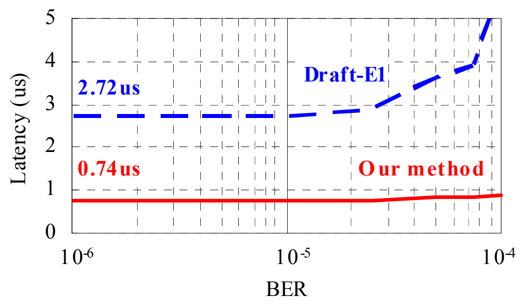
(c) VC0 data rate = 85%

Figure 5 Comparison of latency in one VC transmission

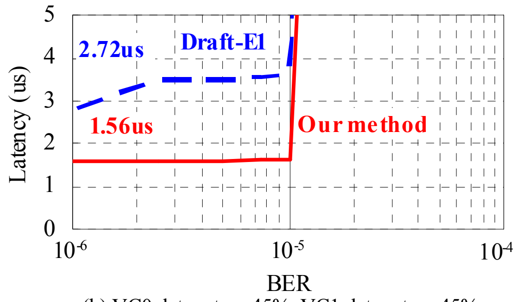
Latency with the method is 1.1 µs, which is decreased 60% from that of the Draft-E1 protocol, for the large input data rate plotted in Figure 5(c). The reason latency is reduced less is because the frame length cannot be shorten. When BER is high in this case, the latency with the method rapidly increases as with the Draft-E1 protocol because the effective bandwidth decreases as shown in Figure 4. Finally the latency with the method is equal to that with the Draft-E1 protocol when there is no room in the SpaceFibre link.

Latency with the method and that with the present Draft-E1 protocol are compared in Figure 6 when both VC0 and VC1 send a data. Figure 6 (a) and (b) plot latency of packets through VC0 where each input data rate is 5% and 45% respectively when the input data rate of VC1 is 45%.

The lateny with the method is 0.7µs when the total input data rate is 50%,as shown in Figure 6(a), and 1.6us when total input data rate is 90% as shown in Figure 6(b). Therefore the method can reduce the latency when there is room to shorten frame even if packets are transmitted from multiple VCs.

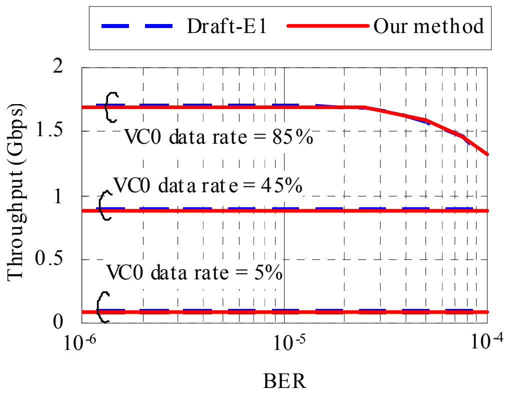


(a) VC0 data rate = 5%, VC1 data rate = 45%

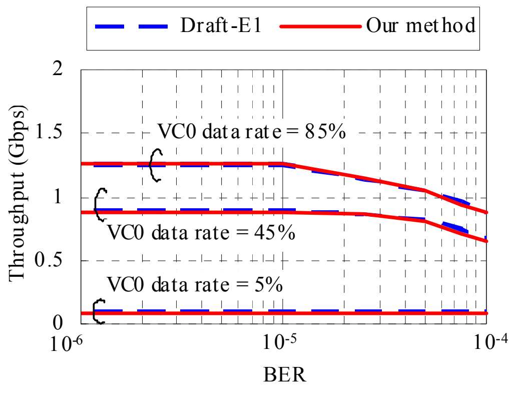


(b) VC0 data rate = 45%, VC1 data rate = 45%

Figure 6 Comparison of latency in transmission with two VCs



(a) VC1 data rate = 0 %



(b) VC1 data rate = 45%

Figure 7 Comparison of data throughput

C. Evaluation of data throughput with this method

Our evaluation of data throughput is plotted in Figure 7 that compares the method and the present Draft-E1 protocol. Figure 7(a) plots throughputs for data transmission from only VC0, in which each input data rate is 5%, 45%, and 85% respectively. We can see that effective data throughput with the method is nearly equal to that with the Draft-E1 protocol because there is sufficient space in a SpaceFibre link even if the numbers of control commands are increased. The same thing can be said for packet transmission from two VCs. Figure 7(b) compares the data throughputs of VC0 for the method and Draft-E1 where the input data rate of VC1 is set to 45% and that of VC0 is set to 5%, 45%, and 85% respectively. We also can see the effective data throughput with the method is nearly equal to that with the Draft-E1 protocol even if the total data rate is over 100%.

The results from these evaluations indicate that the method can reduce latency of packet transmission when there is a great deal of vacant space in the SpaceFibre link. Latency with the method is 1.6us which is decreased 40% from that of the Draft-E1 protocol when total input data rate is 90%. Furthermore, we confirmed that data throughput with the method was also almost same as the Draft-E1 protocol.

IV. Conclusion

We implemented and evaluated the SpaceFibre Draft-E1 protocol. The results from evaluations indicated that the average latency of packets increased rapidly when BER was above $5*10^{-6}$ and the maximum data throughput decreased 30% when BER was 10^{-4}. We also found that packet latency transmitted through the virtual channel was mostly caused in the virtual channel layer on the TX side and the retry layer on the RX side due to at least one frame of data being buffered.

We therefore found that a new method could reduce packet latency by changing the method of determining the frame length in the virtual channel layer. The latency with this method can be reduced 74% from that with the Draft-E1 protocol because the method can utilize vacant space in the SpaceFibre link effectively. We confirmed that the packet latency with the method can be reduced even if there is slightly space in the SpaceFibre link and the throughput of the method is nearly equal to the Draft-E1 in any input data rates.

References

[1] Steve Parkes at el, “SpaceFibre Standard Draft E1,” University of Dundee, 28th Sep. 2012.

[2] ESA-ESTEC, “SpaceWire,” ECSS-E-ST-50-12, 31 July 2008.

[3] Virtex-5 FPGA Family, “http://www.xilinx.com/”, Xilinx Inc.

Test & Verification (Short)

SpaceWire Interoperability Characterization

Test & Verification, Short Paper

G. Fernández Berzosa, P. Rodríguez Perochena, A. Pérez Gómez, R. Regada Álvarez, L.R. Berrojo Valero, L. Basanta Alonso.
Digital & Detection Product Line
Thales Alenia Space España
Madrid, Spain
gonzalo.fernandezberzosa@thalesaleniaspace.com, pedro.r@thalesaleniaspace.com, alberto.perezgomez@thalesaleniaspace.com, raul.regadaalvarez@thalesaleniaspace.com, luis-rafael.berrojovalero@thalesaleniaspace.com, luis.basantaalonso@thalesaleniaspace.com

***Abstract*—The generalization of the SpaceWire protocol utilization on different space systems has led to the emergence of a large variety of software/hardware components and tools from several different developers. A key point is to evaluate the interconnectivity between all these elements to avoid the risk of future problems on the integration phases of any spacecraft project. Thus, TAS-E has made an effort to develop a complete test and characterization system including various logic SpW VHDL cores, one specific ASIC and commercial debugging tools.**

***Index Terms*— SpaceWire, Interconnectivity, Star Dundee, Bepi Colombo, Goddard, SMCS332SpW, GRSPW.**

I. Introduction

It is a fact that each instrument or equipment composing any spacecraft, however small, is designed and manufactured by several different developers. This segregation makes the integration task one of the most critical points of all the existing phases in a project and thus, any possible risk must be evaluated and checked on the earlier phases of the design (if possible, before the final architecture definition).

Facing low level interconnectivity problems at the integration phase is not an option since the maturity of the equipments at this point typically does not allow any modification without involving a costly redesign of the hardware architecture of the system.

In the frame of the development of the next generation of Iridium satellites constellation, TAS-E was asked to provide the necessary support and to perform the required tests to evaluate with enough anticipation the correct interconnectivity between some equipments composing the satellite's payload. Specifically, the architecture to evaluate is composed of four equipments that are being developed by different companies with their respective headquarters located on different countries. The interconnection between these equipments is based on simple point-to-point SpaceWire links, and to implement these links, each developer has made use of a different solution.

On the frame of this support project, TAS-E decided to expand the test cases as much as possible to create a reference for future projects expecting to integrate several equipments with different SpaceWire components.

Due to the different nature of all the elements tested (VHDL IP, ASIC or external equipments) different kind of tests and interconnection architectures have been performed. The aim of this paper is to describe the parameters and conditions of every test conducted.

II. Evaluated Elements

The interoperability tests performed by TAS-E have covered, on different ways, the following elements:

- Star Dundee SpaceWire CODEC IP: VHDL code embedded on a Virtex5 FPGA.
- Bepi Colombo SpaceWire IP: VHDL version of the core developed by TAS-I in the frame of the Bepi Colombo mission embedded on a Virtex5 FPGA.
- Goddard SpaceWire IP: NASA IP core tested directly on a EM equipment composing the Iridium satellite payload.
- Atmel AT7911 (SMCS332SpW): this ASIC provides three SpaceWire interfaces to an external processing device.
- Gaisler GRSPW core: evaluated within the GR-RASTA system which provides, among other interfaces, 3 independent SpW links (RASTA 101 configuration).

Star Dundee's SpW laboratory cables have been used to interconnect every link. Also an external Link Analyzer from the same developer has been used to check the correct protocol implementation, to analyze the data transmissions and to inject errors over the links. This device has become a key element to perform a complete and fast debug on any SpaceWire link implementation.

III. SpaceWire Characterization Platform

The main element composing the evaluation system is the 'SpaceWire Characterization Platform', a board developed by TAS-E including, among other elements, the following items:

- Virtex5 FX130T: reconfigurable FPGA embedding the SpW VHDL IPs to be evaluated. It is also in charge of controlling the AT7911 SpW ASIC and providing all the necessary interfaces to access the system via an external controller (typically a PC). The internal design

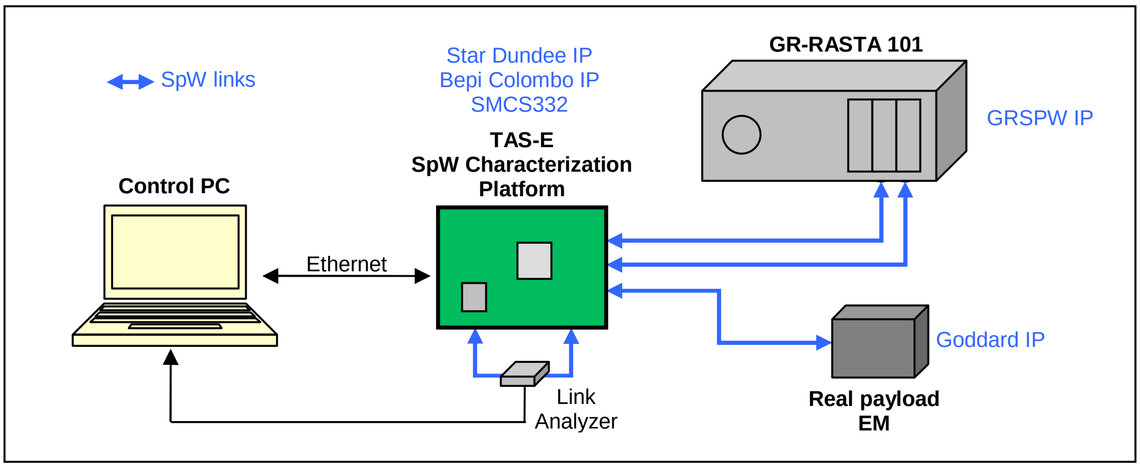


Fig. 1. Testbed configuration for the interoperability project.

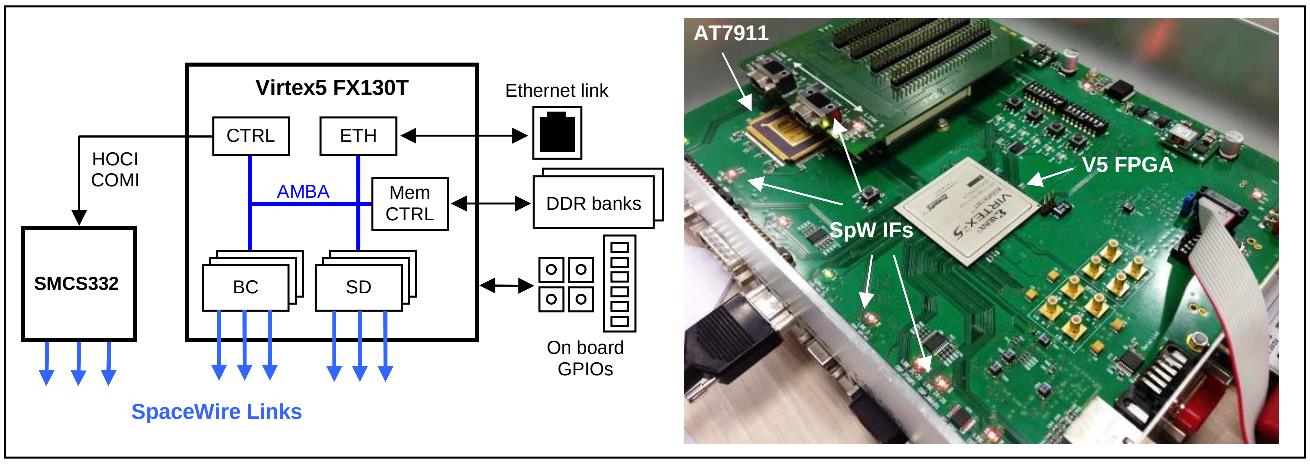


Fig. 2. SpaceWire Characterization Platform diagram and board overview. (Bc: Bepi Colombo IP, SD: Star Dundee IP)

of the FPGA is based on an AMBA bus surrounded by all the necessary interface controllers and logic cores to govern all the elements present on the board.Atmel AT7911 (also known as SMCS332SpW): this device provides an interface between three SpaceWire links and a data processing node. The processor interfaces are straight connected to the V5 FPGA allowing controlling the device independently or via an external system (user PC).

- External interfaces: Ethernet and serial links are available to access the system via an external controller, typically, a common personal computer. A simple PC application has been developed to provide a graphic user interface to control every SpaceWire link and execute automated tests.
- On board memory: a total amount of 512MB DDR SDRAM memory is available and accessible to the AMBA bus within the FPGA.
- General purpose IOs: several interfacing elements are included on the board to interact with the evaluation system. Also a mezzanine board has been developed to access all the FPGA spare pins allowing the expansion of the system for future implementations.
- External/Internal LVDS drivers: Another point to be evaluated on the frame of the project was the usage of the internal Virtex5 FPGA LVDS drivers to generate the differential data-strobe signals. Thus, the evaluation platform was designed to allow the selection between external and internal drivers for each link embedded on the device. This method allowed checking the correct operation of the FPGA drivers to implement the protocol without needing to add any external component.

IV. START DUNDEE AND BEPI COLOMBO IP CORES EVALUATION

Integrating the different SpaceWire IPs on the platform FPGA is the most flexible testing solution among all the different configurations that has been implemented. The possibility to configure every parameter with no restrictions allows performing a complete characterization of the link on different situations and conditions. Also, the FPGA configurability allows implementing further application layers attached to the interfaces to evaluate specific architectures (as example, to serve as simple EGSE).

Several instances of both IP cores have been embedded into the platform FPGA and a custom PC application has been developed to provide a user interface to control every link allowing their manual configuration or performing more complex automated tests. Due this flexibility different kinds of tests were performed:

A. Interoperability tests

These tests aim to check the correct interoperability between the links from the physical layer up to the packet level. A specific application running on the control PC allows creating custom transmission/reception tests with several different parameters:

- Link speed: transmission and reception speeds can be individually set for each link. Several tests covering the standard range from 2Mbps to 200Mbps were successfully executed.
- Data content: packets are completely customizable including its address value, the data content (random, constant, or ramp) and the size.
- Number of packets and delay: the number of packets composing the data flow is specified by the user as well as the transmission delay between packets.
- Number of iterations: dividing the tests into different iterations allows executing long tests without system overflow issues. For each iteration, the transmitted and received contents are compared and evaluated.

A large number of tests combining above parameters have been performed and no error has been detected on any case.

B. Robustness tests

The purpose of these tests is to evaluate the robustness of the links when subjected to hot resets. Once the link is established, an automated burst of resets is applied on one side of the link with a configurable interval to check the reconnection on the different status phases of the protocol.

TABLE I. ROBUSTNESS TESTS PARAMETERS

Resets	Interval	Protocol state
50	1 us	Error Reset
50	10 us	Error Wait
50	25 us	Run
50	1000 us	Run (with data flow)

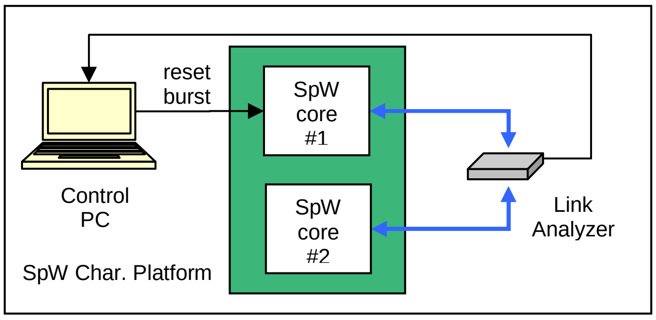


Fig. 3. Robustness test configuration

The external link analyzer not only has been used to check the correct reconnection process but to inject disconnect errors over the established link, providing another way to test the robustness of the link.

As result, no reconnection problems were detected on any of the executed tests. The reconnection timings checked with the link analyzer were coherent with the standard on every case.

C. External/Internal drivers evaluation

The usage of external or internal drivers to implement the signal layer made no difference in terms of interoperability for every link. This allows the system designers to choose the most appropriate solution for their architectures, either decreasing the external component population or the FPGA pins utilization.

However, a deeper characterization of the electrical layer comparing both solutions is planned.

V. AT7911 EVALUATION

The inclusion of the ASIC on the Characterization Board allows testing its interoperability with the rest of SpW links included on the board and with any other external equipment implementing the protocol.

The device can be supplied with 5V or 3V3, limiting the maximum transmission speed to 100Mbps on the second case. For the Characterization Platform, the 3V3 configuration was used, allowing the following interconnections:

TABLE II. AT7911 INTEROPERABILITY

	SMCS332SpW
Star Dundee IP	3.125 Mbps to 100 Mbps
Bepi Colombo IP	3.125 Mbps to 100 Mbps

Transmission and reception tests were executed successfully. Also, all the robustness tests performing hot resets on the IPs side result on successful reconnections by the SMCS332SpW device.

VI. GODDARD SPW IP INTEROPERABILITY

The ITAR condition of the Goddard SpaceWire IP used on the Iridium equipments did not allow its inclusion into the SpaceWire Characterization Platform. Thus, the interoperability tests were performed directly on an engineering model (EM) of real equipment belonging to the Iridium satellites payload on nominal system conditions (i.e. fixed configuration parameters and fixed data transmissions). Next matrix summarizes the connection conditions:

TABLE III. GODDARD SPW IP EVALUATION

		Payload EM
		Goddard IP
SpW Characterization Platform	Star Dundee IP	10Mpbs
	Bepi Colombo IP	10Mpbs
	SMCS332SpW	10Mpbs

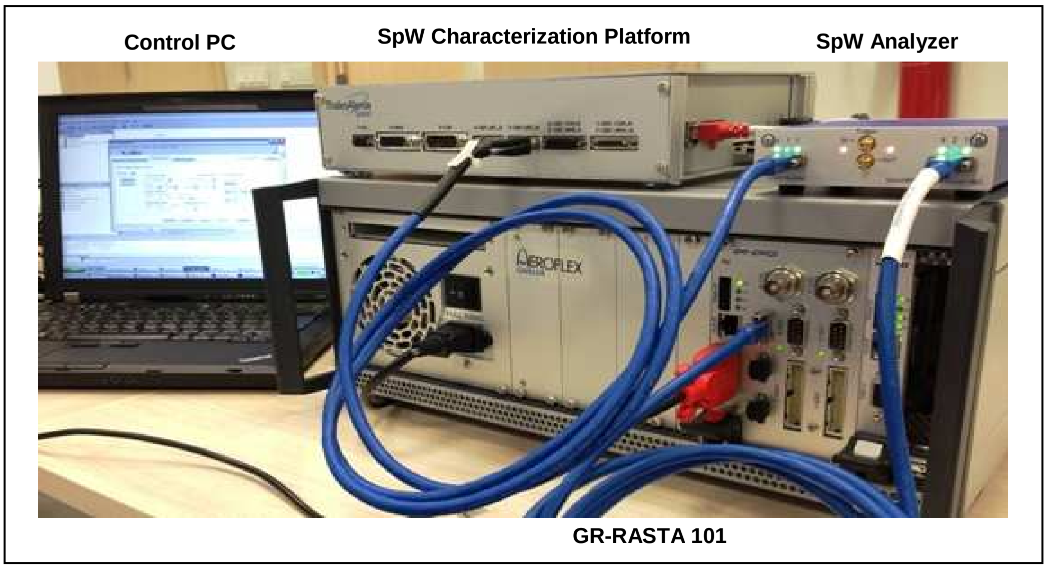


Fig. 4. GRSPW evaluation within the GR-RASTA platform

The nominal operation of the equipment involves a bidirectional data flow of 256 Bytes packets. Assuming that this test does not provides a complete characterization of the link, the successful synchronization and data transmission can be considered enough to accept the correct interoperability between the American IP and the European ones.

VII. GRSPW Interoperability

GR-RASTA development system has been used to evaluate the interoperability between Gaisler's GRSPW cores and the SpaceWire links included on TAS-E's SpW Characterization Platform. The 101 configuration includes a Virtex4 IO board providing, among others interfaces, three separate SpW links.

With this arquitecture following interconnections have been evaluated:

TABLE IV. GRSPW interoperability tests

		GR-RASTA 101
		GRSPW IP
SpW Characterization Platform	Star Dundee IP	2 Mbps to 200 Mbps
	Bepi Colombo IP	2 Mbps to 200 Mbps
	SMCS332	3.125Mpbs to 100Mbps

VIII. Conclusions

Even known this project was born to serve as support for a specific external project, TAS-E has made an effort to expand the test cases and the elements evaluated to create a reference to be used in the future by system designers expecting to integrate equipments with different SpaceWire components composing their interfaces.

As final conclusion, a succesfull interoperability between all the elements evaluated can be assumed. The SpaceWire Evaluation Platform has result a suitable evaluation system for VHDL SpaceWire components and a flexible platform to test future cores or protocols based on this kind of interfaces.

References

[1] ECSS-E-ST-50-12C (SpaceWire – Links, nodes, routers and networks, Issue 2, 31 July 2008).

[2] AT7911E ASIC, www.atmel.com/devices/at7911e.aspx

[3] GR-RASTA-LEON2/3/4 spacecraft avionics development platform, www.gaisler.com/index.php/products/systems/gr-rasta

MOST: MODELING OF SPACEWIRE TRAFFIC

SpaceWire test and verification, Short Paper

Brice Dellandrea
Command Control & Data Handling Engineer
Thales Alenia Space
France
Brice.Dellandrea@thalesaleniaspace.com

David Jameux
On-Board Data Systems Division (TEC-ED)
European Space Agency
Netherlands
David.Jameux@esa.int

***Abstract*—MOST (Modeling of SpaceWire Traffic) is a representative and powerful SpaceWire traffic simulator designed to support conception, development and validation of SpaceWire networks. Its recent improvements have targeted simplification and performance enhancement while still being used for sizing the SpaceWire networks of multiple TAS missions. This presentation will focus on its current capabilities and how they were employed on real use-cases then will present the new improvements brought to MOST.**

With the increasing complexity of SpaceWire networks embedded on board satellites and the development of SpaceWire standards and components, this simulator tool proves itself more and more useful.

***Index Terms*—SpaceWire Networks, Simulation, OPNET, MOST, Design, Traffic analysis, Performance assessment, Failure injection, FDIR, Protocol testing**

I. INTRODUCTION

MOST offers the possibility to build SpW network models, selecting and configuring SpW components, simulating high-level applications (FDIR for instance) and to test designs without waiting for HW testing on Avionics benches:

- It allows keeping control on traffic load and identifying weak parts of the network topology,
- It gives load margins and traffic performances (end-to-end delays, buffers sizing),
- It simulates many SpaceWire failure cases and gives the possibility to run various FDIR scenarios,
- It allows decreasing design risks and securing planning thanks to early verification,
- It allows testing the impact of change of Node or Switch behavior to help assessing the criticality of a supplier's non-compliance which can occur during any satellite development phases.

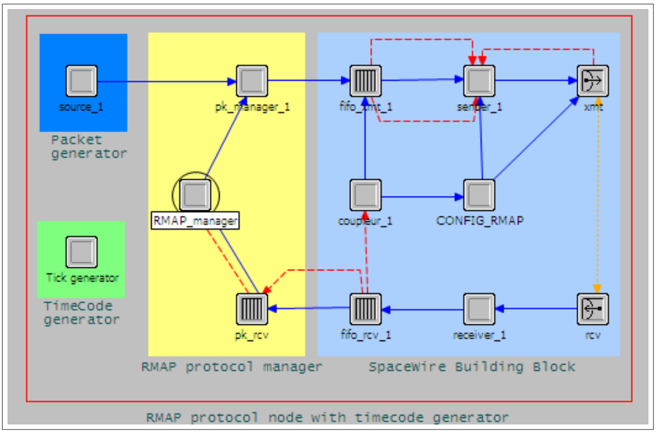
MOST has been developed as a library of OPNET Modeler® (Open NETwork modeler) in version 16.0. This object-oriented software allows SpW devices configuration thanks to a set of attached attributes. Its graphical editor provides a full set of possibilities to display and analyze simulation outputs.

Two versions of MOST currently exist: one early version which has been intensively used by Thales Alenia Space to make its internal simulations and including a wide SpaceWire components library: MOST v1.4. Another version is currently under development and brings many enhancements: MOST v2.2. Both versions will be presented in this document with a focus on the latter.

II. FROM MOST v1.4 TO MOST v2.2

MOST is the result of continuous developments efforts performed since 2006: first as internal Thales Alenia Space development, then with support of ESA to bring MOST to an operable stage through SpaceWire library development, validation with representative test cases (scientific mission and robotic mission), and finally cross-validated with real hardware.

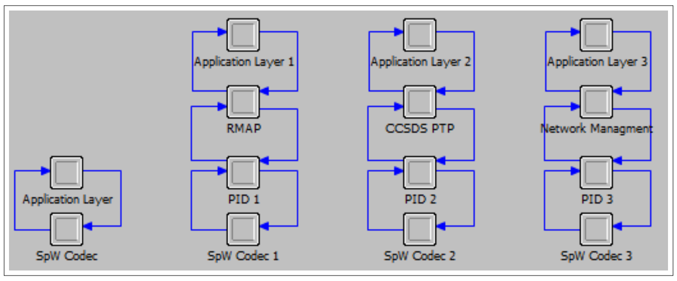
The progressive stages of development of MOST v1.4 followed the protocol stack of the SpaceWire standards: Physical level, Character level, Packet/Network level then User layer for SpW (PID, RMAP, PTP,…) and was internally developed accordingly as depicted in the following figure:



This brings a highly modular structure managed through finite-state automates which was very convenient during the initial SpaceWire CODEC development phase at the cost of additional OPNET processing time for automates processing. This first development generated MOST v1.4 which was delivered to ESA by end of 2011 and was used, for instance, by ESA to analyze the Bepi-Colombo SpaceWire-based payload Command & Control network and by TAS for the MTG SpaceWire network along with other TAS missions.

MOST v1.4 includes components developed according to some specific SpW component datasheets: the SpW-10X switch with GAR mechanism and round robin for priority management, the SMCS116SpW, the SMCS332SpW, the RTC. This MOST library has also been enriched with generic nodes fully representative of the SpW standard and including protocols building blocks such as RMAP, STUP and CPTP.

In parallel to the TAS simulation activities and continuous MOST v1.4 improvements with the addition of many FDIR functionalities (including dynamic reconfiguration of switching table through RMAP messages), ESA started the development of an experimental branch of MOST (v2.1). This development aimed at improving the usability of the tool by merging the physical, character and packet/network layers in a single SpW CODEC layer and at allowing the construction of protocol stacks as depicted in the following figure:



This new architecture had the advantage of providing a clearer view on the atomicity of the SpaceWire elements: one SpaceWire CODEC including the Physical, Character and Packet/Network levels developed in C-code instead of automates, then PID, RMAP or CPTP elements connected together through OPNET Modeler® exchange links. MOST v2.2 currently under development by TAS took as input this new architecture with the aim to, keep the advantages of the ESA solution while adding all features from MOST v1.4.

III. MOST v2.2 DESCRIPTION

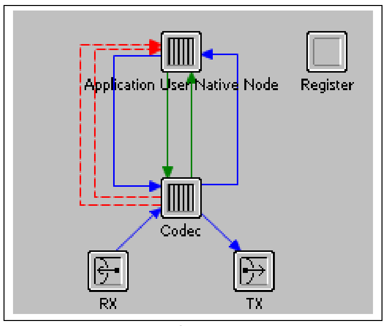
MOST v2.2 targets a release to the SpaceWire community. In that respect, the library has to be robust and easy enough to be used not only by its developers. This has been at the heart of the development undergone since end of 2012.

First of all, this new MOST library includes some new highly generic components to allow designing networks with very generic behaviors. Each of these generic components is made of multiple building blocks and shares a common building block with all the others: the SpW CODEC which fully complies with the ECSS-E-ST-50-12C SpaceWire. This CODEC building block can be tuned by the MOST user changing a set of parameters available in the OPNET Modeler® user interface:

- Link Enabled to enable/disable a port,
- Autostart is used to enter in Ready mode, node does not send Null characters but waits for Null characters to switch to Started mode,
- Link Start is used to enter in Started mode (sends Null characters). This attributes should be set to disabled if Autostart is set to enabled but MOST accepts both,
- TX Data Rate: this value corresponds to the physical transmission rate of the SpaceWire link,
- RX Buffer Size, this value represents the size of the CODEC reception buffer used to send the right number of FCT,
- Show NULL Messages allows to display NULL messages on SpaceWire link,
- Timer Disconnect is the time at which the CODEC disconnects its SpaceWire link (FDIR),
- Timer Parity Error is the time at which the CODEC simulates a parity error on the next characater received. This Error causes a Disconnect of the SpaceWire link (FDIR),
- Delay For Disconnection After Parity Error is the time between the detection of a Parity Error and the disconnection of the SpaceWire link (FDIR),
- Debug Level with 4 different values corresponding to the level of details on the internal CODEC behavior to print in the Console

Three components are currently implemented in MOST v2.2: a Native Node, a Generic CPTP & RMAP Node, and a Generic Switch.

The Native Node is a very simple component, implementing a SpW CODEC and a generic User Application.



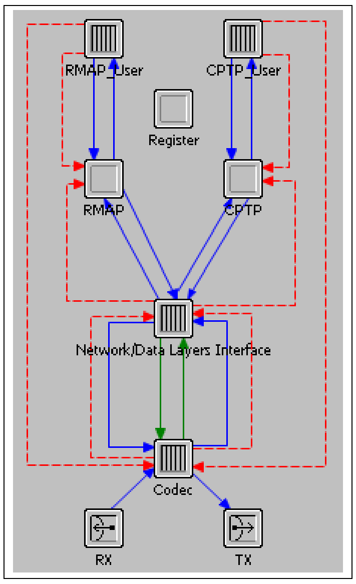
The User Application of the Native Node manages the SpW CODEC as a packet handling level which can be seen as a higher level from the ECSS-E-ST-50-12C point of view. It handles the TICK_IN & TICK_OUT interfaces and the exchange of bytes with the CODEC. It provides packet management: packet generation and packet reception. To support this feature, it implements an input buffer to assemble the bytes received from the CODEC before using it, and an output buffer to send byte per byte their content to the CODEC. It is configurable through the following parameters:

- Timecode Master (Enabled/Disabled) defines if the Node can issue Time-Codes,
- Timecode Interarrival Time is the period of Time-Codes emission,
- Time Code Start / Stop Time is a time defining when the Time-Code service is enabled (respectively beginning and end),
- Debug Level with 4 different values corresponding to the level of details on the internal Native Node User Application behavior to print in the Console,
- Packet Type is an integer between 0 and 98, this parameter allows to identify the packets sent by the

Node so that OPNET can compute its specific End-To-End Delay,

- Cargo Size is the size of the packets sent by the application to the CODEC,
- SpW Packet Interarrival Time is a waiting time between each packet sent by the Node,
- SpW Destination Address defined the destination of the packet. MOST accepts logical and physical addressing,
- Packet Generator Start / Stop Time is a time defining when the Packet generation service is enabled (respectively beginning and end),
- SpW Packet Deadline is the maximum time a packet can take to cross the network from its source to destination. The destination Node computes the real end-to-end delay and in case it is higher than the specified value, generates an error in the OPNET console.

The figure here-below provides an overview of the Native Node MOST architecture with a Register for Time-Count storage:



As it can be seen, the Generic CPTP & RMAP Node is more complex; it includes the implementation of PID protocol (ECSS-E-ST-50-51C), RMAP (ECSS-E-ST-50-52C), CPTP (ECSS-E-ST-50-53C) and a generic User Application on-top of each of the CPTP and RMAP protocol layers:

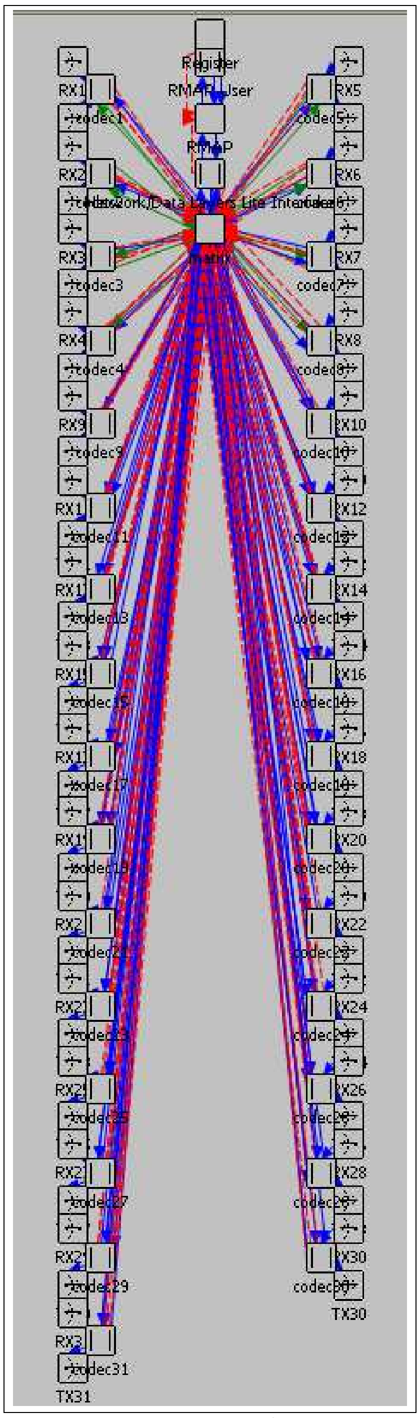
The Network/Data Layers Interface is a layer introduced for direct management of the SpW CODEC interfaces including time-codes handling and exchange of bytes with the upper-layers, it also performs the PID check on the received packets to route the packet to the relevant upper-layer protocol between RMAP and CPTP. To do so, this layer implements on the top-down direction an output buffer to assemble the packets from the data received from the CPTP and the RMAP blocks before sending their bytes one by one with a FIFO process and, on the bottom-up direction, transfers the bytes from the CODEC to one of the protocol layers depending on the PID information. CPTP and RMAP modules implement the protocol part of each standard: packet formatting according to each protocols. Their respective User Applications implement a reception buffer and manage the actions related to the packet content analysis.

For the Native Node as well as for the Generic CPTP & RMAP Node, the embedded User Application basically perform packet sending and consumption whatever their actual content. The RMAP User Application is more advanced; it allows sending a request which is pre-configured by the MOST user (allowing for instance to configure a switch configuration table). The format of this request is checked at receiver level to determine its effect (for example: READ, READ-WRITE, READ-MODIFY-WRITE), or simply discard in case of invalid request.

The Generic CPTP & RMAP node can be configured through a similar set of parameters than the Native Node with some additional features:

- Network/Data Layers Interface specificities:
  - NDLI Emission Buffer Size is the size of the packet emission buffer,
  - NDLI Local address is the logical address of a Node, this value shall be comprised between 32 and 254. It is optionally used to check the received packets address and discard invalid packets,
  - NDLI Local address Check, this value enables/disables the address check,
- CPTP & CPTP User Layers specificities:
  - CPTP packet EEP Status allows to end a packet with EEP (value = 1), by default all packets are ended with EOP,
  - CPTP Elephant Message Size: this parameter defined the size of an elephant packet,
  - CPTP Elephant Message Destination Address,
  - CPTP Elephant Message Start Time is a time defining when the elephant message is sent,
  - CPTP Reception Buffer Size: this size shall be greater than the biggest packet received,
  - CPTP Service Rate sets the rate at which packets are destroyed by the application
- RMAP & RMAP User Layers specificities:
  - RMAP Command Value is the content of a RMAP command to be transmitted,
  - RMAP Service Rate sets the rate at which packets are destroyed by the RMAP application,
  - RMAP Key is the value of local key for compare with RMAP request,
  - RMAP Reception Buffer Size: this size shall be greater than the biggest packet received,
  - RMAP Reply Delay is the delay between the reception of a request and the creation of its reply,
  - RMAP Local Address is the local address used to send a reply and shall be set between 32 and 254,
  - RMAP Reply Packet Type is the packet type use for reply packet, this value shall be an integer between 0 and 98

The Generic Switch is a 32-port (31 external ports, plus 1 connected to a configuration port) Switch configurable either in Static Mode (no reconfiguration on the initial table) or in Dynamic Mode (taking into account RMAP messages to change the routing table). It is able to perform Group Adaptative Routing and message priority management.



As it can be seen on this figure, the Generic Switch includes 31 SpW CODEC connected through a switching matrix that implements the SpW Network level. This matrix can switch packets from a port to another including configuration packets to be handled by a local RMAP User able to receive and interpret the requests and to reconfigure the matrix dynamically.

The routing switch building blocks can be configured through the following additional set of parameters:

- Watchdog Timer (Enabled/Disabled): protects the network of elephant messages. If the time taken to transmit a message is higher than the timeout value, the packet will be destroyed automatically in the switch
- Timeout of watchdog timer,
- Switching Table to configure the Switch (including header deletion capability, priority and one or more output ports per logical address for GAR).

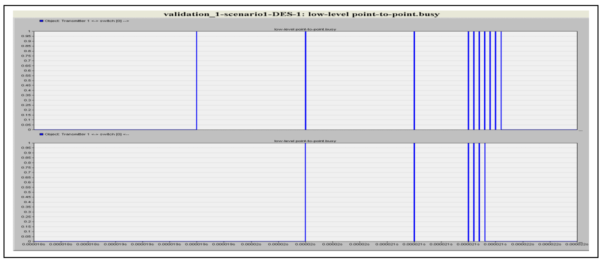
Apart from the RMAP reconfiguration requests, the RMAP & CPTP User Applications implement currently very generic behaviors based on data generation (with selection of size, address, periodicity, emission buffer sizing) and data consumption (reception buffer sizing, application service rate). These basic settings can be refined through the use of parameter files (“gdf” files) which allows configuring for instance a non-periodic data generation sequence or a packets sequence of different lengths, with different destination addresses, ended with EEP/EOP, etc... However, no special action is taken pending on the packet content. This kind of behavior is related to upper-level application (PUS for instance) and is currently not implemented in MOST v2.2. However, a MOST user can implement such functions in C-code in the RMAP User or CPTP User applications. This has already been done successfully by TAS in the frame of multiple simulations.

IV. SOME OF THE MOST v2.2 FEATURES

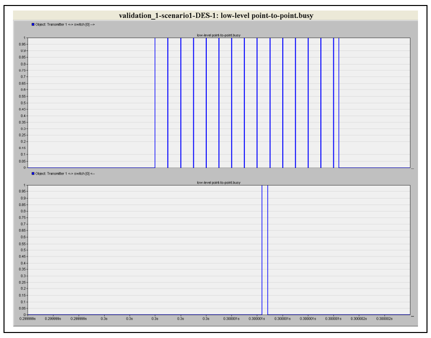
MOST v2.2 aims at providing full visibility on SpaceWire networks behaviour and provides many ways to configure them. Addressing can be either physical or logical, priority can be provided as per ECSS-E-ST-50-12C standard as well as header deletion. Address check can be performed optionaly at receiver level. Moreover GAR and dynamically configurable switching tables are implemented in the Generic 32-port switch.

MOST v2.2 allows configuring the links data rates; the CODECs reception buffer sizes, the packet emission and reception buffers. This has proved very useful to test the effect of SpaceWire network congestions over applications and the possible loss of packets due to emission buffers saturation. As the application does not necessarily consumes incoming packets at the speed of its underlying SpW CODEC, a service rate has been put in place to simulate the actual data consumption capability and provide better representativity for network sizing. For instance a Payload Data Hanling Unit receiving science packets from a high speed SpaceWire network and sending it over a lower speed Radio-Frequency Unit or Mass Memory with an intermediate buffering might block the network in case of reception buffer saturation.

Initialisation of the SpaceWire network is also taken into account with the simulation of exchange of Null messages, and implementation of LinkStart, AutoStart and LinkEnable flags, triggering the corresponding intialisation sequence with the final sending of the Flow Control Tokens:

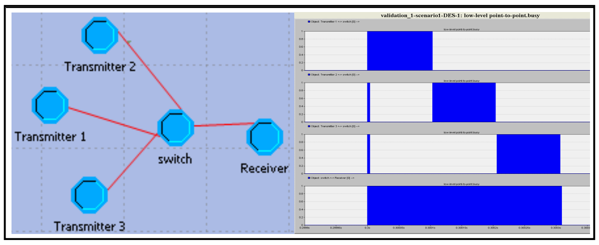


FCT are implemented and exchanged according to the received data characters flow. We can see here-below that on the sending direction of the link, a small packet is emitted with data characters and an EOP (the "small" ending character), while on the reception direction of the link, a FCT is emitted after 8 data characters have been sent:



NULL characters are also taken into account, providing realistic time-code propagation jitters and time spacing between symbols. These NULL characters can either been showed or masked in the simulation with the drawback to lengthen the simulation time and results processing as the links appears very busy (constant oscillations on the link). Their effect is anyway taken into account to compute the sending time of the emitted characters so masking them does not impair the representativity of the simulation.

Wormhole routing and the corresponding switching ports blocking until end of packet transmission is an important aspect of SpaceWire and can be analysed in details using MOST:



We can see on the previous figure the effect of congestion on three nodes willing to send packets to a single "Receiver": some data characters are stored in the input buffers of the routing switch then, when saturated, the communication is blocked until the emission port is free.

FDIR is a major feature for avionics and data handling engineers. In that respect, many events can be triggered in MOST, from parity bit error to EEP insertion, elephant packet generation, or spontaneous disconnections.

At last, MOST v2.2 has a clear implementation of the SpaceWire protocol stack, providing high flexility to the design of SpaceWire networks through the delivery of generic nodes and switches including basic applications with possible insertion of user-made C-code in identified areas of the generic User Application code for more advance behavior implementation (PUS or instrument HKTM packets generation pending on the reception of a special TC).

It is also possible to design user-made SpaceWire components through the development of specific assemblies (for instance a single User Application with multiple underlying CODECs) through modifications of the generic components using the OPNET interface (for more advanced users). This is optimized through the possible re-use of the already developed building blocks and has been performed by TAS in multiple occasions (for Virtual Channel Multiplexing machines, simulation of Masss Memory behaviors, SpaceWire couplers with different buffering schemes, etc...).

This library is scheduled to be enriched in the future with specific components existing on the market, for instance the 10X switch, the RTC, SMCS116SpW and SMCS332SpW as it was the case in the former MOST v1.4 version. Other implementations could be foreseen on a case-cy-case basis.

V. CONCLUSION

MOST aims at enriching OPNET Modeler® with an operational SpaceWire library available to the SpaceWire community. The recent development performed in the frame of an ESA contract extension brings MOST a major step closer to this objective. Simplification and modularity enhancement to facilitate the design of networks and new protocols are the key features driving the current developments.

ESA own the MOST IPR and intend to release MOST 2.2 to the SpaceWire community in 2013. Some maintenance of MOST 2.2 and improvements will be performed for ESA by Thales Alenia Space until October 2014.

VI. REFERENCES

[1] "SpaceWire – Links, Nodes, Routers and Networks", ECSS-E-ST-50-12C, 31st July 2008
[2] "SpaceWire protocol identification", ECSS-E-ST-50-51C, 5th February 2010
[3] "SpaceWire – Remote memory access protocol", ECSS-E-ST-50-52C, 5th February 2010
[4] "SpaceWire – CCSDS packet transfer protocol", ECSS-E-ST-50-53C, 5th February 2010
[5] ISC 2010 – "Simulation of SpaceWire Network" – Thales Alenia Space - France
[6] "Modeling of SpaceWire Traffic" DASIA 2011 & 2012 – Thales Alenia Space - France
[7] "MOST User Manual" 100435074I, 8th March 2013 – Thales Alenia Space - France

SpaceWire EGSE: Real-Time Instrument Simulation in a Day

SpaceWire Test and Verification, Short Paper

Stephen Mudie
STAR-Dundee Ltd
Dundee, Scotland, UK
stephen.mudie@star-dundee.com

Martin Dunstan, Steve Parkes
School of Computing
University of Dundee
Dundee, Scotland, UK
mdunstan@computing.dundee.sc.uk,
sparkes@computing.dundee.ac.uk

The SpaceWire Electronic Ground Support Equipment (EGSE) is a test and development unit produced by STAR-Dundee, which simulates instruments or other SpaceWire equipment in real-time. The SpaceWire EGSE is configured using a simple yet powerful scripting language designed specifically for SpaceWire applications. A script is compiled and loaded into the EGSE unit using software. Once configured the SpaceWire EGSE operates independent of software and therefore exhibits real-time behavior. The capabilities of the scripting language combined with the real-time operation of the hardware make it possible to rapidly mimic real-time behaviour of SpaceWire equipment, vastly reducing traditional development time and cost associated with writing equivalent software in a real-time operating system. The SpaceWire EGSE can generate detailed packets in pre-defined sequences at specific times and data rates, controlled by state machines and events. To integrate with external equipment it has three external input triggers and one external output trigger. For additional control over the SpaceWire EGSE, a software API is provided that, amongst other things, can provide notifications of state changes and events.

***Index Terms*— Relevant indexing terms: SpaceWire, Networking, Spacecraft Electronics, Electronic Ground Support Equipment.**

I. Introduction

Design of SpaceWire enabled units often require other SpaceWire units to be simulated. For example, the development of a mass memory unit will require simulation of all the instruments that are sending it data. The simulation needs to run in real-time resulting in the need of SpaceWire interface devices, a computer running a real-time operating system, and bespoke real-time software development. The SpaceWire Electronic Ground Support Equipment (EGSE) aims to provide real-time simulation of SpaceWire equipment without the need for designing real-time software.

The SpaceWire EGSE is a test and development unit produced by STAR-Dundee. It is configured using a scripting language designed specifically for SpaceWire applications. The scripting language can be used to send packets in pre-defined sequences at specific times and data rates. Once a script is written in which the SpaceWire instrument simulation behavior is defined, it is compiled and loaded into the EGSE unit using software. When configured the SpaceWire EGSE operates independent of software resulting in real-time performance.

This paper briefly describes the SpaceWire EGSE hardware, software, scripting language, capabilities, benefits and the current known limitations.

II. SpaceWire EGSE Hardware

The SpaceWire EGSE hardware consists of two SpaceWire interfaces for transmitting and receiving SpaceWire traffic, four external triggers (three in, one out) for interfacing with external equipment, 128MB of memory in which packet definitions are stored, two mictor logic analyser connectors to show device state and a USB connection to the host PC. The status of the SpaceWire interfaces, external triggers and USB port are indicated by LEDs.

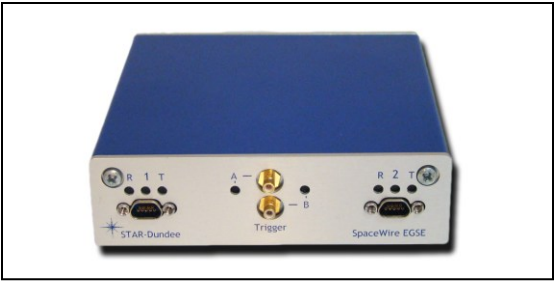
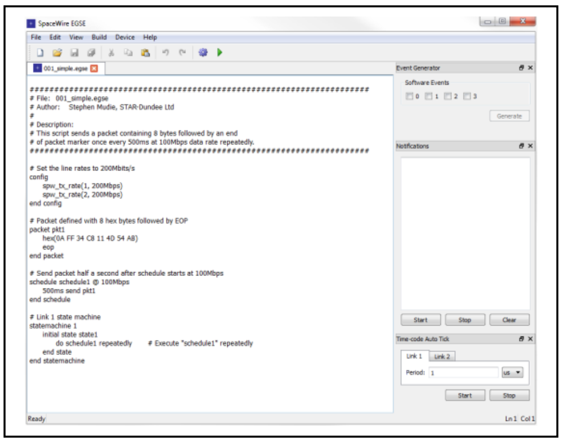


Fig. 1. SpaceWire EGSE Front Panel

III. SpaceWire EGSE Software

The SpaceWire EGSE software consists of four components: a compiler, loader, C API (Application Programming Interface) and GUI (Graphical User Interface). The compiler is a command line application that is used to compile EGSE scripts to configuration files. The loader is another command line application that is mainly responsible for configuring a SpaceWire EGSE unit using a configuration file. The C API software allows users to write their own code to interact with a SpaceWire EGSE unit. The GUI application combines a text editor, for creating and modifying EGSE scripts, with much of the functionality provided by the compiler, loader and C API including: script compilation, EGSE configuration, software event generation, state and event notification monitoring and periodic time-code generation.



```
##########################################################################
# File:   001_simple.egse
# Author:    Stephen Mudie, STAR-Dundee Ltd
#
# Description:
# This script sends a packet containing 8 bytes followed by an end
# of packet marker once every 500ms at 100Mbps data rate repeatedly.
##########################################################################

# Set the line rates to 200Mbits/s
config
    spw_tx_rate(1, 200Mbps)
    spw_tx_rate(2, 200Mbps)
end config

# Packet defined with 8 hex bytes followed by EOP
packet pkt1
    hex(0A FF 34 C8 11 4D 54 AB)
    eop
end packet

# Send packet half a second after schedule starts at 100Mbps
schedule schedule1 @ 100Mbps
    500ms send pkt1
end schedule

# Link 1 state machine
statemachine 1
    initial state state1
        do schedule1 repeatedly      # Execute "schedule1" repeatedly
    end state
end statemachine
```

Fig. 2. SpaceWire EGSE GUI

IV. SpaceWire EGSE Scripting Language

The SpaceWire EGSE is configured using a simple yet powerful scripting language that was designed specifically for SpaceWire applications. The scripting language can be used to send packets in pre-defined sequences at specific times and data rates. Dynamic packets are defined using packet and variable definitions. The sequence, timing and data rate at which packets are transmitted are defined in schedules. The current executed schedule is controlled by state machines and events.

```
# Set the line rates to 200Mbit/s
config
        spw_tx_rate(1, 200Mbps)
        spw_tx_rate(2, 200Mbps)
end config

# Packet defined with 8 hex
# bytes followed by EOP
packet pkt1
        hex(0A FF 34 C8 11 4D 54 AB)
        eop
end packet

# Send "pkt1" 0.5s after schedule
# starts at 100Mbits/s
schedule schedule1 @ 100Mbps
        500ms send pkt1
end schedule

# SpW link 1 state machine
statemachine 1
        # State in which "schedule1"
        # is executed repeatedly
        state state1
                do schedule1 repeatedly
        end state
end statemachine
```

Fig. 3. Simple SpaceWire EGSE Script

A. Packet Definitions

Packet definitions can consist of data defined in hexadecimal or decimal bytes, data imported from file, variable references, CRC and checksum calculations, EEP and EOP markers and time-code manager instructions.

B. Variables

Variables are used to define packets with dynamic data. Declared variables can be referenced in packet definitions. The value produced by a variable reference is dependent on its type:

- Constant: Value remains the same each time it is referenced.
- Random: Random value each time it is referenced.
- Increment: Value is incremented by one each time it is referenced.
- Decrement: Value is decremented by one each time it is referenced.
- Rotate right: Bitwise rotate right is performed on the variable each time it is referenced.
- Rotate left: Bitwise rotate left is performed on the variable each time it is referenced.
- CRC: Used to perform automatic CRC calculations. This is the RMAP CRC.
- Checksum: Used to perform automatic checksum calculations.

C. Schedules

Schedules are used to send pre-defined packets at specific times and data rates. The timing of packet transmission can be relative to the start of the schedule or relative to the start of the previous packet transmission.

D. State Machines

State machines are used to control the SpaceWire EGSE state. One state machine is defined per SpaceWire interface. Each state in a state machine is associated with a schedule which is executed when that state is entered. State transition statements specify the event(s) on which to transition from one state to another.

E. Events

Events are used to control the current state of the SpaceWire EGSE state machines and therefore the current packet generation schedule. The different event types are:

- Software: Transition from one state to another in response to events generated from host software.
- State machine: Raise an event when a state of interest is entered.
- External trigger in: React to an external input trigger signal received from other equipment.
- External trigger out: Generate an external trigger output signal in response to an event of interest.
- Time-code received: React to the receipt of a time-code on a SpaceWire interface.

- Time-code transmitted: React to the transmission of a time-code from a SpaceWire interface.
- Received pattern matched: Transition from one state to another when the SpaceWire traffic received on an interface matches a specified pattern.
- Link error: React to a link error detected on a SpaceWire interface.

V. SpaceWire EGSE Example Script

The following example script briefly illustrates some of the SpaceWire EGSE scripting language (please note that lines starting with '#' are comments). Assume there is a requirement to simulate a camera that sends two images with a 100ms gap between each at a data rate of 100Mbits/s. Please note that two images are used in this example to keep the script size sensible but this could easily be modified to handle more images.

```
# Set the line rate of
# link 1 to 200Mbits/s
config
        spw_tx_rate(1, 200Mbps)
end config

# Define events
events
        # Software in event 0
        swEvent0 = software_in(0)
end events

# Define packet "image_001"
packet image_001
        file("image_001.ppm")
        eop
end packet

# Define packet "image_002"
packet image_002
        file("image_002.ppm")
        eop
end packet

# Define empty schedule "nothing"
schedule nothing
end schedule

# Define schedule "sendImages"
schedule sendImages @ 100Mbps
        100ms send image_001
        200ms send image_002
end schedule

# Define SpW link 1 state machine
statemachine 1
        # Define starting state "off"
        initial state off
                do nothing
                on swEvent0 goto sendImages
        end state

        # Define state "sendImages"
        state sendImages
                do sendImages
                goto off
        end state
end statemachine
```

Fig. 4. SpaceWire EGSE Camera Script

This example consists of a configuration block, an events block, two packet definitions, two schedules and a state machine. Within the configuration block is a command that sets the link speed of SpaceWire link 1 to 200Mbits/s. The events block contains a software in event named "swEvent0" associated with software event 0. A packet named "image_001" is defined containing the data held in the image file named "image_001.ppm" followed by an EOP. A packet named "image_002" is defined containing the data held in the image file named "image_002.ppm" followed by an EOP. A schedule named "sendImages" sends the two image packets, "image_001" 100ms after the schedule starts and "image_002" 200ms after the schedule starts at a data rate of 100Mbits/s. An empty schedule named "nothing" does nothing. A state machine for SpaceWire link 1 contains two states named "off" and "sendImages". "off" is the starting state and executes the schedule "nothing". When software event "swEvent0" is detected a transition to the state "sendImages" will occur. The state "sendImages" executes the schedule "sendImages" before transitioning to the "off" state.

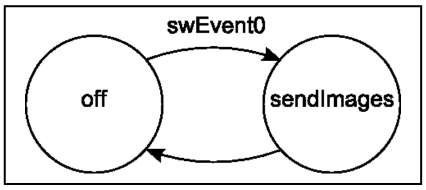
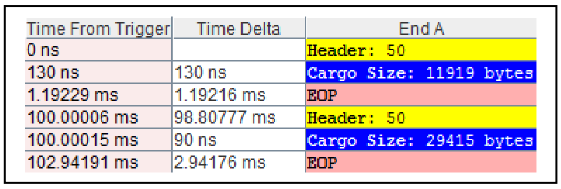


Fig. 5. Camera State Diagram

When this script is compiled and the resulting configuration file is loaded into the SpaceWire EGSE, initially nothing is transmitted. When software event "swEvent0" is detected a state transition occurs and two pre-defined images stored on disk are sent as single packets. These are transmitted at 100ms intervals at a data rate of 100Mbit/s. The state machine then transitions back to the "off" state where nothing is transmitted.



Time From Trigger	Time Delta	End A
0 ns		Header: 50
130 ns	130 ns	Cargo Size: 11919 bytes
1.19229 ms	1.19216 ms	EOP
100.00006 ms	98.80777 ms	Header: 50
100.00015 ms	90 ns	Cargo Size: 29415 bytes
102.94191 ms	2.94176 ms	EOP

Fig. 6. Link Analyser Mk2 Capture of Camera Output

VI. Capabilities and Benefits

Thanks to the ability of the SpaceWire EGSE to operate independent of software and the scripting language used to configure it, using the SpaceWire EGSE it is possible to rapidly simulate SpaceWire instruments or other equipment in real-time. The SpaceWire EGSE can transmit pre-defined sequences of packets consisting of varying payloads with accurate timing at specific data rates. The 128MB of memory available to the EGSE, in which pre-defined packet data is stored, makes it possible to simulate large payloads from high data-rate instruments. The SpaceWire EGSE monitors the received SpaceWire traffic and can respond to a matched pattern, time-codes and link errors. To interface with external equipment the EGSE has one output trigger, which can generate a signal in response to a specific event, and three input triggers, that allow the EGSE to react to input signals. The SpaceWire EGSE supports time-code generation and can act as a time-code master. Notifications of state changes and events are sent from the EGSE unit to the host software. These can be used to monitor and debug the EGSE activity using the provided GUI. Alternatively, custom applications written using the provided C API can use these for their own purpose e.g. to integrate with other equipment which use an alternative to SpaceWire.

VII. Limitations

The SpaceWire EGSE scripting language is designed to meet the requirements of as many SpaceWire instruments and equipment as possible but there will be times where an instrument cannot be successfully simulated and traditional EGSE software development will be required. The SpaceWire EGSE has some limitations that we are aware of. One limitation is many of the structures used in the language have a finite limit. The number of variables, events, packets and states declared and schedule references each have a limit. Many of these limits far exceed the requirements of most simulations but there may be rare cases where these restrict the ability to simulate an instrument in its entirety. Another known limitation is the inability to retransmit the contents of a received packet. For example, using the pattern match received event it is possible to detect an RMAP read command received on a SpaceWire interface, however to transmit the read reply requires the transaction ID of the received read command. A solution to this problem has been identified and is expected to be implemented in the near future.

VIII. Conclusion

This paper has briefly described the SpaceWire EGSE, including the EGSE hardware, software and scripting language. To demonstrate some of the key concepts of the scripting language (link speed configuration, events, packet definitions, scheduling and state machines) an example script was shown. The capabilities, benefits and known limitations of the SpaceWire EGSE were then discussed.

The compact and powerful nature of the SpaceWire EGSE scripting language combined with the hardware's ability to operate independent of software and therefore in real-time, make it possible to simulate SpaceWire equipment in real-time in little more than a day. This rapid development time makes the SpaceWire EGSE an attractive alternative to traditional EGSE development.

References

[1] STAR-Dundee, SpaceWire EGSE User Manual, v1.03.

[2] STAR-Dundee, http://www.star-dundee.com/sites/default/files/SD_TN_006%20spw_egse_camera_simulation_v3.pdf, SpaceWire EGSE: Simulating a Camera, STAR-Dundee Website.

[3] STAR-Dundee, http://www.star-dundee.com/products/spacewire-egse, SpaceWire EGSE, STAR-Dundee Website

STAR Fire: SpaceFibre diagnostic interface and analyser

SpaceWire Test and Verification, Short Paper

Albert Ferrer Florit, Alberto G. Villafranca,
Chris McClements, Steve Parkes
STAR-Dundee Ltd
Dundee, UK
albert.ferrer@star-dundee.com, alberto.gonzalez@star-dundee.com, chris.mcclements@star-dundee.com,
steve.parkes@star-dundee.com

Abstract— SpaceFibre is a new technology for use onboard spacecraft that provides point-to-point and networked interconnections at Gigabit rates with Quality of Service. SpaceFibre carries SpaceWire packets over virtual channels and provides a broadcast capability similar to SpaceWire time-codes. In order to assist with the development, testing and validation of the first SpaceFibre system a SpaceFibre diagnostic interface and analyser unit, called STAR Fire, was built by STAR-Dundee.

This paper describes STAR Fire, the first complete test and development solution available for SpaceFibre. STAR Fire has two independent SpaceFibre interfaces compliant with the SpaceFibre standard, each one with an embedded link analyser and multiple very high data rate hardware data generators and checkers. The unit can be configured in interface or sniffer mode. The sniffer mode is used to monitor protocol and user data produced by an external unit passing in both directions along a SpaceFibre link, similar to the STAR-Dundee SpaceWire Link Analyser. The STAR Fire unit can also be used as a bridge between SpaceWire and SpaceFibre links, using an embedded router that interconnects some SpaceFibre virtual channels with the two SpaceWire ports provided.

These and other functionalities are easily configured using a Graphical User Interface software in the host PC. The user can supervise the status of the unit and set the parameters of each link, broadcast channel, virtual channel data rate, Quality of Service and error injection. The link analyser module decodes and shows the SpaceFibre protocol and user data stream which can be analysed at character, word or frame level.

STAR Fire has been designed to support the rapid and painless adoption of the SpaceFibre technology within the SpaceWire community.

***Index Terms*—SpaceFibre, SpaceWire, STAR Fire**

I. Introduction

SpaceFibre is a very high-speed serial link designed specifically for use onboard spacecraft and to be compatible with SpaceWire protocol [1]. The aim of SpaceFibre is to provide point-to-point and networked interconnections for Gigabit rate instruments, mass-memory units, processors and other equipment, on board a spacecraft. SpaceFibre is designed to be compatible with the SpaceWire protocol at packet level but providing a much higher data rate.

STAR-Dundee in collaboration with the University of Dundee has developed STAR Fire, a complete SpaceFibre diagnostic unit configured through a Graphical User Interface (GUI) which also provides status information and analysis capabilities. Hence, STAR Fire provides a complete SpaceFibre test and development solution.

STAR Fire hardware unit features two independent SpaceFibre interfaces compliant with the latest draft of SpaceFibre ECSS standard [2], each one with an embedded link analyser and multiple very high data rate hardware data generators and checkers. Furthermore, STAR Fire unit provides two SpaceWire ports and an embedded SpaceWire router. It also provides hardware triggering capabilities and the ability to access the data of the embedded analyser using two logic analyser MICTOR connectors or a PC, by using specific software. Additionally, STAR Fire unit allows user update through the USB port. In this way, it is easy for users to keep track of new developments and functionalities added to the design.

STAR Fire software is based on a GUI that allows the configuration of the SpaceFibre interfaces and the use of the embedded link analyser. It also controls the parameters of the data generators and monitors the status of the data checkers for virtual channels and broadcast data. Furthermore, there is a trigger module that decodes the SpaceFibre data stream which can be analysed using the word or the frame based view.

II. SpaceFibre Overview

SpaceFibre high-speed serial link carries SpaceWire packets over multiple channels, called virtual channels (VC), each one with a defined Quality of Service (QoS) and provides an improved broadcast mechanism similar to SpaceWire time-codes but offering much more capability. SpaceFibre has two types of user interfaces to send data. The VC interface comprises a number of virtual channel buffers for sending SpaceWire packets and the same number for receiving SpaceWire packets. SpaceFibre is compatible with the packet level of the SpaceWire standard. This means that applications developed for SpaceWire can be readily transferred to SpaceFibre. The broadcast interface is designed to send short messages of up to 8 bytes with very low latency across the network, in a similar manner as the SpaceWire time-codes, but providing not only timing distribution but also signalling and interrupt services. SpaceFibre currently operates at 10 times the maximum data-rate of SpaceWire – i.e. link speed of 2.5 Gbps – and can run over fibre optic (up to 100 m) or copper media (up to 8 m).

SpaceFibre provides a completely reliably link with the fastest possible error recovery time for transient and persistent errors. This is fulfilled with a retry mechanism that guarantees reliability in the communications link. This allows recovering from transients and persistent errors on the SpaceFibre link. The retry mechanism uses the following Fault Detection, Isolation and Recovery (FDIR) mechanisms:

- Notification of data or control information using positive and negative acknowledgements (ACKS/ NACKS)
- Error detection using sequence numbers, 8B10B error detection capabilities and CRC codes
- Automatic resending of data frames, broadcast frames and flow control tokens using a Go-Back-N scheme when sporadic errors occurs
- Automatic re-initialisation of the link when an error is persistent

In addition, SpaceFibre provides timely data delivery and determinism using a medium access controller that determines which channels can send data and in which order. The QoS is independently configurable for each VC. Three mechanisms can be configured and combined:

- Priority: provides less latency to virtual channels with higher priority
- Bandwidth allocation: provides a minimum guaranteed throughput
- Scheduling: provides deterministic packet delivery

These different QoS parameters work together in a consistent manner. Hence, it is possible to work at the same time with a VC that requires minimum latency for command and control operations, a VC with a guaranteed throughput for payload data, and a deterministic delivery for packets that need to be sent and processed in a specific order.

III. System Architecture

The STAR Fire hardware unit consists of two SpaceFibre interfaces (eSATA connectors), two SpaceWire interfaces (micro-miniature D-type connectors), four external triggers (SMB connectors, three input and one output) for interfacing with external equipment and two logic analyser (MICTOR connectors) interfaces. The status of the SpaceWire and SpaceFibre interfaces is notified by LEDs. The hardware design provides, in addition to the SpaceFibre and SpaceWire ports, multiple very high data rate data generators, data checkers, link analysers and an embedded SpaceWire router. The system architecture is shown in Fig. 1.

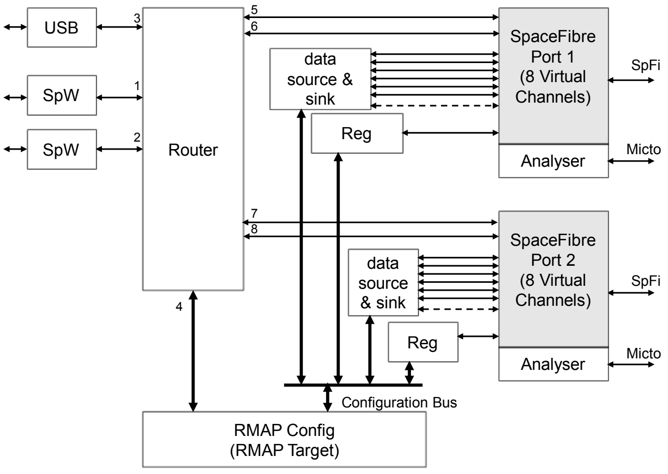


Figure 1 System Architecture

Each SpaceFibre port contains eight VCs that are arbitrated following the QoS requested. The SpaceWire interfaces and some SpaceFibre VCs are connected to an embedded SpaceWire router as shown in Figure 1. This allows SpaceWire packets from SpaceWire interfaces to go into SpaceFibre VCs and vice versa. However, in order to achieve the much higher data rate of SpaceFibre, the hardware data generators and checkers connected to virtual channels 2 to 7 can be used. Similarly, each SpaceFibre port also features a broadcast data checker and generator. Finally, an RMAP [3] target allows accessing to configuration and status registers of the SpaceFibre cores and the data generators and checkers. The RMAP target is accessed by the software through the router by a USB port but can also be accessed through the SpaceWire ports.

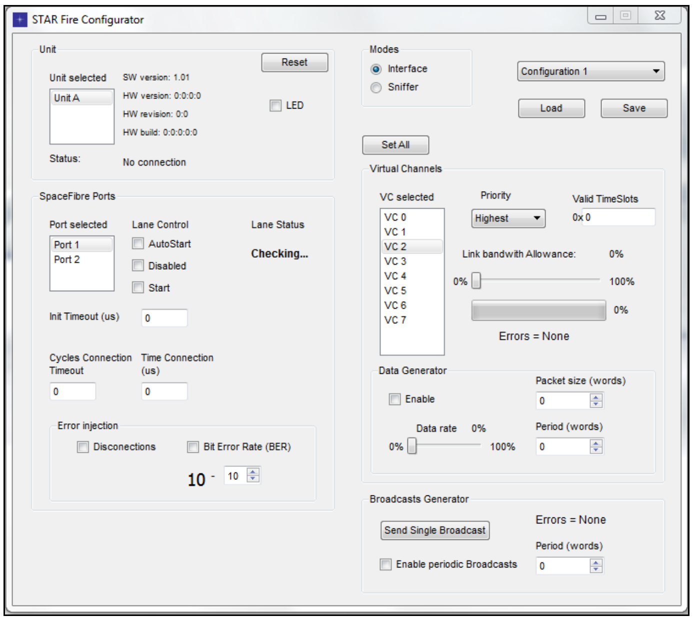
IV. STAR Fire Configurator SW and Trigger

STAR Fire also includes dedicated software developed to control and monitor the hardware unit. STAR Fire Configurator allows the configuration of the SpaceFibre interfaces and the use of the embedded link analyser. It also controls the parameters of the data generators and monitors the status of the data checkers for both VC and broadcast data. The software suite also includes STAR Fire Trigger. This Trigger module allows programming the trigger and decoding the SpaceFibre data stream. Two different display views for the decoded data are offered. Analysis of the data is possible using either Word or Frame based view.

Figure 2 presents a screenshot of the STAR Fire Configurator (top) and Trigger windows (bottom). There are different regions in the Configurator tool window which control the different parameters of the unit.

A. Unit

This region identifies the unit selected. Several units can be connected to the same computer and controlled from a single Configurator instance at the same time.



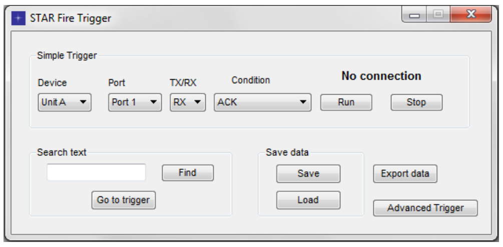


Figure 2 STAR Fire Configurator (top) and Trigger windows (bottom)

B. SpaceFibre Ports

The two SpaceFibre ports can be controlled in this region. The lane status of the port is displayed. The starting mode (Start or AutoStart) can be configured. Additionally, the initialisation timeout can be controlled for debugging purposes.

C. Modes

STAR Fire can be configured in sniffer or interface mode. When the sniffer mode is set the STAR Fire is only used to analyse the data stream of an external SpaceFibre capable unit. On the other hand, in interface mode each SpaceFibre port of the unit is a source and a destination of SpaceWire packets encapsulated in SpaceFibre frames. Besides, the whole setup of a unit can be saved and loaded into different files for faster configuration.

D. Virtual Channels

Each SpaceFibre interface has eight VCs. Independent data generators and checkers are connected to each of the input and output of VCs 2-7. The generation rate, packet size and working period can be controlled in this region. The generation rate specifies the duty cycle of the specified period. For example, if a 75% bandwidth and 100 word period are selected, the data generator will generate consecutive 75 words and remain idle for 25 words before starting to send data again.

VCs 0 and 1 are connected to an internal SpaceWire router and are used to transmit SpaceWire packets from the two SpaceWire ports.

QoS parameters (priority, bandwidth, scheduling) for each VC are also configured here. A bar shows the current generation rate of the selected channel and an error counter verifies that no errors are encountered in the data pattern.

E. Broadcasts Generator

Broadcast frames can also be generated. A single broadcast or periodic broadcasts can be sent through the SpaceFibre selected port with a configurable period. The data received is checked by a broadcast checker and the number of errors is displayed.

F. Error Injection

In addition to the status and control information, it is possible to automatically insert random disconnections on the selected link or a specific bit error rate in the form of a power of 10. This feature is useful when simulating persistent errors or bit flip conditions on the line.

G. Trigger

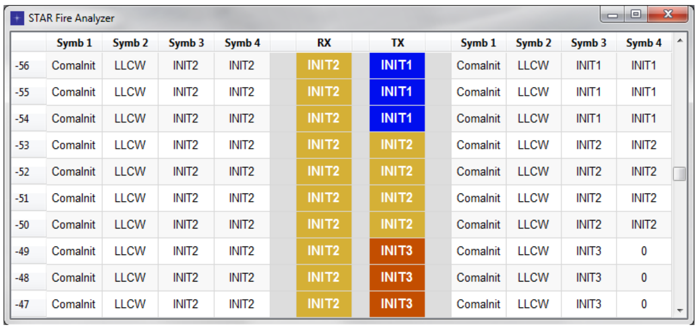
The Trigger window allows two operation modes. By default the Simple Trigger mode is shown. However, the Advanced Trigger mode can be selected in the GUI. This advanced mode shall only be used for debugging purposes or when using complex setups (e.g. several STAR Fire units connected to the same PC, using external trigger signals, cross-triggering between units, etc.). For the sake of simplicity only the Simple Trigger is shown.

When different units are connected to a PC, the Device drop-down list allows selecting the appropriate one. Port 1 and 2 can be selected for any unit, and also whether the trigger analyses the RX or TX side of the selected port. Finally, the condition that triggers the unit is selected in the Condition drop-down list. Any control word defined by SpaceFibre can be selected. The Advanced Trigger offers the possibility of triggering the unit not only on words but also on certain events, namely, disparity or not-in-table errors, data checker errors, etc. This can be useful for analysing specific situations during development. Furthermore, the Analyser will show by default the RX and TX sides of the selected Port. But the Advanced Trigger also offers the possibility of displaying the RX side of both ports of the selected Unit instead.

After setting up the trigger, the Run button activates it. The status of the trigger is continuously shown. When triggered, a Search Text box allows searching for particular strings in the captured data. The row in which the text is found is shown in the STAR Fire Analyser window. The data can also be saved or loaded, and exported to be analysed in other software if necessary.

V. STAR Fire Analyser

The STAR Fire Analyser shows two separate views which analyse data captured by the trigger or loaded from a data file (Fig. 3). The main window is called the Word Viewer (top panel of Fig. 3) and shows the SpaceFibre words received. The Word Viewer presents the analysis of data at word level. Each SpaceFibre word consists of four 8B10B symbols or characters [4]. In the central part of the window the word is decoded, sometimes complemented with some additional information such as the frame sequence number. Apart from this information, the four different symbols composing a word are also displayed in the external part of the window.



STAR Fire Analyzer

	Symb 1	Symb 2	Symb 3	Symb 4	RX	TX	Symb 1	Symb 2	Symb 3	Symb 4
-56	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT1	ComaInit	LLCW	INIT1	INIT1
-55	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT1	ComaInit	LLCW	INIT1	INIT1
-54	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT1	ComaInit	LLCW	INIT1	INIT1
-53	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT2	ComaInit	LLCW	INIT2	INIT2
-52	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT2	ComaInit	LLCW	INIT2	INIT2
-51	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT2	ComaInit	LLCW	INIT2	INIT2
-50	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT2	ComaInit	LLCW	INIT2	INIT2
-49	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT3	ComaInit	LLCW	INIT3	0
-48	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT3	ComaInit	LLCW	INIT3	0
-47	ComaInit	LLCW	INIT2	INIT2	INIT2	INIT3	ComaInit	LLCW	INIT3	0

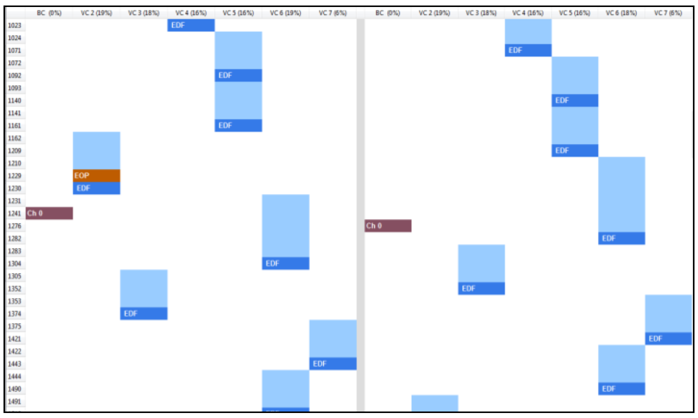


Figure 3 STAR Fire Analyser: Word Viewer (top) and Frame Viewer (bottom)

The other window is the Frame Viewer (bottom panel of Fig. 3) and shows a more compact analysis of the data. The received data and broadcast frames of each side is displayed, with a separate column for each VC. Note that it is possible to have broadcast frames in the middle of data frames, but at any particular time selected by a row there can only be one VC data frame on each side (as these frames are being multiplexed through a single SpaceFibre link). If there is an EOP within a data frame it is shown together with the number in bytes of the SpaceWire packet that the EOP terminates. This allows a quick inspection of the SpaceWire packets travelling through the VCs.

Finally, both viewers share the same row numbers. Thus, their view is automatically updated to always show the selected row in both windows.

VI. Conclusion

The STAR Fire diagnostic interface and analyser unit has been presented here. This unit can be easily configured through a Graphical User Interface providing complete status information and analysis capabilities. STAR Fire provides a complete SpaceFibre test and development solution. It features an internal SpaceWire router which allows connecting two SpaceWire interfaces to the SpaceFibre virtual channels. Furthermore, STAR Fire also contains embedded data generators and checkers which can individually operate up to 2 Gbps, broadcast generators, and error and link disconnections insertion. An embedded analyser allows triggering on certain events (e.g. errors or specific data words) and to display and store the captured data. These hardware capabilities are combined with a software package which provides a GUI to control STAR Fire operation and triggering, and also to access the analyser with byte, word and frame level views. All in all, STAR Fire is a flexible and powerful tool which provides support for adoption of the SpaceFibre technology.

References

[1] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire, Links, Nodes, Routers and Networks”, Issue 1, European Cooperation for Space Data Standardization, July 2008.

[2] S.M. Parkes, A. Ferrer, A. Gonzalez, & C. McClements, “SpaceFibre Standard Draft E1”, University of Dundee, 28th September 2012.

[3] ECSS, “SpaceWire - Remote memory access protocol”, ECSS-E-ST-50-52C, Feb 2010.

[4] A. X. Widmer and P. A. Franaszek, “A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code”, *IBM J. Res. Dev.*, Vol 27, Issue 5 (440-451), 1983.

Missions & Applications (Short)

SpaceWire in Solar Orbiter

Missions & Applications, Short Paper

Paul Norridge, David Pecover
Astrium Ltd
Stevenage, SG12AS, UK

Stefan Thürey, Wahida Gasti, James Windsor
European Space Agency, ESTEC
Noordwijk, The Netherlands.

Joachim Poeckentrup
Astrium Gmbh
Friedrichshafen, Germany

Michele de Meo
Thales Alenia Space
Milan, Italy

***Abstract*—The Solar Orbiter spacecraft data handling and instrument communication architecture comprises a SpaceWire network. This includes command and monitoring of instruments, transfer of data for on-board storage/retrieval and inter-instrument communication.**

***Index Terms*—SpaceWire Network, Solar Orbiter DHS, FDIR.**

I. Introduction

The European Space Agency's Solar Orbiter mission is designed to carry an extensive complement of scientific instruments to the near-Sun environment and generate unique insights into the workings of the Sun. While the mission presents many challenges, this paper focuses on just one: the distribution of data and command packets between instruments and the data handling functions of the spacecraft platform.

The Solar Orbiter payload consists of ten instrument suites, each providing scientific data and engineering telemetry data for downlink, and requiring command interfaces to the spacecraft. In addition, many instruments intend to share measured parameters to optimize the scientific return. The Spacecraft must support all these data flows. As the downlink bandwidth is small at large spacecraft-Earth distances, high capacity storage of science data intended for transmission must also be provided.

SpaceWire has been selected as the sole communication interface between the payload instruments and the spacecraft, building on heritage from the BepiColombo mission. This paper gives an overview of the architecture and the design choices made in response to the need for inter-instrument communication and robust failure handling.

II. Mission overview

The Solar Orbiter mission takes the next step in the Sun's observation from space. The mission profile and instrumentation enables the exploration of the largely uncharted innermost region of the solar system. The selected orbit allows observations of Sun from as close as 0.28AU and includes perihelion fly-bys that are tuned to the Sun's rotation rate in order to provide a co-rotating vantage point of the Sun's surface.

The mission focuses on four fundamental questions about the interaction of the sun with the heliosphere:

- What drives the solar wind and where does the coronal magnetic field originate?
- How do solar transients drive heliospheric variability?
- How do solar eruptions produce energetic particle radiation that fills the heliosphere?
- How does the solar dynamo work and drive connections between the Sun and the heliosphere?

In order to address these questions, the Solar Orbiter spacecraft carries then instrument suites that will gather data and provide observations of the Sun. The instruments can be partitioned into "remote sensing" – which image the solar photosphere and the corona in a number of different wavelengths – and "in-situ" – which measure key characteristics of the solar wind (e.g. particle content and magnetic & electric fields from DC to several MHz) at the spacecraft location. The in-situ instruments are expected to generate a steady, low bandwidth flow of data throughout the spacecraft operations. The remote sensing operations are restricted to three key phases in each orbit and will typically alternate between quiet periods and periods of high bandwidth data production.

The spacecraft will follow a highly elliptical orbit between 0.28AU and 0.9AU. The orbit is design to incorporate a number of Gravity Assist Manoeuvers at Venus which will systematically increase the inclination of the orbit with the solar equator over the mission life, making the poles of the sun accessible to the imaging instruments. The characteristics of the orbit mean that the available bandwidth for downlink of science data will vary greatly over time and there will be long periods when the spacecraft is out of line-of-sight from the Earth. For this reason, the spacecraft data handling must provide a central mass memory function where data can be stored until sufficient downlink resources are available. Naturally, the mass memory function must have sufficient flexibility to receive data from the instruments according to their selected data generation schedules.

As well as generating data for downlink, there is a requirement for the instruments to be able to transfer information amongst themselves. These interactions are intended to facilitate optimal science return both by allowing measurements to be focused on the most interesting events and by ensuring optimal use of the able downlink bandwidth via intelligent data reduction.

The inter-instrument interactions come in two types. The first is the transfer of key measured parameters to allow results from one instrument to be shared with others in real-time. For example, the Magnetometer (MAG) instrument will provide a measurement of the magnetic field vectors to the Solar Wind Analyser (SWA). SWA measures the velocities of incoming electrons. Although it is able to measure a full 360^0 field of view, this cannot be done at high rate. When provided with field vectors it is able to focus high rate measurements of electron velocities at the most interesting point – along the magnetic field lines.

The second type of inter-instrument communication permits one instrument to alert others to specific events, which may trigger specific or high rate measurements. Bandwidth restrictions would not permit continuous high rate measurements, but there are occasions when more frequent measurements are particularly desirable. For example, the Radio & Plasma Wave (RPW) experiment will indicate the detection of an interplanetary shock to SWA (and others). This trigger allows the receiving instruments to switch into a so-called burst mode for the duration of the shock event.

In addition to supporting messaging between payloads, the spacecraft is required to provide a precise time distribution to allow multiple instruments to carry out coordinated and correlated measurements.

III. Architecture

SpaceWire has been selected as the sole communication interface between each of the instruments and the spacecraft data handling subsystem (DHS). It also provides a key interface within the DHS itself, between are the On-Board Computer (OBC) and the Solid State Mass Memory (SSMM).

A. Network Requirements

From the above system description we can identify a number of key traffic flows that the SpaceWire network must support:

- Telemetry & science data from instruments to SSMM
- Telecommands & operational telemetry between the OBC and instruments
- Telecommands & operational telemetry between the OBC and SSMM Processor
- Inter-instrument exchanges (see above)
- Time Code distribution

For Solar Orbiter, the average science data rates are sufficiently low that a multiplexing of data and TM/TC packets is not critical for system performance. The average science data rate is driven by the available downlink bandwidth; the maximum allocation for any instrument is 20.5 kbps. Further, the commanding rate is expected to be low.

The network must ensure that the message latency – both from routing and packet collisions – is low. The primary driver on latency is the inter-instrument communication. It will only be possible to take full advantage of instrument collaboration if the time to transfer triggers and parameters is kept short.

Naturally, the network architecture must also ensure failure-tolerant routing between the various nodes on the network. This should include handling of failed links and unwanted node behavior.

B. Nodes

Before detailing the network implementation, it is useful to summarise key aspects of each of the nodes that must be supported – the OBC, SSMM, and instruments.

The OBC (supplied by RUAG, Sweden) is the primary computer on the spacecraft and hosts the central control software. As well as processing resources, it provides reconfiguration electronics, 8 Gibits (EoL) of redundant Mass Memory (used for storing platform housekeeping data) and the Transfer Frame Generator (TFG), which handles all traffic for transmission to the ground. The OBC provides two sets of SpaceWire interfaces. One set is associated with the processing function and is available for management of the other units on the network. The other is associated with the TFG and is dedicated to reception of data from the SSMM for downlink.

The SSMM (supplied by TAS, Milan) provides a central Mass Memory of 512 Gibits (EoL) for storage of telemetry packets. The storage is available to the instruments for both science measurement and housekeeping data. (It is also able to accept data from the OBC, but this capability is not utilized by the Solar Orbiter system design.) The SSMM is functionally partitioned into a Memory Array, which provides the data storage; an input function, which routes incoming packets to the appropriate memory array locations; an output function, which transfers packets from the memory array to the OBC TFG; and a Memory Controller, which provides control and monitoring.

The instruments vary in implementation, but typically include an instrument controller consisting of a LEON processor and accompanying FPGAs. Each instrument is required to provide redundant SpaceWire interfaces to the Spacecraft, which support both commanding, telemetry, and the transfer of science data. It is left open to the instrument designers whether these interfaces are each connected to separate, cold redundant controllers or whether they are both managed by a single controller.

Many instruments have elected to incorporate large memories internally for buffering of science data. This allows some flexibility in selection of the data transmitted to the ground. For example, an imaging instrument may store a large number of image files and only send to the SSMM those associated with the most interesting events. This ensures that available downlink bandwidth is focused on the most important information. The consequence for the spacecraft is that these instruments are likely to employ a very low data rate during an observation window followed by a high rate transfer once data selection is complete.

The SSMM and instruments all function as packet terminals based on the ECSS Packet Utilisation Standard.

C. Network Implementation

The Solar Orbiter SpaceWire network builds on the architecture developed for the BepiColombo Data Handling Subsystem (under project prime contractor Astrium Gmbh). Both missions share the same basic network configuration between the Onboard Computer (OBC), Solid State Mass Memory (SSMM) and instruments. The differences are limited to the number of instruments in each mission and the detailed transactions that the network is expected to support.

When considering the network design, it is important to keep in mind the accommodation of the routing function as well as the basic topology. There are many possible topologies that could support the required data flows, but the final selection must map to a hardware configuration that is feasible both technically and industrially. For example, an optimal theoretical solution might involve a standalone routing box near to a group of instrument nodes, but the need to provide a dedicated power converter and mechanical housing could quickly result in an over-complicated implementation.

For Solar Orbiter and BepiColombo, the decision has been made to accommodate the entire network infrastructure into the SSMM alongside the data storage and retrieval. In fact, this is a natural choice since all other network nodes already interface with the SSMM directly, either for storage or commanding. The supplementary function that the SSMM must provide is to allow packets from the OBC to be routed through the SSMM to the instrument and vice versa.

The SSMM contains a suite of routers that connect the external units to SSMM resources and also allow external units to send packets amongst themselves. In this way, packet transfer between external units is transparent to the SSMM and places no load on the SSMM processing resources. The SSMM provides cross-strapping between nominal and redundant interfaces by linking routers directly. This reduces the amount of cross-strapping needed external to the unit and provides a high degree of flexibility.

All routing in the system is implemented using Logical Addressing (with the exception of router configuration packets). The same Logical Address is used for both nominal and redundant interfaces with external units. This simplifies messaging for the OBC and instruments, since it is not necessary for them to know which interface is operational when constructing SpaceWire packets.

A simplified depiction of the network architecture is given in figure 1. Further details of the SSMM implementation are provided in [1].

While the OBC has overall responsibility for the system configuration, the SSMM handles the low-level configuration of the routers. For instance, the OBC can specify a particular instrument interface to be used and the SSMM will reconfigure the routers as necessary. Similarly, if a router is identified as failed, the OBC commands the SSMM to power this component down and the SSMM updates the network to ensure that packets are still routed correctly. For complex recoveries, the SSMM provides commands for low-level configuration.

D. Protocols

As noted above, all nodes on the SpaceWire network are PUS packet terminals and, hence, data are transferred on the SpaceWire links via the CCSDS Packet Transfer Protocol. The only exception is the use of RMAP packets sent within the SSMM to configure the router ASICs.

Solar Orbiter and BepiColombo implement the same scheme for distribution of the on-board time (OBT). The OBC sends a time code each second. The “time count” in the packet is in units of 1 second and reflects the current OBT value (modulo 2^6s). On arrival of a time code the receiving unit sets the sub-second component of the local time to zero and aligns its local seconds count with the value supplied in the time code packet. For a full update of the local time of a payload, a combination of time code and telecommand packet is used. In this case, the TC packet specifies the full OBT to be applied at receipt of the next time code.

IV. Inter-instrument communication

As discussed above, it is important that the Solar Orbiter instruments have the ability to communicate with each other in real-time and without the need for ground interaction.

One possible fulfillment of this requirement would be to simply allow all instruments to send data directly to each other via the SpaceWire network. So, for example, the MAG payload could generate a packet containing field vectors and send copies of this to the interested instruments. A cursory inspection of the architecture outlined will show that the Solar Orbiter routing topology has the flexibility to support this. However, this approach would not be robust against anomalous

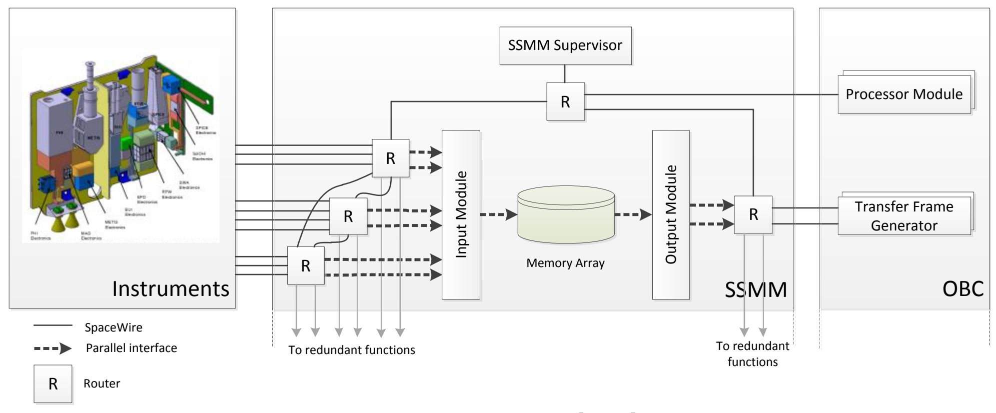


Figure 1: SpaceWire network architecture

behavior of a single network node. A better solution is to use the OBC as a hub for collating & disseminating information. That is, the OBC provides a facility for each instrument to submit data for distribution; at regular intervals it then consolidates the data provided by all instruments and sends a single combined packet to all the payloads.

Two system-level considerations motivate this centralized approach. The first is the risk associated with permitting payload mode changes that are not managed by the ground or the platform (e.g. transition to a measurement burst mode). The concern is that such mode changes could impact spacecraft performance without warning or opportunity for OBC veto. The second is related to spacecraft validation. It is important to demonstrate that the whole system can operate in a correct and robust way. Clearly, allowing free communication between all network nodes will make this validation process highly complex, both in analysis of traffic scenarios and in testing. In particular, final proving of robustness would not be possible until all components are integrated.

Although the primary motivation comes from the system level, a centralized architecture presents a number of benefits for the instrument teams as well. To understand these, it is useful to note that the proposed approach is strongly analogous to the Mediator pattern used in object-oriented software engineering and brings with it many of the same advantages. Most importantly, use of a Mediator object allows decoupling of the system components and reduction of interface complexity. For Solar Orbiter the consequences of decoupling are that the instruments can be designed with minimal reliance on other payload developments. In testing, they can be validated with a single, simple interface and do not need to consider the interaction of messages arriving from multiple sources. Operationally, the source instrument for a trigger does not need to know if the recipients are present or operational before sending packets into the network; only the OBC needs to be available and, as it already has knowledge of the system, can distribute information appropriately. Similarly, receiving instruments do not need to be aware of the ultimate source of data or triggers, or even the interface definition of other system components. If an instrument is not operational for some reason or changes its interface (rate, format) late in development, only the OBC needs to know and the other units will be largely unaffected.

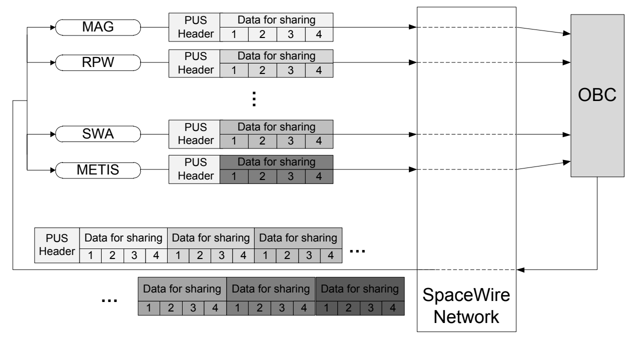


Figure 2: Centralised architecture for inter-instrument communications

The messaging approach used to support this centralized architecture is based around PUS service packet exchanges (see figure 2). Each instrument sends a fixed TM packet to the OBC at regular intervals. The data in these TM packets are extracted by the OBC software and placed in a data pool. The OBC then collates the data to be shared from the data pool into a single periodic TC packet; copies of this TC packet are then distributed to all operational payloads. The rate at which the TM and TC packets are transferred can be defined by the instrument developers up to a maximum of 8Hz.

As will be seen below, this periodic messaging between OBC and instruments has further advantages for network failure detection. It also provides a natural method for instruments to be warned of potential spacecraft anomalies: When the spacecraft experiences a major anomaly, the OBC will halt communications with the instruments and focus on essential tasks to save the spacecraft. It is also likely that all instruments will need to be powered down in this mode. In some cases it may be necessary for an instrument to take action before power is removed (for example, closure of a protective door). With the scheme envisaged here, the instrument may identify an imminent power down by monitoring the incoming TC packets used for inter-instrument communications and/or Time Code arrivals. When a pre-defined number have been missed in a row, it may assume that a problem has occurred and a power down should be expected. It is not important to the instrument whether this loss of traffic is due to a controlled recovery by the spacecraft software or an OBC hardware issue.

V. FDIR

The design of a SpaceWire network and messaging must ensure the system can respond to failures and to unexpected behaviour of nodes. The experience of Solar Orbiter and BepiColombo has clarified a number of problems that must be addressed to ensure a robust system. Issues identified in these missions include network hardware failures, excessive traffic to the DHS units and failures in communication with instruments.

A. Network failures

The primary need to ensure a robust network is the ability to detect any link or routing failures, isolate the problem and reconfigure appropriately. For Solar Orbiter, the detection mechanism has two layers. At system level, detection is built on the inter-instrument communication protocol. Failure will be identified by noting halts in the regular packet transfer. This provides a natural way to identify problems in the paths between the instruments and OBC. In effect, the regular periodic TM packets act as a "heartbeat" for each respective source, which the OBC can monitor. Any extended loss of the "heartbeat" can be used to identify a potential failure and, if appropriate, trigger recovery actions.

In fact, the "heartbeats" can be used to diagnose the location of a failure with some precision. For example, if the missing packets are limited to a single source then it is likely that there are problems either in the instrument itself or on the link between the instrument and the SSMM. Alternatively, if the OBC fails to receive packets from multiple nodes then it is likely that the problem is in a router common to those nodes.

The architecture of the SSMM network is sufficiently simple that this will often be enough to localise the problem.

In addition to the system-level detection, the SSMM itself provides a further layer of protection. Since the SSMM memory controller has visibility of the key network components it can monitor for low-level failures and report these to the OBC. For some cases, the recovery can be made autonomously (e.g., time-outs due to stalled transfers, see [1]).

The instruments will also indicate if they do not receive expected packets. In particular, if time codes are missing then this can be signaled either with a dedicated telemetry packet or by setting a dedicated bit in the instrument time field.

B. Anomalous instrument behaviour

It is well know, that a so-called “babbling idiot” has the potential to compromise operations on a bus or network. With a large number of instruments present on the BepiColombo and Solar Orbiter networks, we must ensure that one unit failure does not threaten the other instruments or even the entire mission.

The feared scenario is that of an instrument sending an excessive number of packets into the network either due to a failure in the unit or due to a cascade of events resulting in a high rate of corresponding TM packets. In some cases this behaviour will be relatively benign, with a highly constrained impact on system performance and simple recovery. For instance, if an instrument sends packets at a high rate to the SSMM Memory Array it may lead to some Packet Stores filling unexpectedly and loss of science data, but will not threaten overall operations. Similarly, packets sent to an incorrect logical address may cause a small amount of congestion in the network, but will eventually be discarded by a switch or node. The critical scenario is when an instrument sends an excessively high rate of packets to the OBC. In this case, there is a strong possibility that the spacecraft control software could become overloaded while trying to handle the high traffic volume. For example, if the SpaceWire packet handling is interrupt-driven then a high number of small packets would lead to a high interrupt rate; processing these interrupts could quickly dominate the CPU budget and ultimately cause the OBC to crash. Some protection against such events can be built into the spacecraft software, but we must also incorporate safeguards into network where possible.

Solar Orbiter and BepiColombo both include guards against babbling instruments, but due to other design considerations have elected to handle the failure in different ways.

BepiColombo gives the SSMM the additional role of a “gatekeeper” for the OBC traffic. That is, instrument telemetry packets are never sent directly to the OBC, but are written to a dedicated Packet Store within the SSMM Memory Array. This packet store acts as a “cache” for the telemetry packets. A separate process in the SSMM then polls this “cache” packet store at regular intervals. If packets are present, they are forwarded to the OBC, but with a strict limitation on packet rate. Configuration of the SpaceWire routers will ensure that instrument packets can never be routed directly to the OBC (even if there are physical links available for this path). In this way, the SSMM isolates the OBC from the instruments. Since the SSMM is designed to handle high packet rates into packet stores it is unlikely to be affected by a babbling source and, in any case, in contrast to the OBC a temporary degradation of SSMM performance is not catastrophic.

In contrast, Solar Orbiter allows the possibility of high packet rates arriving at the OBC, but will react to any problems by deactivating the SpaceWire link between the OBC and SSMM. That is, rather than employing a “gatekeeper” to control the flow, the gate is simply closed under excess traffic conditions. The link deactivation can be triggered in one of two ways: either the software detects the problem and commands the hardware appropriately or, if the failure forces an OBC reconfiguration, the hardware links are not restarted automatically and will only be reactivated when it is considered safe.

As noted above, these different approaches are driven by other system needs and are not simply transferable between the two. For Solar Orbiter, the caching approach is not viable due to the tight latency requirements. The value of the inter-instrument communication depends on maintaining low message latencies throughout the network. If data is buffered in a packet store during transfer then the latency requirements are quickly exceeded. On the other hand, link deactivation is not possible for BepiColombo because the SSMM is used to store all of the system telemetry. For this reason, the link between the OBC and the SSMM Memory Array must be available at all times and especially during disruption of normal operation. If the link is lost for a significant period then important telemetry data will not be available to ground operations, impeding monitoring and, possibly, fault diagnosis.

C. Loss of packets

Since SpaceWire does not currently provide a guaranteed delivery service, the system operations design must ensure that no key events are missed due to packet loss. This is particularly important when routing though multiple switches. In a point-to-point transfer, both of the nodes will be aware of link interrupts and, hence, will know when packets have been lost. For a network such as the one considered here, an intermediate link might fail and cause a packet spill. In this case, one can envisage scenarios where neither end of the transaction has sufficient information to initiate a retransmission: the sending node is unaware of the loss and the need to retransmit; the receiving node may receive a partial packet, but partial or corrupted packets cannot be evaluated to determine the source.

The approach taken in Solar Orbiter is to ensure that all critical information that requires a reaction by the OBC is conveyed via status parameters in cyclic TM packets (e.g. in periodic housekeeping packets). Loss of one of these packets will only result in a short delay in the recovery action, since the next edition of the packet will provide the missing information. Non-critical information can be conveyed by a non-cyclic TM packet, with the accepted risk that the packet may be lost.

REFERENCES

[1] M. de Meo, G. Saldi, G. Rosani, W. Gasti, J. Noyes, J. Windsor, J. Poeckentrup, R. Eilenberger, “BepiColombo Solid State Mass Memory employing SpaceWire,” in these proceedings.

Fast readout CCD camera with high performance SpaceWire to PCI express acquisition board

Missions & Applications, Short Paper

C. Cara, M. Donati, E. Doumayrou,
F. Pinsard, M. Lortholary
AIM Paris Saclay, UMR CEA/CNRS/UP7
CEA / Irfu / SAp
Gif sur Yvette, France
christophe.cara@cea.fr

Abstract— **In order to demonstrate the feasibility of the instrument concept the development of a prototype was initiated following the submission of the NEAT instrument as a M-class mission to ESA. The development is currently supported jointly by CNES, CNRS and CEA in France. This prototype relies on a fast readout (1000 frames per second) CCD-based camera. The CCD is located in the focus of an optical bench to simulate the instrument transfer function and thus creates interferogram patterns on the detector. The space electronics laboratory of the Astrophysics Department of CEA Saclay is in charge of the design of the fast CCD camera while the Institute of Planetology and Astrophysics of Grenoble is in charge of the optical bench design. The functions of the camera are split in two units: the front-end electronics encompasses the analog functions and an acquisition system for control visualization and archive functions. The acquisition system features the new 4 SpaceWire to PCIexpress interface board using the SpaceWire CEA IP to handle the 160 Mbps camera data rate. In the present paper we will describe the acquisition hardware focusing on the SpaceWire to PCIe interface board and present the end-to-end performance of the acquisition system as well.**

Index Terms—**EGSE, Linux, NEAT, PCIexpress, , SpaceWire.**

I. Introduction

Since long time ago the Astrophysics Department of CEA in Saclay is developing innovative imaging detector systems for the astrophysics. These detector systems are primary designed to operate aboard satellites but find application in ground based astrophysics instruments as well. Detector systems are composed of the sensor whose function is to convert photon to electron and readout electronics. In turn readout electronics is usually composed of a proximity readout electronics (the so called cold electronics since most of the applications require cooled-down sensor) coupled as closed as possible to the sensor and a front end electronics (the so called warm electronics) The space electronics laboratory main activity is to develop such warm electronics for instruments aboard satellites, for ground based instruments and for detector test bench as well. Thus we have recently developed electronics for the readout of CCD for demonstrators and test benches for the forthcoming ESA Cosmic Vision missions such as EUCLID, PLATO, ECHO and NEAT [1]. Similarly to other applications, astrophysics instruments require more and more spatial resolution leading to million-pixel sensors. However since astrophysics instruments deal in observing mainly faint sources, image integration time are long (up to hundreds of second) and consequently output data rate of readout electronics remains reasonable. In the opposite the NEAT (Nearby Earth Astrometric Telescope) aiming in identify and characterize planetary systems close to our solar system requires smaller sensors but high readout rates for specific observing modes [2]. Indeed instrument is constantly switching between a low image rate and high rate image mode respectively for star observing and metrology observing. Metrology consists in imaging Young's interference fringes and is implemented to measure regularly the telescope geometry (split in two parts since the mission relies in two satellites in formation flying configuration) in order to achieve a localization of the source with a resolution of 10^{-6} with respect to the pixel size. To reach such extreme performance a large number of photons has to be collected within a shortest period of time to limit statistics errors leading to image rate beyond 1000 per second.

The development of the demonstrator electronics has been initiated in early 2012 on the basis of previously designed cameras. Thus we choose to implement a SpaceWire interface between the front-end electronics and the acquisition system taking advantage of existing hardware (*SpaceWireCEA* IP, SpaceWire to PCI express interface board) and software pieces. Beyond the development of the demonstrator we have considered the opportunity to experiment high data rate SpaceWire links and

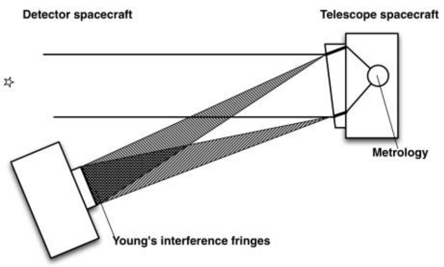


Figure 1 - Instrument layout

acquisition system and thus be ready for further high data rate demanding developments.

II. Overall Architecture

The demonstrator consists in a mirror, a CCD camera, five punctual white sources (they represent stars, so called “pseudo stellar sources”). The pseudo stellar sources are fed with white light, the wavelength ranging from about 400 to 800 nm. Additionally, single-mode metrology optical fibers are located on the mirror plane. A schematic of the system’s components is shown in the next figure (figure 2). The most innovative aspect of this experiment is the metrology system that will allow the micro-pixel calibration of the CCD. This system consists in at least two metrology bases (i.e. two pairs of single mode fibers transmit a laser generated coherent illumination), respectively aligned along the horizontal and vertical axis. The fiber extremities are located next to the mirror and project Young’s fringes on the detector. Additionally a phase modulator is used to dynamically sweep the fringes over the focal plane. By measuring the intensities variations of the signal for each pixel, one can characterize the inter- and intra-pixel response of the CCD and bring the centroid error down to the level of a few micro-pixels. The optical test bed is located inside a vacuum vessel in order to limit effect of thermal fluctuation of the atmosphere along the optical path.

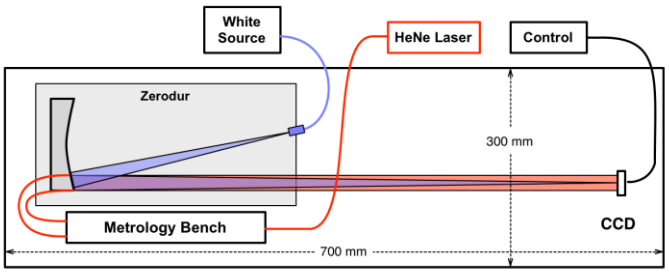


Figure 2 - Test bed layout

III. Front End electronics Overview

In the framework of the demonstrator development we have fully designed the front-end electronics. These electronics housed in an enclosure for mechanical, thermal and EMC aspects is mounted directly on the optical test bed enclosure to limit as much as the distance between the detector and the readout electronics. The electronics unit has analog interfaces toward the CCD for clocks and biases driving and from the CCD for video signals processing. The unit has also digital interfaces; one being devoted to its control and configuration while the second; which will be discussed later in this paper is devoted to the transmission of the processed video signals. Finally power supplying is achieved by mean of an external laboratory power supply (Fig. 3). Internally the various functions of the front-end electronics are implemented in three printed circuit boards. On top the first board hosts the four video signal processing chains. Each chain features an AC coupled preamplifier followed by the double sampling stage. Finally a single to differential amplifier feeds the 16-bit analog

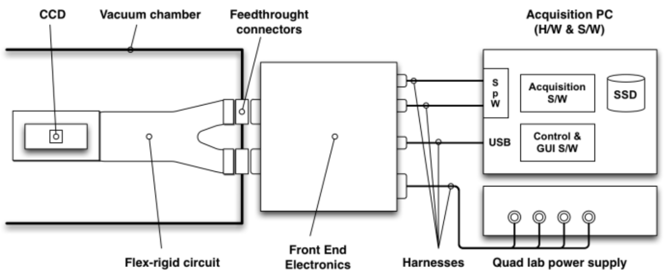


Figure 3 - Electrical Architecture

to digital converter with the processed video signal. This analog chain is optimized to sample the detector’s video signal at a 3-MHz pixel rate. Bellow, a second board is hosting the sensor’s clocks and biases generators: it receives from the digital board the clock sequence pattern and shifts the logical low and high levels to detector’s compatible low and high analog levels. Thanks to a bench of digital to analog converters all clocks and signals levels are tuneable to optimize detector’s performance. Monitoring of all the generated voltages is achieved by mean low speed analog to digital converters. The last board is a digital board whose function is manifold. Its main function is to control and configure the analog functions of the units including the digital to analog converters setting, the acquisition of the four digitized video signals and the generation of the clock sequence. It implements the digital interfaces of the camera as well. A demonstrator requiring high level of flexibility a fully programmable clock sequencer is implemented to allow users to experiment with CCD readout modes. Basically this sequencer has a time resolution of 100 ns that is compliant with the slow readout of large sensors but too coarse for fast readout as required by NEAT. Therefore this clock sequencer has been upgraded to achieve better time resolution: a 50-ns time resolution design have been developed. Better clock resolutions could probably be reached with highest FPGA’s clock frequency but keeping in mind further developments for space born instruments our design is optimal.

IV. Acquisition Hardware description

The acquisition hardware which will determinate the performance of the camera in term of image relies on a dual SpaceWire link interface between the front-end electronics box and the acquisition system. Two *SpaceWireCEA* IP cores are embedded into one of the XILINX SPARTAN FPGA of the digital board of the front-end electronics. Along with the IPs a state machine are in charge of the readout of the four 16-bit 3 Msps parallel interface analog to digital converters by using the combined busy flags of the devices. This state machine then transfers the digitized data of two of the analog to digital converters to one of a first SpaceWire transmitter while it transfers the two others to the second transmitter. On the same digital board a second FPGA (so-called SECOM) is hosting the clock sequencer for controlling the CCD readout sequence and the sequencing of the front-end electronics analog operation (including CDS switches and analog to digital converters control). It provides as well an image sync signal to the SpaceWire FPGA (so-called INAC). This signal being asserted synchronously with the beginning of the detector readout sequence trigs the generation of both previous image frame trailer and the next image frame header. Since the generation of the header data may occurs while the first pixels are read out a small memory buffer is implemented in the detector data path. During the header generation the incoming pixel data are stored temporarily into this small memory buffer. Thus this memory may have a depth equal to the size of the frame header only (few tenth of bytes). Once the whole header data has been transferred to the link's transmitter the buffer memory is flushed toward the SpaceWire at the max data rate of the interface. Then the link transmission rate has to be selected such as the overload of the small memory buffer never occurs. In our application a link signalling rate of 120 MHz is required to afford an incoming data rate of 80 Mbps. Again this solution is optimal for space-born instrument since it limits the size of the memory buffers to be implemented to amount compatible with FPGA internal memory capacity. A functional block

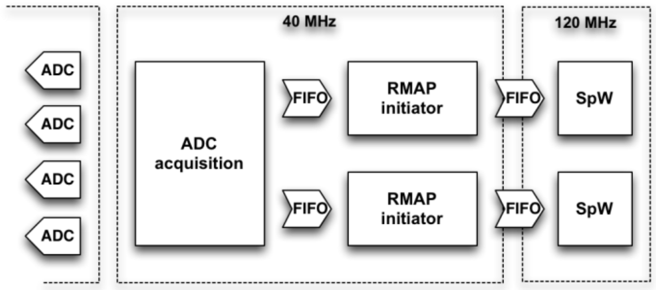


Figure 4 - SpaceWire FPGA block diagram

diagram of the SpaceWire FPGA is depicted in figure 4. As shown two clock domains are defined: a 40 MHz signal is clocking the ADC acquisition blocks the memory buffers working as FIFOs and the RMAP initiator block. A second signal running at 120 MHz clocks the two SpaceWire IP cores. In between two FIFOs are implemented for the transfer of the data from one clock domain to the other. As described previously the frames are RMAP formatted: the header and the trailer are compliant with the RMAP standard and thus contain addresses for both 'target' and 'initiator' identifiers for 'process' 'instruction' and 'key' and 'data length' and CRCs as well [3]. The 'data' field of the packet contains the 'frame number' followed by the 'image' data and a 'data CRC' (Figure 5).

Octet			
#1	#2	#3	#4
Target logical address **(0x20)**	PID **(0x01)**	Instruction **(0x64)**	Key **(0x00)**
Initiator logical address (SpW#1 : **0x40** SpW#2 : **0x50**)	Transaction ID (MS)	Transaction ID (LS)	Extended address **(0x00)**
Address (MS) **(0x00)**	Address **(0x00)**	Address **(0x00)**	Address (LS) **(0x00)**
Data length (MS) **(0x00)**	Data length **(0x00)**	Data length (LS) **(0x00)**	Header CRC
Frame number (MS)	Frame number	Frame number	Frame number (LS)
Flag (MS) **(0xAA)**	Flag **(0x55)**	Flag **(0xAA)**	Flag (LS) **(0x55)**
Data1 ADC1 (MS)	Data1 ADC1 (LS)	Data1 ADC2 (MS)	Data1 ADC2 (LS)
Data2 ADC1 (MS)	Data2 ADC1 (LS)	Data2 ADC2 (MS)	Data2 ADC2 (LS)
...	...	...	...
Padding **(0xCE)**	Padding **(0xAC)**	Padding **(0xEA)**	Data CRC
EOP			

Figure 5 - RMAP formatted frame

The next major part of the acquisition hardware is the SpaceWire PCIe acquisition board (so-called PCIe4SpW). This board has been designed in our laboratory to fulfil test equipment needs. It is derived from our previous PCI board and is focusing in high data rate acquisition application. The board is equipped of a PEX8311 PCIe bridge from PLX that merge a PCI to local bus bridge and a PCI to PCIe translator. This device implements a one lane PCIe compliant with the 1.0a specification of the bus. Along with the bridge a VIRTEX 4 FPGA from XILINX is implementing the four SpaceWire links as well as the interface with the PCIe bridge. Both input and output data buffering is achieved by means of two-4Mbytes SDRAM chips, one for each direction. The board has

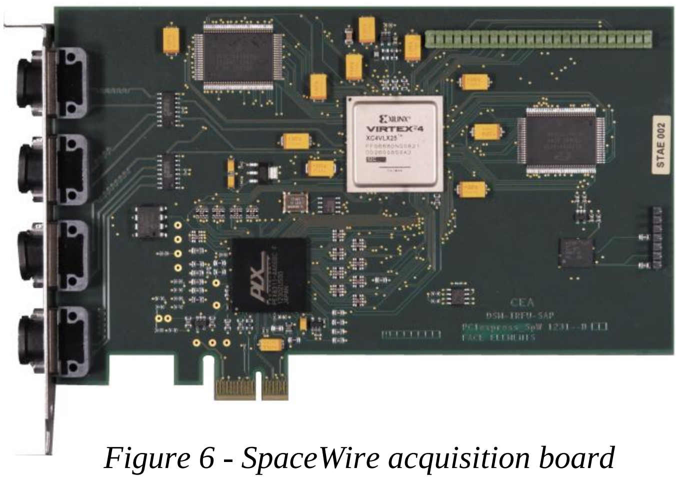


Figure 6 - SpaceWire acquisition board

a extension connector (see picture in figure 6) whose purpose is be able to add SpaceWire link interfaces using space grade LVDS transceivers as required for the design of an electrical ground support equipment (EGSE).

The complete FPGA architecture is depicted in figure 7. As shown each one of the four channel encompasses a SpaceWire block (the SpaceWireCEA IP core) interfacing with three blocks: 'data formatting' – for RMAP format implementation, 'cmd sequencer' – for command scheduling and 'time management' for implementation of CCSDS Time Management Standard [4]. In addition a 'clock manager block' is in charge of the generation of various clocks either

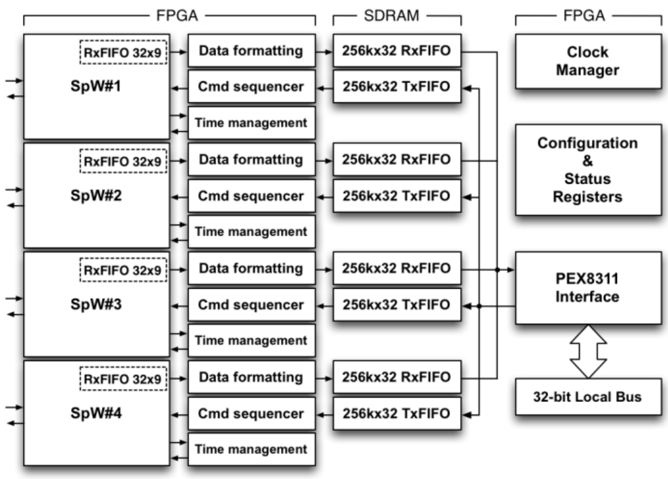


Figure 7 – PCIe4SpW board architecture

used by the 'PEX' interface or by the SpaceWire blocks. In particularly it enables the selection of link signalling rate individually ranging between 2 and 200 MHz. Another block ('configuration & status registers') is a bench of read / write & read only registers respectively used to configure the board, i.e. to set the link signalling rate and for the communication with the acquisition software, i.e. the number of data to be read out. Finally the last block ' PEX interface' implements the interface between the blocks of memory and the local bus interface with the PEX8311 device.

V. Acquisition Software Description

The acquisition software has two high-level functions: the acquisition task and the control task (Figure 8).

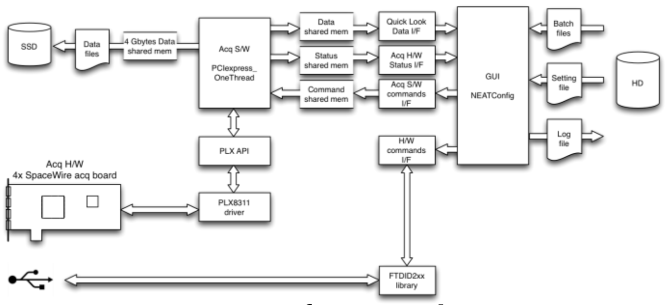


Figure 8 - Software Architecture

The acquisition task is in charge of the control of the acquisition board: when launched it initializes the board by resetting the FPGA and loading the configuration settings. Once initialization is achieved the task starts polling the number of data available into each link's input buffer by reading SpaceWire FPGA registers. Then the DMA engine of the PCIe bridge is set to perform copy of data from the board's memory into a DMA buffer. This operation is performed successively for each link. The next step of the acquisition process consists in storing the contents of the DMA buffers either directly into the hard disk of the acquisition PC or into a very large bank of the PC memory (sized to 4 GBytes according to the NEAT demonstrator requirements). In our Linux operating system this memory is declared as a shared memory. Once full or upon reception of a low-level command the data stored in the shared memory is archived into a 6-Gbit/s SATA SSD disk. The acquisition task is in charge as well to feed a second shared memory: this memory is implemented to allow the transfer to the GUI of small amount of data used to perform a quick look analysis of the images. Finally a third shared memory is implemented to achieve reception in the acquisition of low-level commands. These commands enables to switch acquisition software between acquisition / no acquisition and file / no file modes. The acquisition software is written in C++ that makes easier the implementation of the four SpaceWire links: an 'acquisition' class is defined and is instantiated for every connected link. The PLX provided API is used to implement access to the low-level bridge driver.
The control task is a large virtual instrument (VI) taking advantage of the flexibility of LabVIEW environment. The VI allows the user to select and download into the camera the operating parameter such as the detector readout sequence script or the detector bias set. Interfaces with the shared memories are implemented thanks to a wrapper, which is linking a LabVIEW VI, using the 'Call Library Function Node' feature, to a Linux shared library (.so equivalent to dll). In the current design of the camera the configuration is achieved by mean of an USB interface. This interface is driven using the API provided by FTDI to dialog with the USB to serial port device implemented in the front-end electronics.

VI. System 'Performance

Performances of the acquisition system have been assessed for two configuration of the acquisition software. In a first configuration the software was directly streaming the data into the acquisition PC's disk while in the second the data is stored in a 4-GByte shared memory. The following sums-up the performances of the system expressed in terms of maximum data rate.

Configuration	**Max data rate**		**Compliancy: data rate ≥ 113 Mbits/s**
	1x SpW (MBytes/s)	***2xSpW (Mbits/s)***	
Direct stream to disk	5.03	80.5	No
Shared memory buffering	9.86	157.8	Yes

As shown in the table the only configuration suitable for the NEAT camera makes use of the shared memory.

VII. Conclusion

We have designed an acquisition system based on the SpaceWire standard, which fulfill the requirement in terms of acquisition data rate of the NEAT demonstrator fast readout camera. In the near future it is planed to extend the handle to the four links of our acquisition board as required for forthcoming infrared detector test bench development. In order to take advantage of the command sequencer capability of the board we also plan to use the SpaceWire uplink to control the camera rather than using a dedicated USB interface. Optimization of the performances is still possible the acquisition being relying on a generic API to have access to the PCIe bridge device.

References

[1] E. Doumayrou; M Lortholary, "A generic readout system for astrophysical detectors," M. Proc. SPIE 8452, Millimeter, Submillimeter, and Far-Infrared Detectors and Instrumentation for Astronomy VI, 84521W (September 24, 2012); doi:10.1117/12.925912

[2] A. Crouzier, F. Malbet, O. Preis, F. Henault, P. Kern, P. Feautrier P, C. Cara, P.O. Lagage, A. Léger, M. Shao, R. Goullioud, "An experimental testbed for NEAT to demonstrate micro-pixel accuracy," SPIE Proceedings, Instrument and Methods for Astrophysics

[3] European Cooperation for space Standardization, Standard ECSS-E-ST-50-52C, Remote memory access protocol, 5 February 2010

[4] Time Code Formats, CCSDS 301.0-B-3, Blue book, January 2002

Application of SpaceWire to Non-Volatile Data Recorder

Missions & Applications, Short Paper

Toru Sasaki, Itao Shoji, Hisayoshi Kurosawa, Tetsuro Kato
Mitsubishi Electric Corporation
8-1-1, Tsukaguchi-Honmachi, Amagasaki, Hyogo, Japan
Sasaki.Toru@eb.MitsubishiElectric.co.jp

Satoshi Ichikawa, Takashi Okamoto, Taeko Seki, Mami Abe
Japan Aerospace Exploration Agency
2-1-1, Sengen, Tsukuba, Ibaraki, Japan

***Abstract*— Some space missions for earth observation and space science require high data rate and large storage capacity data recorders for spacecraft. The Japan Aerospace Exploration Agency (JAXA) and Mitsubishi Electric Corporation (MELCO) have developed a high-speed and large-volume non-volatile data recorder (NVDR) for these advanced requirements. This NVDR has 1 Tbytes storage capacity at the end of its life excluding ECC redundant regions. The input and output data rate for recording and replaying are over 1 Gbps. The NVDR is composed of input interface boards, output interface boards, memory boards, a DCDC board and a control board. The memory boards use NAND flash memories for non-volatile data storage. The control board controls the other boards by SpaceWire via a backplane. The SpaceWire network increases the storage capacity and the data transmission rate. SpaceWire requires fewer I/O signals of each FPGA than other types of interface. Therefore, the FPGAs in the NVDR are able to have more I/O signals for controlling NAND flash memories. Consequently, the NVDR achieves high storage capacity and high data rate by implementing many NAND flash memories. The SpaceWire used in the NVDR has a unique protocol based on RMAP. The NVDR uses this protocol only for internal control. For user interfaces, the NVDR offers standard SpaceWire protocols such as CPTP and RMAP. In this paper, we show how the architecture of the NVDR applies SpaceWire and describe the effect.**

***Index Terms*— mass memory, data recorder, NAND flash, RMAP, non-volatile.** *(key words)*

I. Introduction

Space missions for earth observation and space science have required high data rate and large storage capacity data recorders for spacecraft [1]. JAXA and MELCO have developed a high speed and large volume non-volatile data recorder (NVDR) for these advanced requirements. The NVDR uses NAND flash memories. So far, data recorders have commonly used SDRAMs which are volatile memories with low storage density. On the other hand, NAND flash memories have much higher storage density, so the NVDR has larger storage capacity. Moreover, the NVDR is able to retain stored data during power down mode because of the non-volatility of NAND flash memories. No vendor provides NAND flash memories for space applications (but screened COTS devices embedded with special circuits with radiation tolerance are available [2] [3]). We verified the radiation tolerance of some commercial products with heavy ion tests, proton radiation tests and total dose tests. As a result, we recognized that commercial products are usable for space applications by selecting appropriate products and using adequate protection methods.

Apart from redundancy, the NVDR consists of two input interface boards, one output interface board, six memory boards, one DCDC and one control board. One backplane board includes these boards. The control board controls the other boards by SpaceWire. SpaceWire improves the performance of the NVDR.

II. Status of Developement

The development of the NVDR started in 2009 with performance targets as shown Table I. In 2009, we made a conceptual design and conducted radiation tests for NAND flash device candidates. The next year, we designed the detailed architecture based on the results. In 2011, we produced a BBM and performed the evaluation tests.

TABLE I. Target Performance

Item	*Target Value*
Capacity at EOL (End of Life)	> 1 [Tbyte]
Input/Output Data Rate	> 1 [Gbit/s]
Mass	< 20 [kg]
Power	< 100 [W]
BER(Bit Error Rate)	$< 1\times10^{-16}$ [/bit/day]

III. NVDR Architecture

A block diagram of the NVDR is shown in Fig. 1. The architecture of the NVDR provides the following features:

- The mission data bus and SpaceWire control bus are independent from each other.

- The input interface boards and output interface board can have various sorts of interfaces.
- Additional memory boards can be mounted in the NVDR for larger capacity and more reliable redundancy

While the mission data bus transfers record and replay data, the SpaceWire control bus transfers data for controlling the NVDR behavior. While the former has a data rate of more than 1 Gbps, the latter has a data rate of 10 Mbps. To avoid decrease of the mission data bandwidth, these buses are separated. The mission data bus is composed of channel links [4], which use dedicated ICs for high speed serialization and deserialization. The control bus uses SpaceWire for which a unique protocol is used. This protocol is based on RMAP [5]. Because we suppose that the NVDR is applied to both earth observation and space science satellites, various input and output interfaces have to be prepared. To handle this requirement, we properly assigned storage functions and interface functions to the memory boards and the interface boards. As a result, the NVDR is only able to meet a variety of interface requirements by replacing the input or output interface boards. The input interface boards are able to receive both low rate data (e.g., HK telemetry) and very high rate data (e.g., SAR sensor data). The output interface boards are able to transmit both low rate data to S-band modulators and high rate data to X-band modulators. Table II shows capable interfaces in the NVDR. The NVDR has communication interfaces to DH subsystems for telemetry and command. These interfaces are included in the control board. SpaceWire, RS422 and MIL-STD-1553B are available.

We can easily increase or decrease the number of memory boards because the storage function and interface function are strictly separated. This results in enhancement of the scalability of the NVDR capacity and redundancy.

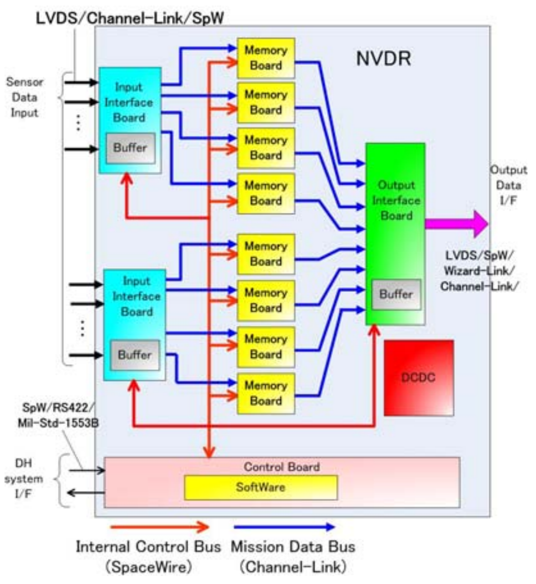


Fig. 1. Block diagram of NVDR

IV. Control Bus

As previously noted, the control board controls the input interface boards, output interface board and memory boards in the NVDR by SpaceWire. These controls mainly include configuration of the interface boards and management of the storage area in the memory boards. All SpaceWire buses are connected to a SoC (system on chip) on the control board via a SpaceWire router. To implement a bus on the backplane, generally we have a choice between shared buses such as PCI and serial buses such as SpaceWire. SpaceWire as a backplane bus gives the NVDR an advantage in terms of I/O signals. A shared bus requires many I/O signals of connected devices. However one SpaceWire node needs only four I/O signals with LVTTL (low voltage transistor-transistor logic), PCI occupies about 50 I/O signals. This advantage enables more NAND flash memories to be connected in parallel to an FPGA or ASIC. Although NAND flash memories have modest access performance compared to SDRAMs, this wide parallel connection increases the bandwidth between the device and NAND Flash memories. This technique provides the NVDR with comparable record and replay performance to SDRAM type data recorders. A shared bus needs many more I/O signals when taking into account redundancy. This results in low record and replay performance. On the other hand, one SpaceWire node needs only eight I/O signals with LVTTL even when including nominal and redundant connections.

TABLE II. NVDR INTERFACES

Item	Interface
Input I/F	LVDS/SpaceWire/Channel Link/RS422
Output I/F	LVDS/SpaceWire Wizard Link [6]/Channel Link
DH I/F	SpaceWire/RS422/MIL-STD-1553B
Internal Mission Data Bus	Channel Link
Internal Control Bus	SpaceWire

V. Record and Replay method

When the NVDR starts recording, the software on the control board decides the location of the storage area and calculates the starting address for replaying. To do this operation, the software sends several sets of commands and addresses to memory boards by SpaceWire (Fig. 2). The memory boards receive data from the input interface boards via channel link. Then, they record the data in accordance with area specified by the command and address. When replaying, the memory boards read data from the indicated memory area. Next, they transmit the data to the output interface board via channel link. After these operations, the memory boards wait for new sets of commands and addresses to continue recording and replaying. To provide new sets of commands and addresses rapidly, the software must recognize the end of the operation, but NAND flash memories have quite different access sequences compared to SDRAMs and SRAMs. The NAND flash memory sequence makes it difficult to recognize the end of sequence. NAND flash memory needs long busy time after write data cycles (Fig. 3). This busy time is indefinite and can vary with each access. To wait for the fluctuating busy time, there are two general methods.

1) *Polling*
2) *Interruption*

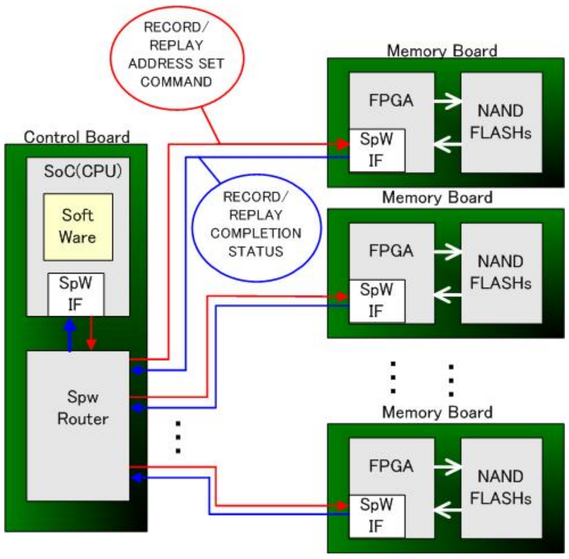


Fig. 2. Control method by SpaceWire

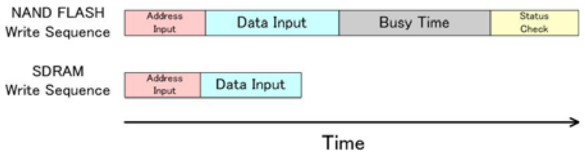


Fig. 3. NAND flash write access sequence

Both methods have disadvantages. The polling method may lower the software processing performance and the interruption method requires an interrupt controller and many signal connections between the memory boards and the control board. SpaceWire can resolve these problems. The full duplex communication of SpaceWire enables the memory boards to send status messages to the control board. These backward messages notify the control board of the sequence end. As a result, the polling method and interruption method are not necessary. The protocol for sending the messages is explained in the next section.

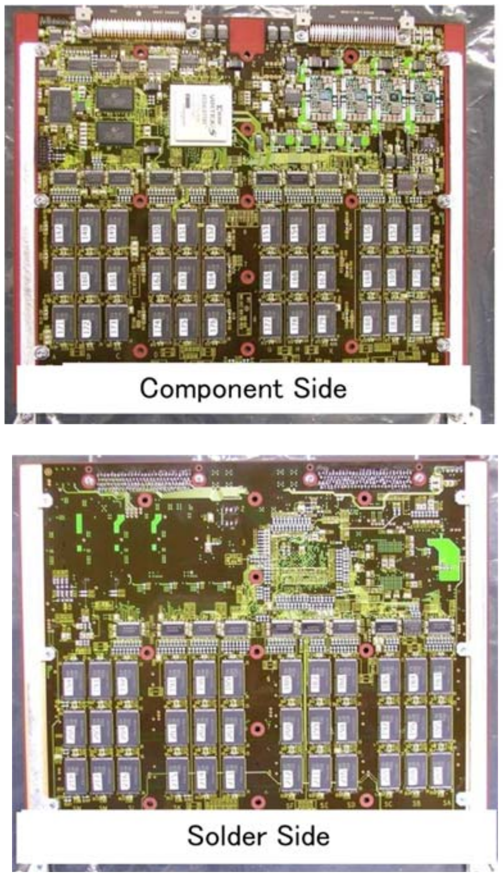


Fig. 4. Photo of memory board

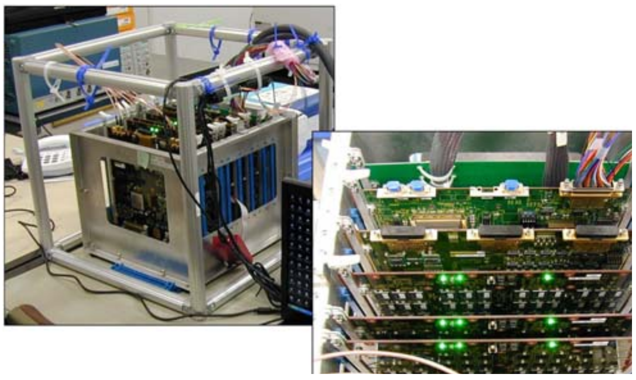


Fig. 5. Photos of evaluation test

VI. Control bus protocol

In the NVDR, the control board controls other boards by SpaceWire with a unique protocol based on RMAP. RMAP is suitable for the control bus. In this case, the initiator of RMAP will be assigned to the control board and the targets to the memory boards. However, each memory board should also be an initiator to notify the control board of the sequence end. Otherwise, the polling method or interruption method will be needed. Another solution using RMAP is to assign the roles of initiator and target to the control board and the memory boards respectively. But to realize a network with multiple RMAP initiators, complex hardware logic is required in the FPGAs in the memory boards. Therefore, we did not employ the multiple RMAP initiator method. Instead, we allow the memory boards to send data with a fixed 64-bit payload to the control board at any time from the target side. Any other rules follow the rules for RMAP. In the payload, the memory boards can include the status data of the record or replay completion. We call the data an "interrupting packet". The initiator in the control board can easily descriminate between RMAP reply packets and interrupting packets by decoding their protocol identifiers. The targets have an arbitration circuit for sending RMAP replies and interrupting packets.

The NVDR uses the unique RMAP with the interrupting packets only in the internal control bus. The NVDR provides standard protocols such as RMAP or CPTP [7] for the external input/output interfaces or the DH interface.

VII. Evaluation tests

We manufactured a BBM for the NVDR, which was composed of one input interface board, one output interface board, one control board and three memory boards (Fig. 4), and evaluated it. In evaluation tests, we tests for the capability of the data rate during recording and replaying, functions for correcting data errors with ECC, and long time stability (Fig. 5).

TABLE III. Result of the evaluation

Item	Specifications
EOL storage	1.17 [Tbyte]
Input Rate	1760 [Mbps]
Output Rate	800 [Mbps]*
Mass	20 [kg]
BER	$< 1\times10^{-16}$ [bit/day]
Power	MAX. 98 [W]
Size	$310 \times 375 \times 265$ [mm^3]

*The output rate was determined by specifications of receiving instruments

The results of the evaluation show the NVDR satisfies the specifications shown in Table III. The FPGA logic in the memory boards can generate intentional data errors for the ECC functional tests. The intentional data errors mimic various errors that happen in the NAND flash memories. We concluded that data errors that may occur in GEO or LEO will be properly recovered by the ECC. Moreover, we confirmed the NVDR can continue the record and replay operation without any failure for up to 80 hours.

VIII. Conclusion

In this paper, we showed how SpaceWire is applied in the NVDR. SpaceWire contributes effectively to the record and replay performance and the scalability. Our unique protocol based on RMAP optimizes the network architecture of the control bus. Our evaluation test showed that the NVDR architecture is a good design that satisfies the required specifications. The next step would be to develop an engineering model and a flight model.

Acknowledgments

The author thanks the many people who assisted with the concept and offered support all along the way.

References

[1] Karl F Strauss, "Memory Technologies and Data Recorder Design," IEEE Aerospace conference, Big Sky, Montana, p.1-15, 2009

[2] 3D Plus, "3D FN64G08VS8305 64Gbit FLASH Nand organized as 8Gx8, based on 1Gx8,"
http://www.3d-plus.com/doc/prod/2dfp_0305_2.pdf

[3] Space Micro Inc, "Radiation Hardened 8Gb NAND FlashModule," http://www.spacemicro.com/pdfs/flash_8G_v6.0.pdf

[4] Texas Instruments Inc., "Channel Link Design Guide," https://www.national.com/assets/enboards/channellink_design_guid.pdf

[5] ECSS-E-ST-50-52C, "SpaceWire - Remote memory access protocol," February 2010

[6] Texas Instruments Inc., "1.6-Gpbs to 2.5-Gbps Class V Transceiver," http://www.ti.com/product/tlk2711-sp

[7] ECSS-E-ST-50-53C, "SpaceWire – CCSDS packet transfer protocol," February 2010

Intelligent Navigation System with SpaceWire for Asteroid Sample Return Mission HAYABUSA2

SpaceWire missions and applications, Short Paper

Hiroki Hihara, Koutarou Moritani
Space Engineering Division,
NEC TOSHIBA Space Systems. Ltd.
10, Nisshin-cho 1-chome, Fuchu, Tokyo, Japan
h-hihara@bc.jp.nec.com, k-moritani@bk.jp.nec.com,

Tetsuya Masuda
Space Systems Division
NEC Corporation
10, Nisshin-cho 1-chome, Fuchu, Tokyo, Japan
masuda@jd.jp.nec.com

Ryu Funase
Department of Aeronautics and Astronautics
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
funase@space.t.u-tokyo.ac.jp

Hisashi Otake, and Tatsuaki Okada
Institute of Space and Astronautical Science (ISAS)
Japan Aerospace Exploration Agency (JAXA)
Sagamihara, Chuo-ku, Kanagawa 252-5210, Japan
ootake.hisashi@jaxa.jp, okada@planeta.sci.isas.jaxa.jp

***Abstract*—A hard real-time intelligent navigation system which adopted SpaceWire has been developed for HAYABUSA2. A deterministic communication scheme established by SpaceWire-D draft standard is applied to the optical navigation system for autonomous touch-down and take-off operation proceeded on an asteroid. An image recognition unit and optical sensor interface modules are connected through a SpaceWire router, which is used as an active backplane. Real-time optical navigation capability has been achieved in every one hertz with its natural image recognition technology and SpaceWire-D compliant transmission system.**

***Index Terms*— SpaceWire, Image Recognition, Real-time, Optical Navigation, Networking, Spacecraft Electronics.**

I. Introduction

HAYABUSA2 is an asteroid probe planned to be launched in 2014, and aims at sample-return from a C-type asteroid considered to contain organic or hydrated materials. Figure 1 shows the HAYABUSA2 prepared for the preliminary integration test.

Round trip communication time between an asteroid and the Earth is more than thirty minutes, in consequence automatic and autonomous operation by the probe itself is required for compensating the navigation operation from the ground station on the Earth in order to achieve touch-down onto the asteroid and take-off from it. Autonomous optical navigation system with LIDAR (Light Detection and Ranging), LRF (Lazar Range Finder) , and ONC (Optical Navigation Camera) are adopted for its optical navigation subsystem.

The electronics unit of ONC accommodates a real-time natural image recognition module and 64bit central processor unit (CPU) based on Space Cube2 design [1], [2], and its SpaceWire interface complies with SpaceWire-D draft standard [3], [4].

Since the deterministic implementation scheme of SpaceWire, which is the premise of real-time operation, was established by SpaceWire-D draft standard, the onboard network system for satellite bus has been replaced with SpaceWire as shown on Japanese scientific satellites and small low earth orbit observation satellites [5], [6]. The first

Fig. 1. **HAYABUSA2 asteroid probe**

HAYABUSA employed original onboard communication protocol PIM (Peripheral Interface Module transmission protocol). The second generation HAYABUSA2 inherited the technology and equipment developed for the forerunner, and we developed protocol bridges for the translation between PIM and SpaceWire. The protocol for accessing the interface of each components and the scheduling scheme is close to RMAP (Remote Memory Access Protocol) and SpaceWire-D, so the development of those protocol bridges were straight-forward.

Although the optical navigation subsystem for AOCS (attitude and orbit control subsystem) of HAYABUSA2 accommodates SpaceWire interfaces and connected to legacy onboard devices with PIM interfaces through protocol bridges, the operation scheme is the same as its predecessor. As a result the ground operation system is the same as that for fully SpaceWire compliant satellites.

II. Natural Image Recognition Module

Natural image recognition module for the optical navigation subsystem is a built-in module of an optical navigation camera electronics unit (ONC-E).

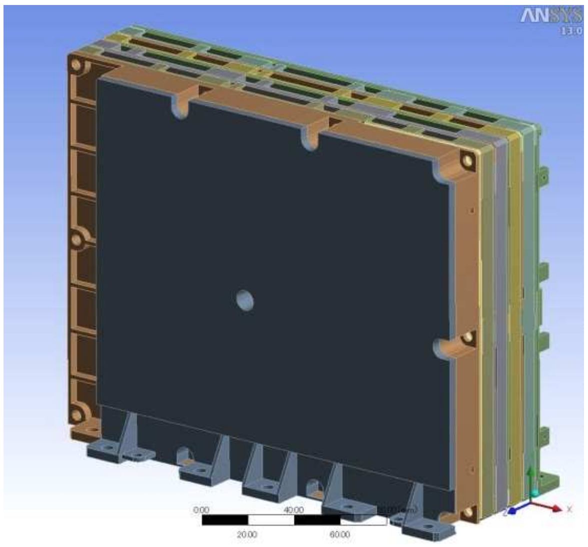


Fig. 2. The outlook of ONC-E

Figure 2 shows the outlook of ONC-E, and the technical feature of ONC-E is shown in table 1.

JAXA authorized 64bit microprocessor is used for ONC-E CPU (Central Processing Unit) module, whereas preliminary image processing is carried out by dedicated hardware implemented on ACTEL RTAX2000S FPGA (Field Programmable Gate Array). This architecture is inherited from the ONC-E of prior HAYABUSA asteroid probe in order to access local image buffer memories without accessing main memory of CPU. This architecture enables synchronous operation of the natural image recognition module with AOCS (Attitude and Orbit Control Subsystem).

TABLE I. Optical Navigation Camera Electronics Technical Features

Parameter	Value
Image recognition rate	2Hz (max) 1Hz (nominal)
SpaceWire port	Telemetry/Command: 2ch (redundant) Data recorder interface: 1ch Sensor interface: 2ch (nominal)
Conventional ports	PIM: 1ch UART (RS422): 3ch (Proprietary interfaces are included)
Memory Buffer	SDRAM: 1Gbytes (*) Flash Memory: 2Gbytes (*) (*) includes Reed-Solomon encoding
Size	95.2(W) x 230.8(D) x 177.8(H) (mm)
Mass	< 2.94kg
Power consumption	< 33.8W

We have extended the deterministic communication scheme with SpaceWire and RMAP protocol onto the inter-module communication inside the ONC-E of HAYABUSA2. SpaceWire active backplane have been implemented in order to guarantee a hard real-time performance required for the

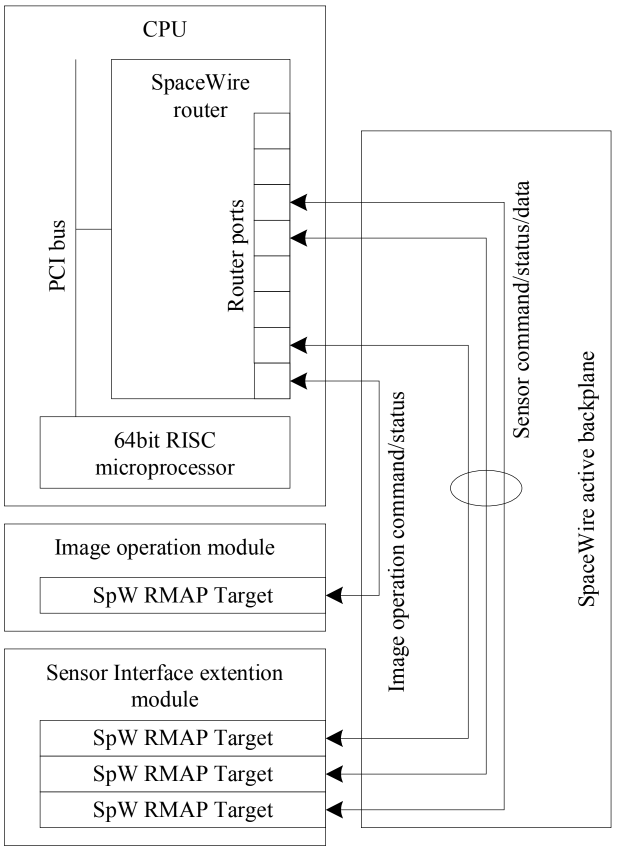

CPU
SpaceWire router
PCI bus
Router ports
64bit RISC microprocessor
Image operation module
SpW RMAP Target
Sensor Interface extention module
SpW RMAP Target
SpW RMAP Target
SpW RMAP Target
Image operation command/status
Sensor command/status/data
SpaceWire active backplane

Fig. 3. SpaceWire active backplane inside ONC-E

optical navigation subsystem.

TABLE II. IMAGE OPERATIONS EXAMPLES

Category	OPE-code	Operation
Arithmetic Operation	ADD, SUB, MULT, DIVOP	Arithmetic image operation between stored images
Compression	COMP_SPIXSEL, COMP_JP2K	Lossless/lossy compression with StarPixle® or JPEG2000
Operation Control	NOP, BUF_CLR	Do nothing, Image buffer clear operation
Transfer Control	MOVE	Transfer an image (copy)
Image processing	MEDIAN, MEAN, MODE, BIN	Calculate median, mean, mode of images or make 2x2 binning image
Image processing (Hardware operation)	HW_THRES, HW_EDGE, TMPLBL, GRV	Binary image acquisition, edge detection, temporary labeling, the center of gravity acquisition

Since a large memory buffer is mounted on the natural image recognition module, high resolution images are captured and stored simultaneously with image recognition operation. The images are used for the calibration of the result of image recognition by an attitude and orbit control processor (AOCP) as well as for scientific purposes. The images are stored in Data Recorder through SpaceWire/RMAP port.

Image operations of ONC-E are programmable using its dedicated script language. Examples of the operations are shown in table 2.

One-chip SpaceWire router is mounted on the CPU in order to configure SpaceWire active backplane system. The real-time natural image recognition module and an optical sensor interface module are connected to the router chip through SpaceWire and RMAP protocol. Each module is connected through bus connectors embedded inside the metal frames of the modules, so that physical backplanes are eliminated, which resulted in reducing the mass of ONC-E in order to meet the requirement for the deep space mission. Figure 3 shows the block diagram of ONC-E, and SpaceWire interconnections are shown.

III. REAL-TIME OPTICAL NAVIGATION SYSTEM

The image operations of ONC-E described in previous section are to be processed simultaneously along with the operation of AOCP, which processes navigation calculation. In order to synchronize operations processed by ONC-E and AOCP, the deterministic operation scheme is adopted. The scheme is compatible with SpaceWire-D draft standard, and established through the activity of the SpaceWire user's group, Japan [7]. The guideline is adopted on ASTRO-H [5] for

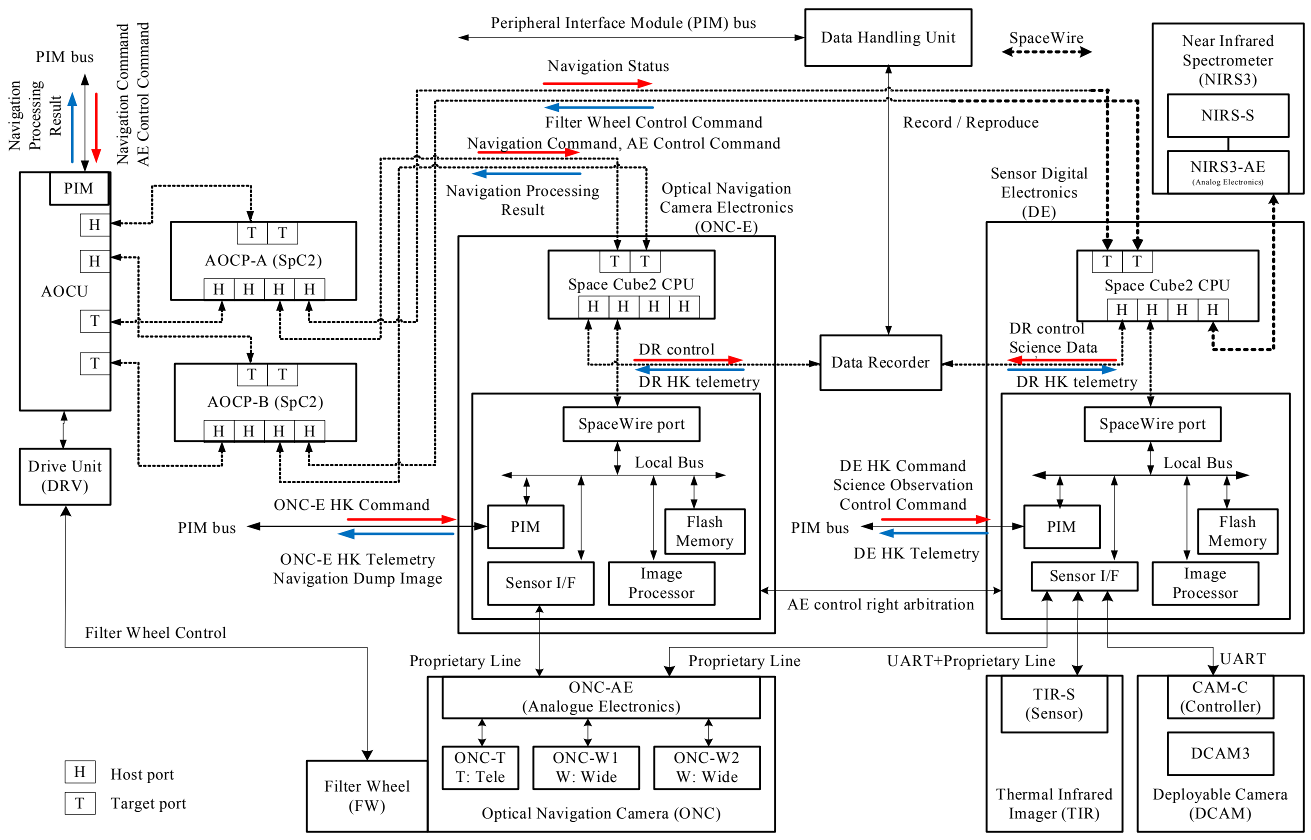


Fig. 4. Optical navigation subsystem block diagram

satellite bus communication, so the same established manner is also adopted for AOCS subsystem of HAYABUSA2.

Figure 4 shows the diagram of the real-time optical navigation subsystem of HAYABUSA2.

The communication protocol of data handling subsystem is PIM protocol, so the protocol bridge in implemented on AOCU (Attitude and Orbit Control Unit), ONC-E, and DE (Sensor Digital Electronics).

Time master of the SpaceWire network for AOCS subsystem is AOCPs. One minute is divided into 64 time slots in accordance with the guideline [7], and SpaceWire Time-Code corresponds to each time slot. The time indicator (TI) is distributed by DHU (Data Handling Unit) as 32bit value for larger than one minute. The time indicator (TI) is concatenated with the 6bit value of SpaceWire Time-Code, and 38bit value of the time code is used to synchronize all components in AOCS subsystem. In order to concatenate system time indicator and SpaceWire Time-Code, CCSDS Unsegmented Time Code is employed as proposed at SpaceWire working group in ESA/ESTEC [8].

The operation scheme enables simultaneous operations by ONC and AOCP, and real-time optical navigation is realized. There are three optical navigation functions are implemented through the deterministic communication scheme described above,

- Asteroid Image Tracking (AIT)
- Target Marker Tracking (TMT)
- Characteristic Geography Tracking (CGT)

AIT is used for the tracking of the asteroid image. The bright points in the field of view are evaluated through the criteria of evaluation based on the brightness, and the center of gravity is derived from several image of the surface of the asteroid. This function is used for processing the whole image of the asteroid.

TMT is used for the tracking of a target marker. As HAYABUSA2 is approaching the surface of an asteroid, it deploys bright target markers with fine reflectors. Once those target markers have been ejected from the satellite bus towards the surface of an asteroid, they are used for tracking. HAYABUSA2 has a flash light for detecting target markers. The target marker images of the off phase of the flash light are subtracted from the images of the on state by ONC-E, then the differential image is evaluated through the criteria based on their size and brightness. The center of balance of those bright points are calculated by ONC-E, and extracted information through the images are transferred to AOCP for identifying target markers.

CGT is used for natural image tracking identifying specific geographical image as rocks and craters. Characteristic point of view is specified by commands from ground stations or an autonomous operation programmed prior to the operation. Once the characteristic image is identified, a window is associated with the image and ONC-E starts to track the specific image within the associated window. The AOCU can specify a template for specific image is the template can be transferred to ONC-E for specific correlation calculation for identifying a specific image. The representative value of axis is transferred form ONC-E to AOCP in real-time, and AOCP use the value for navigation.

IV. Conclusion

Real-time optical navigation has been achieved in every one hertz for HAYABUSA2, which is an asteroid probe planned to be launched in 2014. Its optical navigation is realized by natural image recognition technology synchronized between an optical navigation camera and an attitude and orbit control processor through the deterministic synchronization scheme established by SpaceWire-D draft standard.

Acknowledgment

Authors thank ASTRO-H project people for their precious suggestions for implementing real-time system for asteroid probe mission.

References

[1] Tadayuki Takahashi, et al., ”The ASTRO-H Mission”, SPIE, 7732, 77320Z, 30 July 2010.

[2] Takahiro Yamada, and Tadayuki Takahashi, “Standard Onboard Data Handling Architecture Based on SpaceWire”, International SpaceWire Conference 2008, 4-6 November 2008, p.253-256.

[3] Takahiro Yamada, “Proposal for Defining Standard Services Over SpaceWire –Revision A -”, The sixteenth SpaceWire working group meeting ESTEC, Netherlands, 22 March 2011.

[4] Hiroki Hihara, Toshiaki Ogawa and Kenji Kitade, “NEXTAR: Small Satellite Bus Based on SpaceWire Deterministic Implementation”, International SpaceWire Conference 2011, 8-10 November 2011, p.344-347.

[5] Takayuki Yuasa, Tadayuki Takahashi, Masanobu Ozaki and Motohide Kokubun, “A Deterministic SpaceWire Network Onboard the ASTRO-H Space X-Ray Observatory”, International SpaceWire Conference 2011, 8-10 November 2011, p.348-351.

[6] Toshiaki Ogawa, Yusuke Kobayashi, Shoichiro Mihara, Koichi Ijichi, and Hideyuki Hamada “Outline and Progress of ASNARO (Advanced Satellite with New System Architecture for Observation) Satellite System”, 8th IAA Symposium on Small Satellites for Earth Observation, Berlin, Germany, 04 – 08 April 2011.

[7] SpaceWire User’s Group, Japan, “SpaceWire Network Design Guideline”, Version 1.0, 13 May 2010.

[8] Aeroflex Gaisler AB, “High Accuracy Time synchronization over SpaceWire Networks - update”, April 2012.

Spacewire Network in MTG Satellites

Missions & Applications, Short Paper

Alain Girard
Command Control & Data Handling Engineer
Thales Alenia Space
Cannes, France
Alain-Felix.Girard@thalesaleniaspace.com

Alain Degardin
Mission Data Handling Engineer
Eurogiciel Sophia-Antipolis
Sophia-Antipolis, France
Alain.degardin@eurogiciel.fr

***Abstract*— The system integrator has to manage many suppliers each providing specific data-source design constraints. Designing a SpaceWire network based on each of these constraints is a challenge at system level as progressive refinement and modification during development phases might impact all other data-sources performances. For instance, insuring the routing for all network resources without losing data requires to oversize all network and source's performance from the theoretical worst cases.**

Based on traffic data analysis, consolidations have been conducted to segregate all data flows of all data-sources in order to optimize links rates and buffers with respect to the useful data-rates and their required margins.

Network optimization had been checked with MOST (Modeling of Spacewire Traffic), a toolset developed by Thales Alenia Space as a SpaceWire library running on OPNET Modeler®. The simulations of the network have been realized on worst-case scenarios in order to verify the previous analytic traffic analysis. The simulation successfully confirmed the analytic analysis.

***Index Terms*—SpaceWire Networks, Traffic analysis, FDIR**

I. Introduction

MTG satellites implement a SpaceWire (SpW) network for handling the science data exchange between payload and platform (mission data). Mission Data packets are provided by four different sources, two instruments (Flexible Combine Imager (FCI) and Lightning Imager (LI)), a RF payload called Data Collection Platform (DCP) and the Satellite Management Unit (SMU). MTG mission data handling main characteristics are the following:

- Continuous acquisition of formatted data according to Packet Utilization Standards (PUS)
- Continuous real time download of data to the ground station from the geostationary orbit
- Fast configuration of FCI instrument with 8,3Mbytes scanning tables
- Internal SpW network in the FCI instrument with several sources routed to the satellite SpW network through a SpaceWire switch
- Fully cross-strapped network to avoid any single point of failure and maximize the network reliability.

As shown in Figure 1, the Payload Data Downlink system (PDD) which holds the satellite SpaceWire routing switch is the merging point of every source (FCI, LI, DCP, SMU).

The first hypothesis was made to cope with bottlenecks possibly occurring at the SpW router side of the PDD. In order to minimize this network access time and the corresponding buffer sizes, all the link rates were specified to the highest rate, 200Mb/s in compliance with the maximal 10X-SpW router theoretical capability. The second hypothesis was that each source sends its packets at maximal rate without insertion of NULL between data characters thanks to the preparation of its packet in an emission buffer.

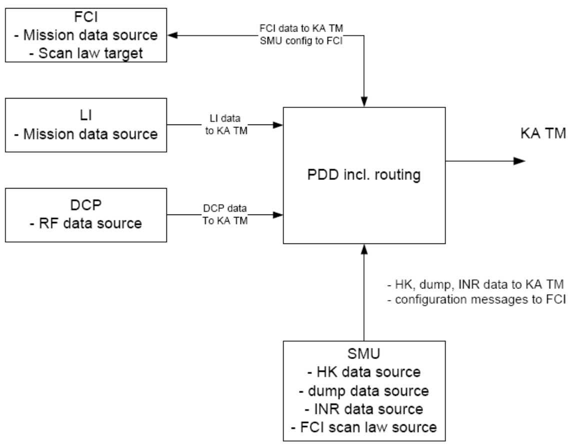


Fig. 1. MTG spacewire network presentation

This approach was considered not suitable because:

- It was difficult to implement the maximal link rate due to severe skew and jitter constraints
- Bottlenecks between instruments creates a strong coupling between multiple sources manufactured by different suppliers, which can create contractual issues. Traffic from each instrument had to be partitioned to avoid this issue and facilitate the network constraints break-down per units.
- Due to the FCI instrument internal SpaceWire network including different link rates, the second requirement (no Null characters interleaved within packets) was not verified. For instance, a source from

the FCI is emitting at 50MHz inside a network running at higher speed. Even if no Null character is inserted on the 50MHz link, they are inserted on the higher speed link to maintain the synchronization

In order to remove contractual and technical risks, the mission data handling using the SpaceWire network has been consolidated taking into account partitioning of the different SpaceWire sources. This segregation has allowed optimizing link-rates for all sources. FCI SpaceWire internal network has been taking into account during the analysis. Analytic occupation budgets on each link has been computed and confirmed later by MOST simulations taking into account satellite and FCI internal SpaceWire network. Initialization and stop sequences, Fault Detection Isolation and Recovery (FDIR) for SpW network have been specified in the frame of the MTG project taking into account its specificities.
Finally, skew & jitter budget has been apportioned on receiver and transmitter sides taking into account current skew & jitter budget on FCI and PDD side.

II. SpaceWire consolidation and link-rate relaxation

Mission data flows have been segregated to ensure that no direct blocking from an instrument to another occurs on the satellite SpW network to avoid taking into account the impact of traffic congestions from an instrument to another due to the routing through the network. To perform this segregation, it has been decided to feed in parallel the Virtual Channel Assembler (VCA) of the PDD system with four “links”, one dedicated to each source (FCI, LI, DCP, SMU).
The merging of each source is ultimately performed by the Virtual Channel Multiplexer (VCM) which merges the different Virtual Channels allocated to each source on the Master channel. As the global amount of data able to be downloaded through the Master Channel is higher than the nominal (mean) data input, the bottlenecks can be simply managed by adding buffers on the VCA input. Hence we considered in the analytic approach that each source is independent from the other to size each SpaceWire link rate.

Each SpW link has been relaxed to fit with the real needs and to maintain acceptable link margins with 50% as objective and 25% as minimum. On MTG, SpW links are specified to use the same link-rate on both directions.
FCI SpaceWire internal network uses the following link-rates:
- 120Mbit/s for two of FCI internal links
- 50 Mbit/s for one of the FCI internal links.

On MTG, the satellite SpaceWire network uses the following link-rates:
- 140Mbit/s for the FCI
- 100Mbit/s for the LI and the RF payload
- 50 Mbit/s for the SMU

Contrary to the first approach, the SpW network has now heterogeneous link-rates. According to SpW standard and SpW router behavior this impacts the method used to compute the link margin.

III. SpaceWire network analysis and MOST Simulation results

FCI aggregates several sources into the output link to the DDU. As the FCI internal SpaceWire network has heterogeneous link-rates and according to the SpaceWire standards, the margin on output link could not be easily computed. If packets are forwarded from a 50Mbit/s link to a 200Mbit/s link, NULL chars will be interleaved between each data characters. As an output port is blocked to a specific source during packet forwarding, this interleaving of NULL char wastes some part of the Bandwidth. The result is the same when the destination link has a lower link-rate than the input link, flow control will be used to slow down the input flow, thus NULL char will be interleaved.

On MTG, it has been decided to compute useful margin, i.e. margin that could be used to increase the data flow. Thus, NULL char interleaved inside a packet should not be taken into account as they cannot be used to carry data.

As a first approach, the minimum time to transfer a packet from a source node to a destination node across a SpaceWire router is equal to the time required to transfer the packet across the slowest link used.

Equation 1 allows computing the time “$T_{transfer}$” taken to carry a packet of size P flowing through a router from a link L1 with link-rate C1 to another link L2 with link-rate C2

$$Ttransfer = \frac{p}{\min(C1,C2)} \qquad (\square)\square$$

As the time to transfer a packet is constant through the network, it is possible to estimate the equivalent size of a packet in each link of the network according to the time needed to transfer the packet and the link-rate.
Equation 2 allows estimating the equivalent packet size P’ on the link L2 of a packet of size P flowing through a router from a link L1 with link-rate C1 to another link L2 with link-rate C2.

$$P' = P * \frac{C2}{\min(C1, C2)} \qquad (2)$$

Equation 3 allows evaluating the equivalent mean data-rate DR’(according to the equivalent packet size) of a flow F1 with a data-rate of DR flowing through a router from a link L1 with link-rate C1 to another link L2 with link-rate C2.

$$DR'' = DR * \frac{C2}{\min(C1,C2)} \qquad (3)$$

As a router can merge several flows into a single output port, it is necessary to sum-up each input data-rate to going on a single output port to get the actual link margin.

Equation 4 allows calculating an estimation of the available margin "Co_{margin}" on the link 0 for n input flows with DR_i Data-rates from n links with link rate C_i merged to the link 0 with a link rate C_o.

$$Co_{margin} = Co - \sum_i^n \left(DR_i * \frac{Co}{\min(Ci,Co)} \right) \quad (4)$$

This latter formula has been taken into account for link margin estimation on MTG SpaceWire network. This estimation is based on the minimum required time to transfer a packet without taking into account possible delays in the router. (Routing delay, delay through the router etc...) Furthermore, the required bandwidth for SpaceWire Flow control has not been taken into account in this estimation. Thus it is clear that the margin is slightly overestimated, but considered correct at first order.
However, concerning flow control, as the mission data streams are mainly from instruments to the PDD, flow control tokens are principally from PDD to instrument thus flow control token are not added to mission data flow. The only Flow control token stream is in the same direction as the mission data and has to be taken into account, is the one corresponding to the FCI mission configuration sent from the platform (SMU) to the instrument. Lower margin during this transfer is acceptable as it is a transient state.

MOST simulations have been defined in order to check if network specification is correct and to confirm the engineering traffic analysis at satellite level.
MOST simulations have been done on the complete SpaceWire network of the MTG mission including the overall satellite network and the FCI actual internal network. As the aim of the simulations is the validation of the nominal state of the network, only the nominal part of the actual nominal & redundant cross-strapped architecture flows have been simulated. These simulations were performed with a worst case scenario with:

- Data acquisition in parallel to the sending of FCI configuration table
- All biggest packet of each source sent at the same time to stress the VCM
- FCI data transfer is done in character forwarding mode (null characters are interleaved within a packet transfer instead of packet forwarding in which the FCI network has to store and then forward the packet without interleaving of NULL char)

Contrary to the margin estimation with analytic solution, MOST simulations allow computing the margin in worst case taking into account the hardware performance and the flow control tokens. That will lead to a worst case margin taking into account the full SpaceWire characteristics.

In MTG there is a requirement for maximum delivery delay between acquisition and availability on ground, thus MOST simulations results have been used to estimate the End to End delay (ETE) between source and destination node in worst case (All sources sending biggest packet at the same time). As all sources are merged on a single RF downlink, the MOST simulations simulated the nominal output data-rate of the PDD in order to check if the mission data network is correctly designed. MOST simulations have been defined to be as relevant as possible to the embedded mission data network and have allowed simulating the satellite network behavior during short interruption and recovery. Moreover, the VCM input buffers have been under-sized in MOST to worsen the traffic analysis and check the impact on the communication from each instrument.

Table 1 compares the available margins computed with MOST simulations and analytic analysis. As expected FCI link margin calculated with MOST simulations is lower than the analytic one as flow control tokens and delay in SpaceWire router(s) are taken into account by MOST. However analytics results are really close to the simulated ones thus we considered that, as a first approach, the proposed analytics computation is validated. Note that in the below table in which both margins are compared, analytic figure (for LI and DCP) may be slightly pessimistic, lower than the one from the simulation, because of short bottleneck terms that speed up the transfer when recovered.

TABLE I. MARGIN RESULT WITH MOST SIMULATION AND ANALYTICS SOLUTION

Link source		**Available margin**	
Source	***Link-rate***	***Analytics calculus***	***MOST simulations***
FCI	140 Mb/s	28,80%	28,00%
LI	100 Mb/s	62.34 %	62,50%
DCP	100 Mb/s	53.97 %	54,10%

IV. SPACEWIRE NETWORK MANAGEMENT

SpW logical addressing without header deletion is used on MTG instead of physical addressing. That allows having a fixed address for each destination whatever the current network configuration. Then, only the router configuration has to be updated when the network configuration is changed for FDIR purpose to use redundant equipments. As all the network is cross-strapped and only one link is enabled at one time, it is also possible to use Group Adaptive Routing (GAR), hence as only one link of the group is active at one time, we can switch the network configuration without having to reconfigure the routing table. Use of Group Adaptive Routing is still optional defined in the project and shall be consolidated in future analysis.

PDD interfaces are configured in auto-start in order to accept any communication started by mission data sources. This

allows commanding only source side of the link and managing easily redundancy at source level. In order to start a communication, the platform computer has to command sources to enable their interfaces. This is performed in MTG using an independent Command/Control link. The disconnection is also very simple by commanding source to disable its interface.
As SpaceWire link starts hand-shake procedure at a slower link-rate than the nominal one, both sides of the link have to alter their SpW emitter link-rate to the nominal link rate autonomously after the connection success when reaching run state. This alteration has to be done at each reconnection.
SMU is in charge of monitoring the SpaceWire network health according to received telemetry packets and event packets from instrument and payload.

The following FDIR detection and isolation actions are performed on-board:

- level-1C for link transient disconnection detected at source level and recovered autonomously by the source
- level 2 for persistent disconnection managed at source and satellite level

As described in the SpW standards, any error on a SpaceWire link leads to a disconnection of the link. A parity check could lead to a disconnection from the receiver side, as showed in the SpW connection process. After a disconnection, it is necessary to request another start of the link to reconnect. In case of transient error, the reconnection will be quick and then it is better to let the source manage autonomously the recovery of the error.

FDIR level 1C has been defined to try to reconnect a disconnected link when detected by the source without intervention of the SMU. In case of successful reconnection, sources shall inform the platform computer that a disconnection has been detected and autonomously recovered.

Since satellite safety is not endangered by a SpW failure, all recovery for more severe errors are planned to be performed by ground. Thus a fail-safe FDIR approach has been defined for level 2.
If a link disconnection is not recovered 100 ms after the detection, (Reconnection without any disturbance could be done in few microseconds, thus 100 ms of disconnection means a permanent anomaly on the link) the failed source shall report to the SMU this problem and then a fail-silent approach shall be performed.

In case of FDIR level 2 the platform computer will isolate the failed link by disabling the two SpW link interfaces. (Source and Destination). As the mission data link is not available, the failed source is reconfigured in standby mode, and then the ground is informed with an anomaly report. During ground investigation the other sources of the mission continue to send their mission data without any impact.

V. Skew & jitter apportionment

One of the critical point for 200Mb/s SpaceWire link was the electrical performance in terms of skew and jitter required at Emitter and Receiver sides to allow correct transmission. To ensure that SpW network will work properly, the electrical level has also been consolidated in compliance with the proposed link-rate relaxation.
As detailed in the SpW standards, the maximum data signaling rate that can be achieved is different from one system to another (Pending on device and harness), limited by skews and jitters. Hence Skew & jitter performance requirement has been built for both Emitter and Receiver. This budget allocation has been needed to certify that any interface which is compliant with this requirement will be able to communicate with any other interface at the MTG maximum SpaceWire signaling rate.

Skew and jitter at Emitter side are merged into a single value which is called Encoder contribution. Encoder contribution shall not exceed 1,4 ns.
As shown in SpW standards, T_{ds} is defined as the minimum specification for the separation of consecutives edges on Data and Strobe signal at the input of the decoder. In MTG T_{ds} shall not be above 3,3 ns.
On MTG, maximum link-rate is 140Mb/s for FCI which leads to 2,44 ns for both harness contribution and margin.

VI. Conclusion

Network has been consolidated and optimized, with suitable contractual management, correct electrical, data-handling designs and FDIR design.
The skew and jitter budget has been specified for Emitter and Receiver and is in line with current SpaceWire chips used in MTG and current maximum link-rates.
The new approach for link margin computation and engineering spacewire network analysis has been confirmed by MOST simulations.
Advanced validation of SpW interfaces for each protocol layer will be performed by coupling payload and platform representative test benches, in order to secure AIT sequences.

References

[1] T. Ferrandiz,, “Maîtrise des latence de communication dans les réseaux bord spacewire,” thesis delivered the 2 march 2012 by « Institut Supérieur de l’Aéronautique et de l’Espace ».

[2] European Cooperation for Space Standardization, SpaceWire – Links, nodes, routers and network, ECSS-E-ST-50-12C, 31 july 2008.

[3] European Cooperation for Space Standardization, SpaceWire – CCSDS packet transfer protocol, ECSS-E-ST-50-53C, 5 February 2010.

The Swift Codec Development

Session: SpaceWire Components, Short Paper

David Juliusson, ASIC design
RUAG Space AB
Göteborg, Sweden
david.juliusson@ruag.com

***Abstract*— SpaceWire is widespread as data interface for flight missions. In order to keep up with the demands from both projects and test platforms a new, fully configurable, SpaceWire codec called Swift has been developed by RUAG Space AB. The codec is easy to integrate into projects, thanks to its internal data buffers and standardized interface. The codec features self calibrating timeout counters, reducing the need for configuration pins or register accesses prior to link initialisation. In order to improve timing and reach high data rates in both ASIC and FPGA technologies, the codec has small and uncomplicated receive and transmit regions. The Swift codec has been fully tested and validated and has so far been incorporated into several ESA funded and commercial projects.**

***Index Terms*— SpaceWire, codec, Swift, component**

I. Introduction

RUAG Space AB has developed systems and components that utilize the SpaceWire interface for years. During this time a couple of different SpaceWire codecs have been used, each with their own merits and drawbacks. As more and more projects use the SpaceWire interface, RUAG Space identified the need for a single solution that is both easy to use for designers and adaptable to suit the needs of different projects.

Furthermore we saw the need for a SpaceWire solution that could be used in different test setups without the need to incorporate more people than the engineers needed to assemble the system.

A. Outline

This paper starts with stating the design goals for the Swift codec development followed by a description of the development process. The next chapter explains the design partitioning of the Swift codec as well as the main features of the new design. The last chapter holds the conclusion.

II. Design goals

Going into this project our aim was to develop a codec that was adapted to the needs of our everyday use of the SpaceWire interface. This would be done by developing a SpaceWire codec that incorporates our experiences of the interface so far and focusing on the issues we would like to address.

Fast turnaround times for test setups as well as easy instantiation were key factors when designing our new SpaceWire codec. This brings us to the main design goals for the new codec; these were twofold and primarily aimed at usability and efficiency.

A. Usability

The codec should be easy to integrate into projects, reducing the manual steps when performing detailed ASIC and FPGA routing of the critical receive and transmit data paths.

The codec should also be a self contained entity and provide an easy to use standardized system interface. This means that the codec should handle all SpaceWire protocol specific characters and be able to initialize and sustain a link without interference from supporting logic.

B. Efficiency

The codec should be configurable to suit different projects. In low bandwidth configurations the codec should be small enough to be suitable for remote terminal applications. In high bandwidth configurations it should have the capability to reach the preferred 200 Mbps data rates in all commonly used ASIC and FPGA technologies.

III. Development

The development of the Swift codec was done in two phases; the prototype phase and the industrialisation phase, i.e. making the codec ready for flight missions.

A. Prototype phase

During the first phase a prototype was developed as part of a master thesis [2]. The goal of the thesis was to design and develop a new SpaceWire codec and in the process learn more about the SpaceWire protocol and how to efficiently implement it in hardware.

The prototype phase enabled us to test design ideas, in a fast and inexpensive format, and to optimize the design from a system point of view early in the process.

The design proposed in the thesis turned out well and proved easy to integrate into projects. Initial tests also showed that the design was both small and fast enough to be able to provide high data rates in both FPGA and ASIC technologies.

The prototype was verified enough to make us confident that the design concept worked. The prototype was also subject to rudimentary live tests against both a well known, validated SpaceWire codec as well as connected in loopback. The test setup can be seen in Figure 1.

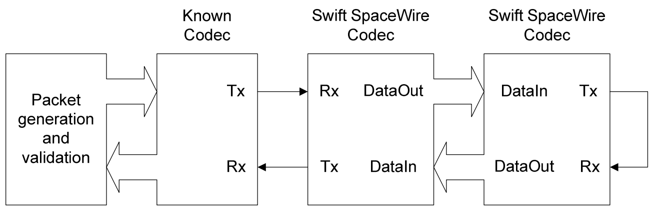


Figure 1 Prototype test setup

The thin arrows in Figure 1 represent the Data/Strobe connections and the thick arrows represent the internal data interface within the system clock region.

B. Industrialisation phase

During the second phase the design was finalized and a full industrialisation procedure was performed. The documentation as well as the specification was updated to incorporate findings from the prototype phase.

A codec was subjected to a complete functional verification of the design to ensure that the codec complied with the documentation. After the verification procedure was complete the Swift codec was validated to ensure that the codec is suitable for space missions.

The codec has been validated using both commercial products and certified flight hardware. The commercial products used during the validation test where;

- 4Links RG401/8 [3]
- 4Links RG408-LS [4]
- Star Dundee SpaceWire Conformance Tester [5]

The Swift SpaceWire Codec has passed the tests conducted with all the above mentioned commercial tools. Tests have also been conducted using certified flight hardware developed both in-house and externally.

IV. SWIFT DESIGN

The design partitioning of the Swift SpaceWire Codec differs from the example design in the [1] Clause 8 in some aspects. The subchapters below will go through each of the major differences and give our reasoning for the decisions. The first subchapter gives an overview of the design partitioning within the Swift codec.

A. Design partitioning

The Swift design partitioning can be seen in Figure 2. A walkthrough of the logic function of each block follows in the subchapters below.

1) Control logic

The structure labeled Control Logic in Figure 2 is responsible for handling the link initialization and link error recovery procedures [1] Clause 8. The Control Logic also functions as a hub for control and status signals inside the Swift codec, relaying them to and from System. The Control Logic also connects the functions of the receive and transmit data paths that needs synchronization i.e. the reception and transmission of flow control tokens.

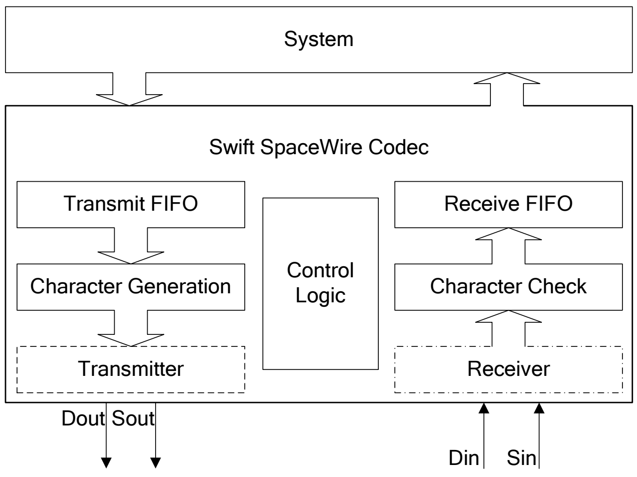


Figure 2 Swift design partitioning

2) Transmit data path

Tracing the transmit data path, starting from the top of Figure 2; The Swift SpaceWire Codec provides the System with a standardized data interface. The structure labelled Transmit FIFO contains the data buffer needed to store the bursts of data from System. End of packet characters are automatically added by the codec at packet limits. The size of the buffer is configurable depending on system specifications. The structure labelled Character Generation contains the logic that generates SpaceWire L-Chars when commanded to by Control Logic. This block is also responsible for making the N-Chars ready for the Transmitter upon arrival of flow control tokens. The Transmitter in the bottom left corner, handles serialisation of the data bound for the link as well as parity and strobe generation.

3) Receive data path

Next we trace the receiver data path, from the Data/Strobe, Din/Sin in Figure 2, towards System. The Receiver samples the data and feeds the data flow directly to the structure marked Character Check. This is done without first decoding the received characters. The Character Check serializes the data stream, decodes the characters as well as performs the parity check. All L-Chars are handles by the Control Logic and the N-Chars are transferred to the Receive FIFO for credit check and storage. The size of the Receive FIFO is configurable to allow for stutter free data transfer up to at least 200 Mbps.

4) Clock regions

All structures except the Transmitter and the Receiver, marked with dashed and dash-dotted lines in Figure 2, are synchronous to the system clock. The transmitter region can be configured to use the system clock in small applications where size considerations are more important than transmit rates. The receiver is always asynchronous to the system clock.

B. Configurable data rates

In order to make the Swift codec a general purpose design, the codec is fully configurable with respect to the bandwidth needs of the project. This enables the designer to trade bandwidth for size when optimizing the system.

When opting for bandwidth, the codec can receive as well as transmit one useful byte, i.e. 10 bits, each system clock cycle. These bits can be any combination of SpaceWire characters.

When opting for size, the Swift codec consumes less than half the digital logic needed in high bandwidth applications. Asymmetrical receive and transmit rates are allowed if the application demands it.

Table 1Resource usage in RTAX FPGA

Size vs. performance	**Resource usage in RT AX2000**		
BAUD rate with 40 MHz System clock	***Combinational Cells***	***Sequential Cells***	***Total Cells***
80 Mbps	1339	495	6 %
160 Mbps	2447	758	10 %
200 Mbps	2691	835	11 %
400 Mbps	3581	1012	15 %

The data in Table 1 is gathered from synthesis of the design. As seen in the Total Cells column of Table 1, the size of the Swift codec scales almost linearly with the bandwidth needed.

So far successful place and route have been performed up to 200 Mbps in RT AX2000. The last entry seen in Table 1 will not be able to transfer data at 400 Mbps due to the limitations of the RT AX2000 technology. This configuration is rather an option when high bandwidth is needed in slow system clock implementations.

C. Self-calibrating timers

The Swift codec features self-calibrating timeout counters. This means that the codec does not need to be configured by the application to know the length of the timeout times.

Instead the codec uses the well defined bit rate during start-up to calibrate its internal timeout counters. This is to avoid forcing the codec to be pre-configured at instantiation, or having to rely on configuration through strapped pins or configuration registers.

This feature saves configuration pins or register accesses in preconfigured off-the-shelf products, enabling us to use the same build for different system clock speeds.

D. Internal data buffers

In order to make the Swift codec a self contained entity, a few changes to the design proposed in the SpaceWire standard was made. All data buffers needed to sustain the SpaceWire link are codec internal, as is the logic responsible for handling the exchange of flow control tokens.

These features, together with a system clock synchronous data interface, make the Swift codec a self contained entity that can connect and sustain the link without interference from supporting logic. This means that all system level functions need only to handle packets of data and time codes, all link specific characters like EOP and EEP are handled by the codec.

E. Clock regions

In order to facilitate synthesis and reach high data rates, special care has been taken to keep the fast running receive and transmit regions as small and as uncomplicated as possible. No advanced decoding or time consuming decisions are performed in the fast running receive and transmit clock regions. Instead all time consuming decisions are performed in the slower running system clock region.

This means that there are three separate clock regions inside the Swift codec. All interconnections and control signals are running via the system clock region and all asynchronous interfaces are codec internal for ease of use by designers.

The region running on the receive clock only contain around 25 registers in a normal instantiation. The transmit region is substantially larger than the receive region and contains roughly four times the number of registers. This is caused by the buffers for the asynchronous interface residing in this region. However the timing paths are short and transmit rates in excess of 200 Mbps are possible in all commonly used technologies.

The decision to divert from the design proposed in the SpaceWire standard [1] Clause 8 was taken since the design is hard to optimize without centralizing all SpaceWire functionality inside the codec.

V. CONCLUSION

The Swift SpaceWire codec has so far been successfully incorporated into systems and IP-cores for ESA funded as well as commercial projects. The codec is used daily in in-house test equipment and has completed and passed all validation tests we have conducted. The design can handle data rates of over 200 Mbps in all commonly used ASIC and FPGA technologies and has proved to be easy to integrate into projects. The Swift codec has fulfilled the design goals initially set up for the project and is in use in flight programmes.

REFERENCES

[1] ECSS Secretariat, SpaceWire – Links, nodes, routers and networks, ECSS-E-ST-50-12C, July 31, 2008, Noordwijk, The Netherlands.

[2] D. Juliusson, Development of a SpaceWire Interface in VHDL, unpublished, Master thesis, Chalmers, Gothenburg, Sweden

[3] 4Links product brief ESL-RG401/8, http://www.4links.co.uk/spacewire-products/product-briefs/4Links-EtherSpaceLink-ESL-RG401-8-product-brief.pdf.

[4] 4Links product RG-408-LS, http://www.4links.co.uk/index.html.

[5] STAR-Dundee, SpaceWire Conformance Tester, http://www.star-dundee.com/sites/default/files/SpaceWire%20Conformance%20Tester.pdf.

Components (Short)

High performance SpaceWire RMAP/DMA engine for the CASTOR microprocessor

SpaceWire Components, Short Paper

Chris McClements, Steve Parkes, Albert Ferrer,
Alberto Gonzalez-Villafranca
STAR-Dundee Ltd
Dundee, UK
chris.mcclements@star-dundee.com, steve.parkes@star-dundee.com

***Abstract*— CASTOR is a new radiation tolerant SPARC V8 processor chip which is currently being developed by Atmel in partnership with STAR-Dundee. The chip is implemented on a 90 nm radiation tolerant process which will deliver an expected processor clock speed of 200 MHz. The CASTOR chip is targeted at data processing and instrument control applications, and will deliver functional improvements over previous SPARC processors. The chip has eight SpaceWire interfaces running at 200 MBits/s, a CAN bus interface and IEEE 1553 bus interface.**

At the core of the CASTOR chip is a number of dedicated high performance SpaceWire Remote Memory Access Protocol (RMAP) and Direct Memory Access (DMA) engine's connected to the SpaceWire interfaces through a SpaceWire router. Each SpaceWire engine is capable of acting as an RMAP target, RMAP initiator or as a general purpose SpaceWire packet transmitter and receiver between the SpaceWire network and packet data defined in internal memory. Dedicated SpaceWire DMA channels are used to ensure software involvement in SpaceWire packet generation and reception is kept to a minimum. The SpaceWire interfaces support the SpaceWire-D protocol used for guaranteed latency and deterministic packet delivery. In conjunction with the RMAP initiator the chip can rapidly be configured as a highly capable SpaceWire-D initiator.

The chip can act as an RMAP target, initiator or both. The RMAP target provides a mechanism to allow remote access to the internal memory space. Two modes of operation are supported to allow direct access to a pre-defined area of memory or controlled access using authorisation by software. The RMAP initiator uses information stored in internal memory by the application software to access remote memory in equipment connected to the SpaceWire network. The engine is capable of initiating a number of RMAP transfers from remote memory, either writing data from internal memory to a remote memory location or receiving data from a remote memory location and writing it to internal memory, then interrupting the host when all transactions are complete.

The DMA channels allow the application software to send and receive data packets using data structures defined in internal memory. Each SpaceWire engine has a number of DMA channels which can operate independently of each other.

***Index Terms*—SpaceWire, CASTOR, RMAP, Sparc V8**

I. Introduction

An onboard SpaceWire [1] system comprises a number of SpaceWire nodes and routers connected together through high speed serial links. The nodes on the SpaceWire network can be sensors, mass memories and processing units. CASTOR is a new radiation tolerant SPARC V8 processor chip which is currently being developed by Atmel in partnership with STAR-Dundee.

The CASTOR chip has eight SpaceWire interfaces to facilitate communication over the SpaceWire network. The application software running on the processor has access to a number of dedicated high performance SpaceWire Remote Memory Access Protocol (RMAP) [2] and Direct Memory Access (DMA) engines to provide RMAP and application specific packet generation and reception without excessive processor workload.

II. Features

The CASTOR chip has dedicated RMAP target and initiator hardware which offloads RMAP packet generation and checking from the processor. The target can be configured to allow a remote unit to read and write memory locations inside the processor memory space without interrupting the host software. The initiator facilitates access to remote memory spaces through RMAP protocol commands and offloads multiple transaction generation and reply packet checking from the processor.

A multi-channel DMA packet transmission and reception controller is available to the processor to send and receive data through a SpaceWire router. The DMA channels are optimised to support high throughput of SpaceWire packets with minimal interruption of the processor. Generation and checking of CRC-8 and CRC-16 checksums are supported by the DMA channels.

Packets are routed to the SpaceWire network through an eight port SpaceWire router. This allows the CASTOR chip to connect too many peripherals and also act as a routing device.

Protocol support is provided for the SpaceWire-D deterministic data delivery protocol [3], the SpaceWire plug and play protocol [4], multiple time-code counters and distributed interrupt time-codes [5].

III. System Architecture

The system architecture is defined in Fig. 1.

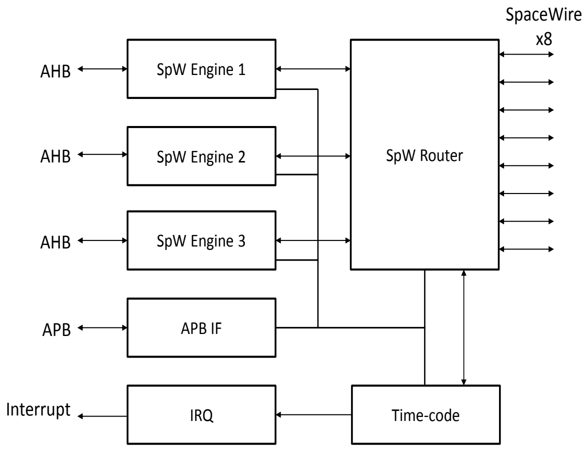


Fig. 1. System architecture

The SpaceWire engines contain an RMAP target [6], an RMAP initiator and a multi-channel DMA controller. Each engine facilitates packet generation and checking of RMAP and DMA transfers between the processor and the SpaceWire router, offloading the processor for other tasks. The SpaceWire router [7] has 8 SpaceWire ports running at 200 MBps and 3 internal FIFO ports for connection to the engines. The routers internal configuration port, port 0, facilitates configuration of the internal registers through RMAP or Plug and Play. The APB interface is used to configure and read status registers from SpaceWire engines, time-code controller and SpaceWire router. The interrupt controller provides event notification to the host processor for packet, time-code and error events which occur. The time-code controller implements time-code forwarding and distributed interrupt forwarding.

IV. SpaceWire Engine

The CASTOR chip has three SpaceWire engines which can act as an RMAP target, an RMAP initiator and to transmit and receive data from internal memory through a multi-channel DMA controller. The engine performs memory accesses through an AHB master interface and is configured through an APB interface.

The SpaceWire engine architecture is shown in Fig 2. The engine is comprised of a protocol multiplexer which connects to the SpaceWire router, an RMAP target, an RMAP initiator, a multi-channel DMA controller, an AHB interface and an APB interface.

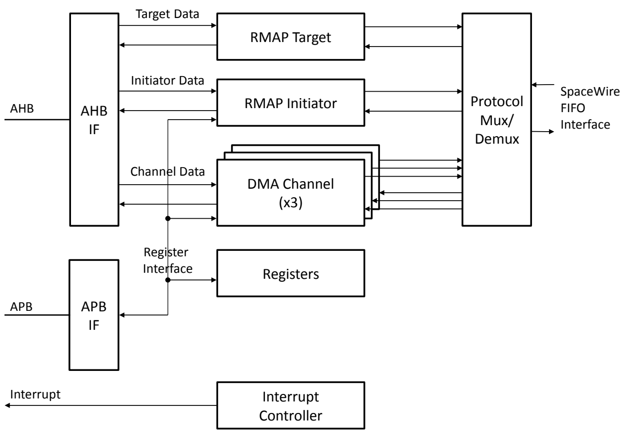


Fig. 2. SpaceWire engine

A. Protocol Multiplexer

When sending, packets to the SpaceWire Router, the multiplexer selects the next packet to be sent and waits for the end of packet before selecting the next packet to be transmitted.

When receiving, packets from the SpaceWire Router, the protocol de-multiplexer checks the first four packet bytes against a configurable pattern and mask to determine the destination of the packet, either RMAP target, RMAP initiator or a specific DMA channel. The pattern and mask are programmable by the host processor through the APB registers.

The protocol multiplexer allows multiple destination nodes or multiple protocols to be handled by the DMA channels. A packet received at a node which conforms to the ECSS-E-ST-50-51C [8] standard will have a leading logical address byte and a protocol identifier byte, followed by the packet cargo bytes and an end of packet. The protocol multiplexer transfers data packets from the RMAP target, initiator and the DMA channels into the SpaceWire FIFO. Arbitration is performed between the channels using a fair arbitration scheme where each packet source takes it in turn to transmit packets.

B. RMAP target

The RMAP target accepts RMAP commands from a remote system, performs read and write memory access commands over the AHB bus to system memory and returns an optional reply packet to the remote system. The target supports all RMAP commands with the option of limiting the commands which can be performed by configuration from software. A 16 byte verified write buffer is provided to support verified write commands.

An RMAP command received by the target is required to be authorised before it can access system memory. The processor can configure the RMAP target to act in two modes of operation.

The first mode requires the host processor to authorise commands through the APB register interface. Authorisation is requested using the Interrupt output of the core. The host software should read all the authorisation fields and then decide if the command is valid by authorising the command through the RMAP target command register. When the target has completed the RMAP command it will interrupt the host processor again with the notification status.

In the second operation mode the host processor sets which RMAP operations are authorised and the address range in which RMAP commands can operate. Any command which is performed outside of the address range or other authorisation fields is not authorised and recorded as an error.

C. RMAP initiator

The initiator uses the RMAP protocol to write data from system memory to a remote system, or read data from a remote system and place it in a pre-defined area of memory. The initiator can be used by the processor to collect data from remote targets into system memory and check the data received. The initiator uses RMAP transaction specific data structures in memory to control the command type and command fields which will be used to generate the RMAP packet. A transaction table is stored in memory to facilitate the transmission of multiple command packets before the replies for those commands have been received. The initiator validates all reply packet fields against the expected fields stored in the transaction table. If an error occurs the error is recorded and the reply packet is not acted upon.

Before the initiator can be used to send RMAP commands it must be given space in system memory to store outstanding transactions. An outstanding transaction is required to tell the initiator where in memory it should store reply data and notification status

The initiator is split into three separate entities: the encoder, decoder and timeout checker. Each of the initiator entities can operate in a different mode. The encoder and decoder have three modes of operation: notification mode, list mode and watchdog mode (modes 1, 2 and 3). The timeout checker has two modes of operation: notification mode and passive mode.

Encoder/Decoder modes:

In mode 1, notification mode, the initiator waits for the host software to respond to each initiator command sent and reply received before continuing. This mode is suitable for hosts which wish to know when commands are sent or received and process the command data and status immediately.

In mode 2, command list mode, the initiator can send a number of commands or receive a number of replies before the host software is notified. The status for each command and reply is stored in a transaction defined notification area of memory. The host can check the command/reply status after the command list has been completed.

In mode 3, watchdog mode, the initiator can send a number of commands and receive a number of replies while the host is waiting for a timer to expire or another interrupt/event to occur. The host uses the timer, or other interrupt/event, to check if the commands have completed and the status of each command. This mode is useful when the host needs to know if the commands have been sent within a defined time period but does not need to check the operation status until the time period has expired.

The initiator implements an optional timeout counter for each outstanding transaction. When a reply is not received within the timeout period the transaction will be discarded and an error recorded.

Timeout checker modes:

In mode 1, notification mode, a transaction which times out, reply not received within the selected timeout period, will cause the notification bit in the status register to be set. The notification bit is acknowledged by the host software before the initiator can perform any further operations.

In mode 2, passive mode, a transaction which times out will be deleted from the initiator table and no notification will be generated. The timeout status will be recorded in the transaction defined notification area of memory.

D. Transmitting packets using the DMA channel transmitter

The DMA controller supports multiple concurrent TX channels which can be programmed to send one or multiple SpaceWire packets continuously. Channels can be disabled and enabled at any time, affecting the data rate of the corresponding channel without producing data loss. This allows a simpler implementation of MAC algorithms by software.

A packet consists of one or multiple data chunks stored in different memory locations. This allows the packet header to be stored in a different location that the packet data content. Sending of PUS [9] packets is supported by providing the hardware computation of its CRC-16. Continuous transmission of packets is provided using circular buffer architecture with data and packet descriptor pointers. Interrupts can be set to monitor the progress of transmission of packets without halting the actual operation. This makes it possible to achieve the maximum SpaceWire data rate with minimum CPU utilization. Errors in one channel do not affect the operation of other channels.

E. Receiving packets using the DMA channel receivers

Each channel can be associated to a different packet type or protocol using a packet filter based on the first four bytes of the header. Packets which are received on the same DMA channel are stored contiguously in memory and their packet length is stored in packet descriptors. Reception of RMAP packets is supported by providing the hardware computation of its CRC-8. Reception of PUS packets is supported by providing the hardware computation of its CRC-16. Continuous reception of packets is provided using circular buffer architecture with data and packet descriptor pointers. It is possible to enforce that a packet is not split at the end of the memory region. Interrupts can be set to monitor the progress of packets received without halting the actual operation. The user application or the SW driver should free the space used by packets already processed. This procedure allows data to be received at the maximum SpaceWire data rate with minimum CPU utilization. When an error occurs the reception is halted and the system is interrupted.

V. SpaceWire Router

The SpaceWire router has eight SpaceWire interfaces, three external port interfaces and an internal configuration port which supports the RMAP protocol. The internal configuration and status registers are also accessible through an APB interface. A control register is used to determine if the router is controlled through the configuration port or through the APB interface. Configuration by both masters at the same time is not supported although reading the status information from both masters at the same time is supported.

The SpaceWire router architecture is illustrated in Fig 3.

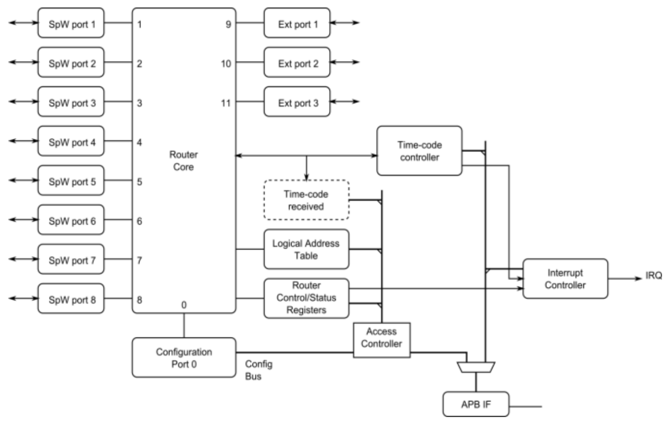


Fig. 3. SpaceWire router architecutre

VI. Time-code Controller

The SpaceWire time-code controller has functions to forward time-codes dependent on the time-code flags or to generate time-codes from software, processor timer interrupt or an internal dedicated time-code master count. The time-code controller has a time-code register for each of the four time-code flags, therefore allowing independent time-code forwarding for each flag code.

The time-code controller stores the last time-code received for each type of control flag and can indicate to the host that a time-code has been received through the status/interrupt interface.

The time-code forwarding mechanism checks that received time-codes are one more than the last time-code received then the time-code will be forwarded through all ports except the port the time-code arrived on. If the time-code is a distributed interrupt code then the interrupt vector is checked and the controller will forward the time-code if the interrupt vector bit is 0. If the interrupt vector bit is 1 the time-code is discarded as the interrupt has already been set. The time-code will be forwarded through all ports except the port the time-code was received on.

The controller can act as a time-code master either by software insertion of a time-code, sending time-code on a processor timer interrupt or by setting up an internal time-code master counter. The time-code frequency can be controlled by the host software with up to 1 micro-second precision.

Status bits and processor interrupts are provided for received time-codes for each time-code flag value, time-codes transmitted for each time-code flag value and distributed time-code interrupt occurred.

VII. Conclusion

The CASTOR chip is a capable SpaceWire processing unit which comprises a SPARC V8 process with an enhanced floating point unit and memory management unit running at 200 MHz on a radiation tolerant process. The SpaceWire engines inside the CASTOR chip provide high performance SpaceWire RMAP and DMA functions including dedicated RMAP target and initiator hardware to reduce the processor workload.

References

[1] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire, Links, Nodes, Routers and Networks”, Issue 1, European Cooperation for Space Data Standardization, July 2008.

[2] ECSS, “SpaceWire - Remote memory access protocol”, ECSS-E-ST-50-52C, Feb 2010.

[3] S. Parkes, Albert. Ferrer, S Mills, A Mason, “SpaceWire-D: Deterministic data delivery with SpaceWire”, International SpaceWire conference, Russia, 2010.

[4] P. Mendham, S. Parkes, “SpaceWire Plug-and-play: a Roadmap”, International SpaceWire conference, Nara, Japan, 2008.

[5] Yuriy Sheynin, Sergey Gorbatchev, Liudmila Onishchenko, “Real-Time Signalling in SpaceWire Networks”, International SpaceWire conference, Nara, Japan, 2008.

[6] Chris McClements, Steve Parkes, “SpaceWire RMAP IP Core”, International SpaceWire conference, Russia, 2010.

[7] S. Parkes, C. McClements, G. Kempf, S. Fishcher, P. Fabry, A. Leon, “SpaceWire Router ASIC”, International SpaceWire Conference, Dundee, 2007.

[8] ECSS standard ECSS-E-ST-50-51C, “SpaceWire engineering: SpaceWire protocol identification”, European Cooperation for Space Data Standardization, February 2010.

[9] ECSS standard ECSS-E-70-41A, “Ground systems and operations – Telemetry and telecommand packet utilization”, European Cooperation for Space Data Standardization, January 2003.

Next Generation Microprocessor Functional Prototype SpaceWire Router Validation Results

SpaceWire Components, Short Paper

Jonas Ekergarn, Jan Andersson, Andreas Larsson,
Daniel Hellström, Magnus Hjorth
Aeroflex Gaisler AB
Göteborg, Sweden

Roland Weigand
Microelectronics Section
European Space Agency
Noordwjik, The Netherlands

***Abstract*—The Next Generation Microprocessor is a quad-processor system-on-chip that contains a SpaceWire router with eight external SpaceWire links and four on-chip AMBA ports. This paper describes the validation work done for the SpaceWire router within the Next Generation Microprocessor functional prototype development.**

***Index Terms*— SpaceWire, Networking, Spacecraft Electronics.**

I. Introduction

The Next Generation MicroProcessor (NGMP) is a quad-processor system-on-chip currently being developed by Aeroflex Gaisler. The design includes four LEON4 SPARCV8+ processors with a shared Level-2 cache, DDR2-800 SDRAM main memory interface, a SpaceWire router with eight external SpaceWire links and four internal AMBA ports, two 10/100/1000 Mbit Ethernet MACs, 32-bit 66 MHz PCI interface and other interfaces.

The SpaceWire router allows the NGMP to act both passively and actively in a SpaceWire network. The target frequency for the NGMP device is 400 MHz. Preliminary results for this target frequency show that, using only internal routing, the architecture is able to sustain a data throughput of 1.5 Gb/s per SpaceWire AMBA port. In a scenario where the two full-duplex Ethernet links and all SpaceWire AMBA ports are run at full speed, the sustainable throughput is roughly 1.5 Gb/s for the Ethernet links and 1 Gb/s per SpaceWire AMBA port. In addition to this, the SpaceWire router will also be able to simultaneously route packets at maximum speed.

The implementation of NGMP in rad-hard technology was put on hold in April 2011, pending advances in the development of a suitable Deep-Sub-Micron technology for space. Development has instead progressed in the development of a NGMP functional prototype (NGFP) device targeting eASIC Nextreme2, a structured ASIC technology based on a 45 nm process. Silicon was received in August 2012 and an evaluation board has been manufactured.

One of the primary goals of the NGFP development is to allow use of the architecture at higher clock frequencies than what is attainable with FPGA prototype implementations. The prototype devices do not reach the full target frequency of the final device (400 MHz) but can be run at a system frequency of 200 MHz with the same clock frequency used for the SpaceWire router.

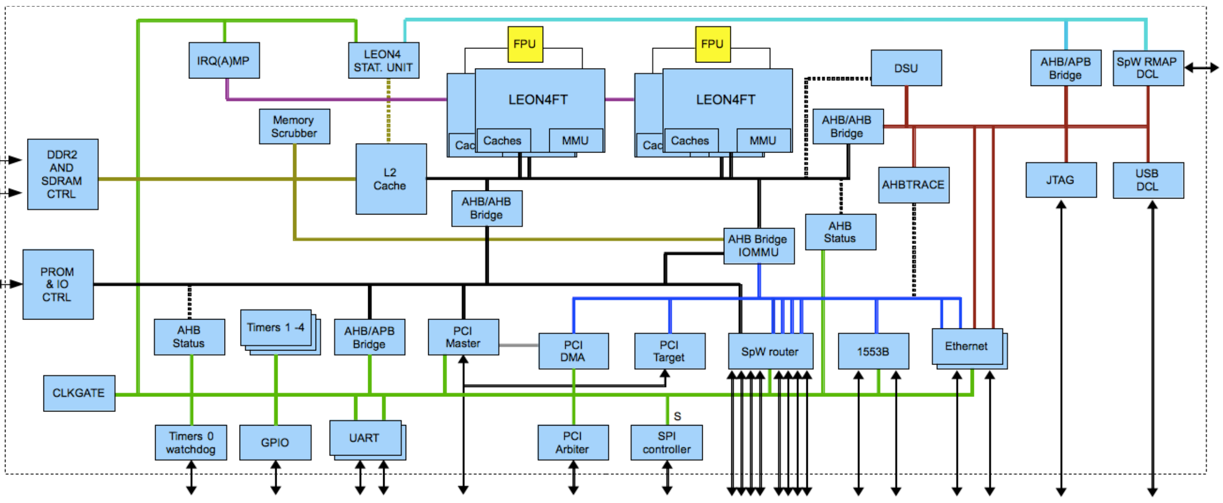


Fig. 1 Block diagram

II. Description of NGMP Functional Prototype Architecture

The system consists of five Advanced High-performance Buses (AHB); one 128-bit Processor bus, one 128-bit Memory bus, two 32-bit I/O buses and one 32-bit Debug bus. The Processor bus connects four LEON4 processor cores connected to a shared Level-2 (L2) cache. The Memory bus is located between the L2 cache and the main external memory interfaces, DDR2-600 SDRAM and PC100 SDRAM, and also connects a hardware memory scrubber. As an alternative to a large on-chip memory, part of the L2 cache can be turned into on-chip memory by cache-way disabling.

The two separate I/O buses connect all the peripheral cores. All memory-mapped interfaces of peripheral cores that can be directly accessed by the processors have been placed on one bus (Slave I/O bus), and all master/DMA interfaces have been placed on the other bus (Master I/O bus). The Master I/O bus connects to the Processor bus via an AHB bridge that provides access restriction and address translation (IOMMU) functionality. The two I/O buses include all peripheral units such as timer units, interrupt controller, UARTs, general purpose I/O port, PCI master/target, Ethernet MACs, MIL-STD-1553B, Serial Peripheral Interface bus and SpaceWire router. All I/O master units in the system contain dedicated DMA engines and are controlled by descriptors located in main memory that are set up by the processors. Reception of, as an example, Ethernet and SpaceWire packets will not increase the CPU load. The cores will buffer incoming packets and write them to main memory without processor intervention.

The fifth bus, a dedicated 32-bit Debug bus, connects a debug support unit (DSU), PCI and AHB trace buffers and several debug communication links. The Debug bus allows for non-intrusive debugging through the DSU and direct access to the complete system, as the bridge connecting the Debug bus to the Processor bus allows unrestricted access to the memory space.

The NGMP architecture has been designed to provide a significant performance increase compared to earlier generations of European space processors. The platform has improved support for profiling and debugging and will have a rich set of software immediately available due to backward compatibility with existing SPARC V8 software and LEON3 board support packages. The design also includes specific support for asymmetric multi-processing configurations. Five memory management units (MMUs), one per CPU core, and the IOMMU provide access protection. Several dedicated interrupt controllers allow interrupt steering to a specific CPU and duplicated timer units allow to run one operating system per CPU core with full space-partitioning.

III. SpaceWire Router IP Core and Configuration

The design includes Aeroflex Gaisler's GRSPWROUTER SpaceWire router IP core. The IP core implements a SpaceWire routing switch as defined in the ECSS-E-ST-50-12C standard. It provides an RMAP target for configuration port 0 used for accessing internal configuration and status registers. In addition to this, the implementation described by this paper implements two different port types; external SpaceWire links and on-chip AMBA interfaces.

One AMBA AHB slave interface is also provided for access in the port 0 registers from the on-chip AMBA bus. Group-adaptive routing and packet distribution are fully supported.

The GRSPWROUTER was implemented with the following characteristics:

- 64 entries per 9-bit receiver FIFO (N-Char FIFO)
- 32 entries per 32-bit AMBA port FIFO
- Four DMA channels per AMBA port
- Hardware RMAP target in each AMBA port

IV. SpaceWire Router Role In System-On-Chip Design

The system-on-chip architecture is a multi-processor architecture that provides a significant performance increase compared to earlier generations of European space processors, with high-speed interfaces such as SpaceWire and gigabit Ethernet on-chip.

The NGMP was initially specified to include for SpaceWire codecs with AHB host interfaces and hardware RMAP targets. The four SpaceWire codecs would use redundant ports giving a total of eight external SpaceWire links, where four of the links could be used separately.

The four SpaceWire codecs where later replaced by the SpaceWire router. The register interface of the AMBA ports of the SpaceWire router are software compatible with the register interface of the previously used codecs, giving little or no overhead for software implementations. While the SpaceWire router supports redundant ports the choice was made to implement the eight ports as separate links and instead to recommend group-adaptive routing as an alternative to the redundant port feature. This gives users the alternative to forego redundancy and instead use all eight available links simultaneously.

The NGMP is targeted at general payload processing with the main design goal of increasing the average processing performance. The main use of the SpaceWire router within this context is not to route SpaceWire traffic from external entities but instead to provide the same functionality as the previously included SpaceWire codecs.

The inclusion of the SpaceWire router provides more options to system designers. The device can be used to both provide SpaceWire connectivity to the on-chip processing components while also acting as a router for external entities. Protection mechanisms in the architecture also allow the use of the SpaceWire router to be completely separate from the rest of the design. In effect packaging the router and microprocessor components together with the gain of reducing the number of required devices.

V. Traffic Routing

The first use of the SpaceWire router within the NGFP validation effort was to study the effects of routing AMBA traffic either through or behind the system's Level-2 cache.

The test consists of an RTEMS application that is transferring data over the four AMBA ports simultaneously.

The router is configured to route the SpaceWire packets from AMBA port 0 to AMBA port 1, then AMBA port 1 to port 2 and so on back to AMBA port 0. This means that every packet will exercise eight DMA operation channels every time one packet makes one round-trip. The number of round-trips is counted and performance figures are calculated based on these counts.

The packets are marked with a unique sequence number and contain 16-bit incremented data. This is done to be able to verify packet receive/transmit ordering and data correctness. The packet sequence is verified for every received packet. The data is verified after all transmissions are finished.

The tests were performed with the internal SpaceWire fabric and AMBA system running at 200 MHz.

The test was run in several different configurations, of which three are considered here:

- CFG2 – Cache-coherent system with Level-2 cache, caching all traffic
- CFG5 – Cache-coherent system with Level-2 cache, SpaceWire DMA buffers and traffic not cached
- CFG10 – System with Level-2 cache. SpaceWire DMA buffers not cached by Level-2 cache. SpaceWire DMA traffic does not pass through Level-2 cache. In this configuration the cache coherency of the L1 cache cannot be maintained through bus snooping. The processor MMU is used to mark the DMA buffers as noncachable to solve the coherency issue.

The results of the tests showed that the highest performing configuration is CFG2 where the Level-2 cache caches all DMA traffic. This is expected as the software execution causes little interference and the Level-2 cache is essentially dedicated for DMA buffers. In a configuration where additional software instances made use of the Level-2 cache it is expected that the SpaceWire traffic throughput and software application performance would be negatively affected due to the shared resource in the Level-2 cache. The combined throughput for all DMA ports was measured to 1.54 Gbit/s.

The test case CFG5 showed that SpaceWire throughput is more than halved when marking the DMA buffers as uncached in the Level-2 cache. In this configuration the Level-2 cache does not add any benefit when fetching data from external memory, instead the cache only adds latency on each DMA access.

The CFG10 test case showed the effects of bypassing the Level-2 cache and routing traffic directly to the main memory controller and was completed using a FPGA prototype due to NGFP IOMMU silicon errata. The test showed that the throughput decreases with 18% in CFG10.

While the data throughput for this particular test is lower when bypassing the Level-2 cache it is important to recognize the effects on the processor system. When bypassing the Level-2 cache the DMA traffic will have negligible, if any, impact on software instances with high Level-1 and Level-2 cache hit rates. This allows large amount of data to be transferred to main memory without processor intervention and without impacting performance of software.

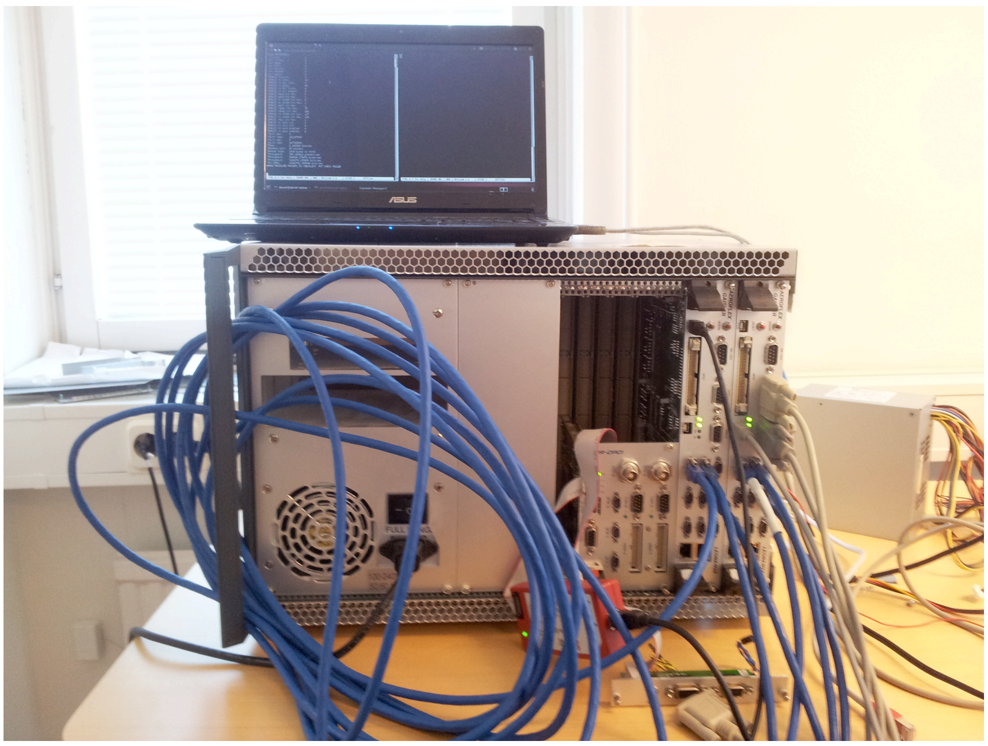


Fig. 2 Example of test rig setup

VI. SpaceWire Routing Tests

The traffic routing test described in the previous section studied the effects of using the AMBA ports to transfer large amounts of data. The second set of validation tests performed on the functional prototype device that involved the SpaceWire router focused on the routing capabilities of the router. The tests were divided into four major groups:

- All SpaceWire ports – Exercise the router by generating traffic on all ports
- Group-adaptive routing
- Packet distribution
- Priority routing
- Packet timeout

All tests described below were performed with the internal SpaceWire fabric running at 200 MHz and the AMBA system running att 200 MHz. All SpaceWire links were configured to operate at a bitrate of 200 Mbit/s.

A. All SpaceWire ports

To validate that all the SpaceWire ports of the SpaceWire router can handle both receive and transmit at a rate of 200 Mbit/s, each SpaceWire port was connected to another SpaceWire port. 4 MiB packets were then sent from an AMBA port, routed out onto a SpaceWire port, received at another SpaceWire port, and then routed to an AMBA port were the data was validated. This test was repeated so that all SpaceWire ports were utilized, and both path addresses and logical addresses were used for the packets.

B. Group adaptive routing

The SpaceWire router supports group adaptive routing for all path addresses and logical addresses. Group adaptive routing means that packets can be routed through the network over different paths depending on which of the router's ports that are available when the packet arrives. For example, a packet with address 0x40 arrives at SpaceWire port 1 of the router, and address 0x40 is configured with group adaptive routing to SpaceWire port 2 and 3. The router will then route the packet to either port 2 or port 3 depending on which port becomes available first. If both ports are available, the router will send the packet on the port with the lowest port number. The group adaptive routing mechanism was validated by connecting four SpaceWire ports together and then sending packets from an AMBA port where the address byte of the packets were configured with group adaptive routing to two of the four ports. When the packets arrived at the router again they were routed to another AMBA port. It was then verified that the packets arrived correctly as long as one of the two SpaceWire used as output ports were connected to another port. If none of the two SpaceWire ports used as output ports were connected then the packet was not received at the AMBA port used as destination. Group adaptive routing as also verified further in the packet distribution validation (see below).

C. Packet distribution

Packet distribution - which means that data arriving at a input port is sent to multiple ports simultaneously - is supported by the SpaceWire router for both path addresses and logical addresses. This feature was validated by connecting four SpaceWire ports to each other and then sending a packet with two address bytes from an AMBA port. The first address byte was configured with header deletion and packet distribution out on the four SpaceWire ports, and the second address byte was configured with group adaptive routing to AMBA ports 0-3. When the packet was sent from the AMBA source port the first address byte was removed by the use of header deletion, and the packet was routed out onto the four SpaceWire ports. It was then verified that the four packets, arriving at one SpaceWire port each, was routed to one AMBA port each (because group adaptive routing was used for the second address byte). This test also adds additional validation of group adaptive routing since the test validates that group adaptive routing works when the destination ports are busy with transmitting data. The validation of group adaptive routing described above only validated the case when the destination links were not running.

D. Priority routing

When packets are to be routed, each destination port is arbitrated individually using a two level priotiy. The priority is based on the first address byte of the incoming packet, and all path addresses and logical addresses can be assigned either a high or low priority. Round-robin is used when one or more packets with the same priority competes about the same destination port. The validation of the priority routing mechanism was done by enqueueing four different packets, each one from a different AMBA port, where all packets were to be routed out on the same SpaceWire port. Three of the packets contained an address that had been assigned a low priority, while the fourth packet contained an address with high priority. The SpaceWire port that the packets would be routed out onto was connected to another SpaceWire port of the router, and the second address byte in all packets was the path address of one of the AMBA ports (same for all packets so that the order could be observed). The three low priority packets were sent slightly before the high priority packet, and it was then validated at the destination AMBA port that the first packet received was the first low priority packet, followed by the high priority packet, and then followed by the two remaining low priority packets. It was also validated that if the high priority packet was instead changed to low priority it was received last of the four packets.

E. Packet timeout

The SpaceWire router implements packet timers in order to prevent situations where the ports becomes blocked for ever if, for example, a source stops sending data without terminating the packet with an end of packet marker (EOP and EEP). In such a situation the router will detect that no data has been sent for a certain amount of time (configurable), and the packet will then be spilled and the destination port released. Connecting two of the router's SpaceWire ports to each other has validated the packet timeout feature. Then a packet containing two address bytes was sent from an AMBA port. The first address byte made the router route the packet out onto the first of the two mentioned SpaceWire ports. The second address byte made the router try to route the packet to a SpaceWire port that was not connected to anything. After the first packet was sent another packet was sent from a second AMBA port. The first address bytes of the second packet informed the router to route the packet out onto the same SpaceWire port as the first packet, and the second address byte was the address of a third AMBA port. In the case that the SpaceWire routers timers were not enabled it was verified that the second packet never reached its destination (because the first packet blocks the outgoing SpaceWire port for ever). If timers were enabled it was verified that the second packet eventually reached its destination, since the first packet was spilled by the router after a timeout period when it failed to route it.

VII. Conclusion

The parts of the NGFP validation effort that included the SpaceWire router aimed to prove the design decision to allow AMBA traffic to be routed so that it bypasses the Level-2 cache and to demonstrate core functionality of the router.

The functionality to bypass the Level-2 cache was successfully demonstrated using the SpaceWire router AMBA ports and as a side effect also verified high-speed communication between the router's AMBA ports.

Core functionality of the router was also demonstrated by generating traffic on all ports and execution of test cases using group-adaptive routing, packet distribution, priority routing and packet timeouts.

The NGMP is part of the ESA roadmap for standard microprocessor components and it will be commercialized under fair and equal conditions to all users in the ESA member states. The NGMP is fully developed with manpower located in Europe, and it only relies on European IP sources. It will therefore not be affected by US export regulations.

Acknowledgment

The Next Generation Microprocessor is developed in activities commissioned and funded by the European Space Agency under contracts 22279/09/NL/JK and 18533/NL/JD.

Galvanic Isolation of SpaceWire Links

Components, Short Paper

G. Baterina, Y. Moghe, P. Francois
Silanna Group Pty Ltd
37 Brandl Street, Eight Mile Plains, QLD 4113, Australia
gil.baterina@silanna.com , Tel: +612 9763 4111

A. Senior
Systems Engineering & Assessment Ltd
Bristol, United Kingdom
alan.senior@sea.co.uk

***Abstract*—This paper summarizes the need for galvanic isolation in SpaceWire networks, reviews the limitations of current isolation solutions, and proposes a new SpaceWire Link Isolator device based on radiation-hardened Silicon-on-Sapphire (SOS) technology. As a stepping-stone to the proposed device, a discrete galvanic isolation solution was demonstrated using Silanna's core isolator chip along with commercial off-the-shelf (COTS) LVDS transceivers and isolated DC-DC converters to power all circuitry across the isolation barrier. SpaceWire links operating at 200 Mbps were successfully isolated and handled the introduction of up to 50 V of common-mode voltage on the demonstration unit. The proposed integrated solution is expected to have an on-chip isolated power and operate up to 400 Mbps, handling a common-mode of 100 V-RMS, and a galvanic isolation of 1 kV-RMS.**

***Index Terms*— Spacewire, SpW, isolation, fault propagation, LVDS, common mode voltage, galvanic, component**

I. Introduction

SpaceWire (SpW) as defined in the standard [1] uses the Low Voltage Differential Signaling (LVDS) electrical interface which has the advantage of reducing the power required for a high speed data link, however the existing LVDS buffer and ASIC devices have 2 principal drawbacks for implementing high reliability systems:

1. limited common mode voltage tolerance
2. fault propagation paths

The common mode tolerance is +/-1V; if this voltage is exceeded then the link data may be corrupted. In the worst case the transmitter/receiver devices may either be stressed or permanently damaged. Stressing of the LVDS buffer may not be evident but often results in a reduced reliability leading to premature failure later. Within a spacecraft, it is practical to control the common mode voltages within the specified limits and thus once launched problems would not be anticipated. Control of the common mode voltages during ground testing of spacecraft with remote Electrical Ground Support Equipment (EGSE) that use long cables becomes more problematic; drivers and receivers have failed in test configurations either due to incorrect test setups, poor grounding setups, or the effects of EMC testing. Clearly it is important to implement an effective grounding scheme and ensure that methods for monitoring the common mode voltages are in place rather than wait for failures to occur or assume acceptable conditions are met.

Fault propagation paths exist between LVDS link ends due to the direct silicon to silicon connection between the devices at the two ends of a link [2]. A power supply failure in one piece of equipment could propagate to another equipment by injecting out of specification voltages at the LVDS buffer terminals [3]. Due to the constraints of high speed signaling, it is not practical to use series protection resistors in the signal lines to reduce potential fault currents to an acceptable level; thus, it is necessary to add protection to the internal supply rails of each equipment.

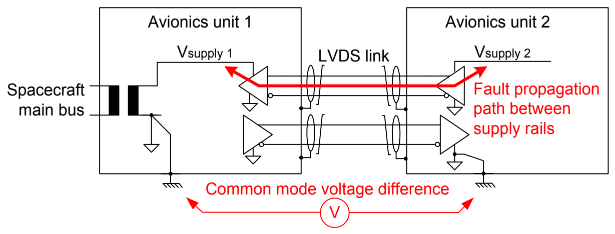


Fig. 1 Common mode voltages and fault propagation

The mitigation methods for both the common mode and fault propagation issues are time consuming to analyse for failure mode effects and they typically result in an increased complexity of the flight equipments.

It is thus highly desirable to incorporate galvanic isolation in the link paths, this will permit the legacy Mil-Std-1553B command and control links to be replaced with the more capable SpW bus and to eliminate failures in test environments with EGSE.

II. Limitations of Current Isolation Solutions

The Data and Strobe lines of SpW are non-DC-balanced signal streams with data rates up to 400 Mbps. Since the signal streams are not DC-balanced, typical capacitive or inductive (transformer) AC coupling methods for isolation are not viable; in comparison, by design, high speed digital isolators are capable of handling non-DC-balanced signal streams. However, the high speed digital isolators available today have maximum data rates of 150 Mbps; this falls below SpaceWire's

400 Mbps maximum data rate. Isolators based on opto-couplers are relatively slow (~10 Mbps), susceptible to radiation, and degrade in performance over time. Silanna has already demonstrated the digital isolation of signal streams greater than 500 Mbps using a 0.5µm Silicon-on-Sapphire (SOS) process. [4]

III. Proposed SpaceWire Link Isolator

The proposed SpW link isolator device would have four high-speed data channels (two in each direction) to handle Data and Strobe transmit & receive signaling; with LVDS levels on the cable-side & selectable LVDS/LVTTL levels on the module-side of the isolation barrier, both discrete and integrated SpW links could be isolated with a nearly drop-in isolation solution. To further simplify the adoption of the SpW isolator, the device would also include the integration of a DC-DC isolator to optionally provide power to the cable-side from the device side without the need for additional active components (see Fig. 2).

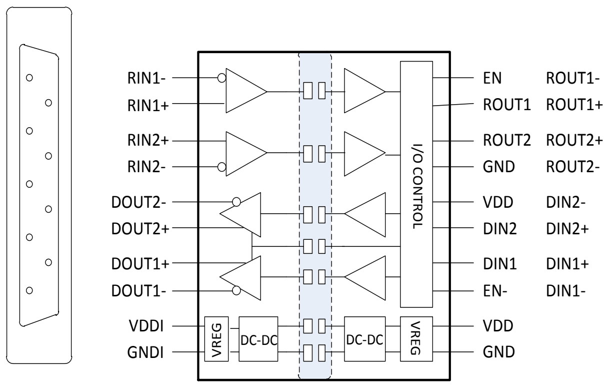


Fig. 2. Silanna SpW Link Isolator

A summary of the target features are:

- 4 high speed (400 Mbps) channels
- Cable-side: LVDS
- Module-side: LVTTL or LVDS
  - LVTTL: LV049 Mode
  - LVDS: Repeater Mode
- LVDS failsafe per SpW standard
- Cold sparing for redundant backup
- Isolation voltage: 1 kVrms
- Working voltage: 100 V (common mode voltage)
- Integrated DC-to-DC isolator to power cable-side from module-side
- Cable-side data lines align well w/ SpW cable connection
- Ground-based device in 20-pin plastic package
- Space grade device in 20-pin ceramic package
- Silicon-on-Sapphire (SOS) technology
- Target Radiation Tolerance > 100 krad(Si) TID (for Space grade)

IV. Demonstration Module

To demonstrate the high speed digital isolation capabilities in a SpW application, a demonstration (demo) module was built around the Silanna SIL1042L 4-channel isolator device (Fig. 3). A pair of LV049 (dual channel LVDS transceiver) type devices was used for the LVDS I/O. An isolated DC-DC converter was also included as an option to power the isolated side of the module. The module demonstrated the wide common mode range of the isolated interface and the capability to handle the non-DC-balanced SpW data streams up to 400 Mbps.

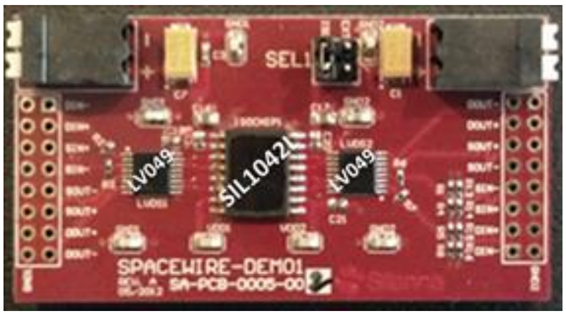
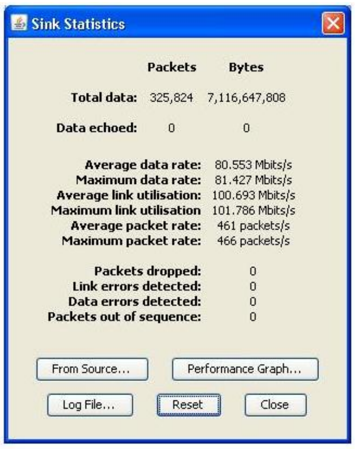


Fig. 3. Silanna SpW Demonstration Module

V. Initial Testing

The demonstration module was successfully tested with non-DC-balanced pseudo-random bit streams at data rates up to 400 Mbps; common mode voltages of up to 10 Volts were also introduced without disruption.

Testing in SpW environments is currently in progress with promising results, the links operating at DC and AC common mode voltages up to 50V.



```
                    Packets      Bytes
          Total data:  325,824  7,116,647,808

        Data echoed:        0              0

          Average data rate:  80.553 Mbits/s
          Maximum data rate:  81.427 Mbits/s
   Average link utilisation:  100.693 Mbits/s
   Maximum link utilisation   101.786 Mbits/s
        Average packet rate:  461 packets/s
        Maximum packet rate:  466 packets/s

            Packets dropped:  0
       Link errors detected:  0
       Data errors detected:  0
   Packets out of sequence:   0
```

Fig. 4. Sample Statistics of Isolated SpW Link Transfer

VI. Conclusions

The initial testing of the SpW demo module clearly addressed one of the principal drawbacks for implementing high reliability SpaceWire systems – limited common mode voltage tolerance. The unit was able to pass 400 Mbps data with up to 50 Volts of common mode voltage present; the proposed integrated SpW link isolator is targeted to have 100 Volts of common mode voltage tolerance. Although galvanic isolation should readily eliminate the problem of fault propagation paths, testing continues at Silanna and within the SpaceWire community to confirm that this drawback is also addressed.

References

[1] European Space Agency - ECSS Secretariat, “ECSS-E-ST-50-12C, Space engineering, SpaceWire – Links, nodes, routers and networks,” 31st of July 2008, 129 pages

[2] M. Suess, J. Ilstad, W. Gasti, “Galvanic Isolated SpaceWire Links, Requirements, Design Options and Limitations,” 2009 ESA Workshop on Reliable Power & Signal Interfaces

[3] R. Malmberg, “Failure Propagations via Power and Signal Interfaces,” 2009 ESA Workshop on Reliable Power & Signal Interfaces

[4] Y. Moghe, A. Terry and D. Luzon, “Monolithic 2.5kV RMS, 1.8V - 3.3V Dual-Channel 640Mbps Digital Isolator in 0.5µm SOS,” SOI Conference (SOI), 2012 IEEE International, On page(s): 1 - 2

Low Mass SpaceWire and Copper based SpaceFibre Links

Components, Short Paper

Gilles Rouchaud
Axo.com Space Products Division
Axon' Cable SAS
Montmirail, France
g.rouchaud@axon-cable.com

Nigel Kellett
UK Division
Axon' Cable Ltd
Rosyth, Scotland
n.kellett@axon-cable.co.uk

Abstract— **This paper reports on the major achievements of the ESA project to develop a Low Mass SpaceWire cable design, the main content of which was previously presented at the last International SpaceWire conference in San Antonio, November 2011.**

Today, the team has completed the project with a cable mass reduction by at least 50 per cent, and is able to report on the final selections of the different cable versions presented in 2011, along with their relative mechanical, electrical and shielding performances versus mass savings.

Two versions remain from the 4 initial versions after final selection. The first version uses twisted shielded pairs and presents electromagnetic, mechanical and electrical performances similar to the existing standard, and is therefore recognized by ESA. The second version uses pairs of coaxial cables and presents higher attenuation, thus limiting the usage length, but significantly increased flexibility and an even lower mass. This version, however, is not ESA recognized.

The project team additionally concludes on the maximum advisable lengths for the different low mass versions assessed.

Therefore, taking all of this into account, the team is currently working with ESA and Star Dundee to feed all of these conclusions into the latest draft of the ECSS-ST-50-12 assembly standard and the ESCC 3902/003 cable standard updates. The screen termination method has been reviewed in depth. The inner and overall shields are terminated together to the connector shell at both ends.

SpaceWire, however, remains limited to 400Mbits/s or its maximum usable length. The new SpaceFibre standard will be a multi-gigabit protocol proposing copper or fiber optic solutions for very high data rates and/or longer cable lengths. A good connector candidate for a copper cable solution is named Axomach®, an Axon' connector previously developed with CNES for transmissions up to 10Gb/s per channel. The latest tests run by Star-Dundee demonstrate that this cable and connector solution works very well for crossover transmission lines running at 2.5Gb/s.

Index Terms— **Relevant indexing terms: SpaceWire, SpaceFibre, Micro-D, Nano-D.** *(key words)*

I. Background, Existing Limitations

This short paper intends to provide a brief overview of the recently completed ESA project to develop a low mass alternative to the existing SpaceWire cable, and will discuss continued limitations and possible future developments in this area.

An overview of the progress (at that time) has previously been presented at the last International SpaceWire Conference in San Antonio, Texas in November 2011. This paper also serves as an update to that presentation.

The existing SpaceWire cables, recognized by the ECSS-ST-50-12 standard, have a number of limitations. They are relatively heavy, at approximately 80g/m for the lightest (AWG28) version, they are fairly rigid, they are not particularly radiation tolerant, and they have quite a large minimum bend radius, particularly the bigger AWG26 version. All of these limitations reduce the suitability of the current SpaceWire cable for installation and use in spacecraft, although they remain currently the only approved options.

In addition to these physical constraints with the cable itself, the standardized interface connector, the 9 way Micro-D, is not impedance-matched and therefore not optimized for the application, and the standardized wiring schedule is not optimized for EMC.

The ESA ITT sought to address many of these issues, the principle one of which was weight reduction, and Axon' Cable, the winner of the tender, has therefore been developing such optimized solutions along with their consortium partners, Star Dundee in Scotland and EADS Astrium in Toulouse, France.

II. Ways to Reduce Mass, Two Potential Variants Selected

Mindful of the electrical performance requirements dictated by the standard, it was necessary still to use certain minimum dimensions and materials. However, three main areas were focused on to bring about improvements:

- The use of lighter materials. Essentially broken down into conductors and insulators, the use of lighter conductors such as aluminium, was proposed where appropriate, as opposed to copper; and for the insulators or dielectrics, the use of expanded or alveolar materials was selected, as being lighter than their solid counterparts.

- Constructional changes. The existing SpaceWire cable is a construction consisting of four individually screened and jacketed twisted pairs all laid together in an assembled bundle, which itself then has an overall braided screen and an overall extruded outer jacket. The project members sought to explore other constructions which could achieve the same or similar electrical performances while using a "lighter" combination of elements.
- Flexibility and radiation. While considering weight reduction, the project team also gave consideration to possible ways of increasing the flexibility and radiation tolerance of the finished cable.

In all, some 12 suggested constructions were put forward for initial analysis and testing, following which two constructions of interest were finally selected.

The first of these, known initially as Variant 03, and then given the project designation C-OA-TPA-A-2819, is similar to the existing SpaceWire AWG28 cable, with the following key differences:

- The silver plated copper conductor is a 19 stranded AWG28 conductor, giving it more inherent flexibility than the existing 7 stranded version.
- The primary insulation is of an aleveolar PTFE (or aPTFE) construction, where the material is not solid but has a number of air gaps in a lattice-like structure, see Fig 2. This has the twin advantages of improving the dielectric constant whilst also reducing weight,

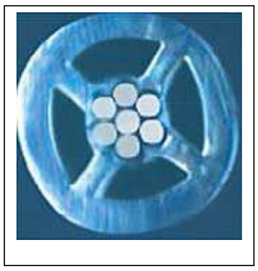


Fig. 1. Cross sectional view of an alveolar PTFE insulated conductor.

- The inner screens of each twisted pair are silver plated aluminium instead of silver plated copper, thus substantially reducing weight (and not requiring termination)
- The filler between the four screened pairs is of expanded PTFE, for which Axon's trade name is CELLOFLON®, a material which is inherently very flexible,
- An outer screen has been retained, also in silver plated aluminium, but which, very importantly, is in contact with the four inner screens,
- And finally an outer insulation is constructed by the use of a CELLOFLON® expanded PTFE inner tape and a polyimide (KAPTON) outer tape, providing an excellent combination of lightweight flexibility and good radiation tolerance.

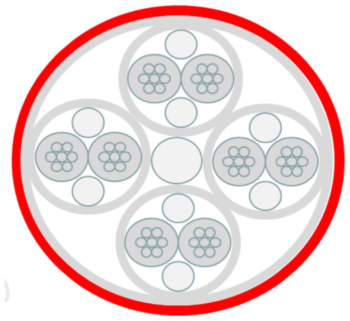


Fig. 2. Variant 03 (C-OA-TPA-A-2819) Low Mass SpW cable (Outer shield in contact with inner pair shields, all shields in silver plated aluminium)

Having been proven to have acceptable performance results in all tests, this variant has been accepted in principle by ESA as a potential lightweight "drop in" replacement to the existing SpaceWire variants, and is in the process of being added to the latest revision of the standard.

The second construction of interest, known as Variant 09, (C-OC-CPC-P-3407) takes a completely different constructional approach, based on four pairs of coaxial cables. Although these are not 100 ohm impedance pairs each coaxial has a 50 ohm impedance, and therefore under the required SpaceWire tests as defined by the Project scope they nevertheless perform satisfactorily. An enhanced version of this variant employs an overall shield for improved EMC underneath an outer insulation of similar construction to that of Variant 03. The mechanical advantages of this variant are substantial;

- Mass of around 33 g/m – almost 70% weight saving,
- Outer diameter of around 4.5 mm as opposed to over 7 mm for existing SpaceWire,
- Extremely small bend radius – the cable can almost be bent double during installation and still perform satisfactorily, see Fig 3.

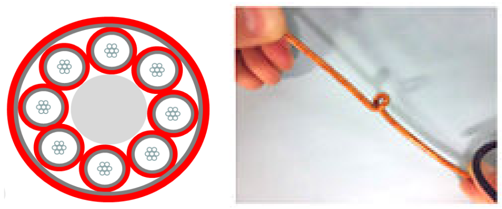


Fig. 3. Alternative variant based on 8 x coaxial cables, with picture showing the tight bend radius possible whilst maintaining electrical performance

However, it is important to note that according to ESA, the coaxial construction is not theoretically suitable for a floating load LVDS application such as SpaceWire, and as such ESA does not currently endorse its use.

This cable can be terminated using existing 9 way Micro-D connectors, or indeed the much smaller 15 way Nano-D connectors, thereby saving even more space and weight (but clearly that would then require devices with Nano-D mating halves). It is not suitable for re-work, however, being based on AWG34 wires, significantly smaller than the minimum acceptable gauge size recognized by ESA, AWG30.

III. Comparison Between Both Versions

Both cable types meet the SpaceWire performance criteria, although because of the smaller gauge size it would not be appropriate to use the coaxial version in longer lengths. If we assume that an "acceptable" value for Insertion Loss over the whole assembly is 6dB, then we can calculate maximum usable lengths for each type, (not including connectors) and summarize as follows:

TABLE I. Insertion Loss comparison

Part number/ Bach N°		P551259A / X19623	P551260A^ / X19371
Comments		*43g/m, twisted pairs, ESA endorsed*	*33g/m, coaxial pairs, not ESA endorsed*
Code		*C-OA-TPA-A-2819*	*C-NO-CPC-P-3407*
Performance at 250MHz	*S21 (dB/m) @ 250MHz*	0.6	1.24
Data rate 100Mb/s	*Max length to reach 6dB in m*	**10**	**4.8**
Performance at 500MHz	*S21 (dB/m) @ 500MHz*	0.85	1.8
Data rate 200Mb/s	*Max length to reach 6dB in m*	**7**	**3.4**
Performance at 1000MHz	*S21 (dB/m) @ 1000MHz*	1.27	2.57
Data rate 400Mb/s	*Max length to reach 6dB in m*	**3.7**	**2.2**

IV. SpaceWire Screen Terminations and Connector Choice

The wiring and screen bonding schedule for the current SpaceWire cable is not optimized, but rather was adopted at the time, partially due to the constraints imposed by a combination of the construction of the cable (where all the inner screens are isolated from the outer screen) and the 9 way Micro-D connector (which does not have enough contacts to terminate all 8 wires and all 4 inner screens). We therefore have the existing standard wherein two of the inner screens are short circuited together and terminated to pin 3 at one end, and the other two inner screens are similarly terminated to pin 3 at the other end, meaning that no inner screen is continuously connected from one end to the other. This was considered acceptable under previous EMI guidelines, but it is generally accepted today that shield bonding at both ends is far better. The outer screen is terminated to the shell of the connector (or backshell) but there is no specific requirement for a 360° shield termination.

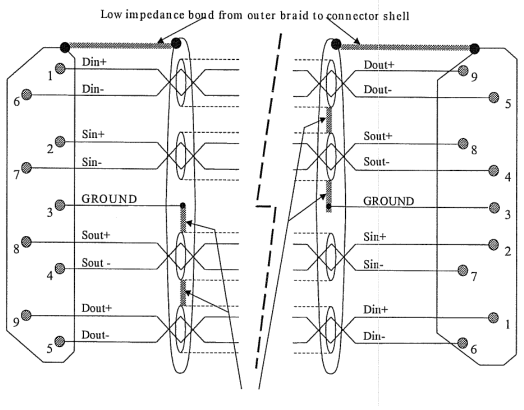


Fig. 4. Existing SpaceWire wiring schedule

Assuming, for backward compatibility reasons, we wish to retain the 9 way Micro-D connector for some time to come, we can now substantially improve this wiring schedule with the new (ESA endorsed twisted pair) cable construction. Here, because all the inner screens are now directly in contact with the outer screen, we can remove any screen termination to pin 3, thus avoiding the transfer of any EMI interference directly into the electronics. We can then simply terminate the outer screen to the body of the connector or backshell, thus effectively terminating all screens together in one go. For EMC purposes, it is highly recommended to employ a backshell at the rear of the Micro-D connector, with a cable entry funnel optimized to be only slightly larger than the inner bundle of four pairs, and then to terminate the overall shield over this funnel with some recognized form of 360° screen termination, such as a EMC band clamp. This simplified, but improved, wiring schedule will now resemble Fig. 5.

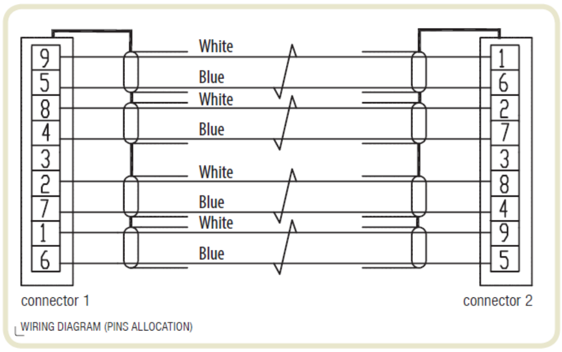


Fig. 5. Proposed Low Mass SpaceWire Wiring Schedule

The Micro-D connector family, however, is not designed to be EMC optimized. There are no guaranteed 360° points of contact between the male and female bodies, so much of the screen connection tends to travel through the jackscrew and jackpost fasteners, known in the Micro-D industry as the "hardware". This can be improved, however, by the addition of an EMC gasket on the mating face of the male flange, which if applied on both ends, will result in significantly better EMC performance, as shown by the Zt (Transfer Impedance) figures in Table II.

TABLE II. TRANSFER IMPEDANCE OF A SPACEWIRE LINK WITH AND WITHOUT EMC GASKETS

Frequency in MHz	**Transfer Impedance (Zt) in mΩ**		
Version	***Without gasket***	***With gasket on one end***	***With gasket on both ends***
DC	31.3	31.3	31.3
0.03	29.7	31.6	28.1
0.10	30.5	31.6	28.45
1	38.2	39.6	34.3
10	120	107	60.2
20	182	150	75.9
50	341	329	108
100	646	453	146
250	1642	594	378
500	2929	2240	1505
1000	28619	25855	1075

As part of the Project scope, a survey was undertaken to identify possible connectors better suited to increasing data rates than the existing, non-impedance matched, 9 way Micro-D (which nevertheless remains adequate for SpaceWire transmission, even at 400Mb/s). A number of potential connector candidates were identified with varying degrees of performance, including both circular and rectangular varieties. These included connectors from Sabritec (US), Airborn (US), Molex (US) and Axon' (F). In all some 10 different connector families were reviewed on paper. All have limitations relating either to performance, ease and suitability for SpaceWire termination, size, or restricted availability.

In summary, there is no optimized connector (for both performance and size) available from a European source, indeed even although only one in particular from the US does appear promising, it is patent pending, and will therefore likely be restricted to a single source.

There is scope, therefore, for a European manufacturer to come up with a suitably optimized connector for existing SpaceWire and Low Mass SpaceWire, which should be impedance matched to 100 ohms, EMC optimized, and with body dimensions that may allow for "forward compatibility" for future higher data rate applications, such as SpaceFibre.

V. SPACEFIBRE – COPPER CABLE SOLUTION FOR UP TO 10GB/S

Out with the Low Mass SpaceWire project, work is ongoing to develop the multi-gigabit SpaceFibre protocol, with potential media solutions in both copper and optical cable. Axon' has already developed a space grade copper based solution in association with the CNES, which (cable and connectors combined) is capable of operating at up to 10Gb/s per 2 way channel. A four channel version of this exists permitting total data transfer rates across the link of up to 40Gb/s. This solution is based on pairs of high frequency coaxial cables.

In order to comply with the intended SpaceFibre requirements, a two way version of such an assembly, trade name, AxoMach®, would be required in a crossover configuration, permitting full duplex operation servicing transmitter and receiver at both ends. Such an assembly has been tested by Star Dundee at 2.5Gb/s with satisfactory results, see Fig. 6.

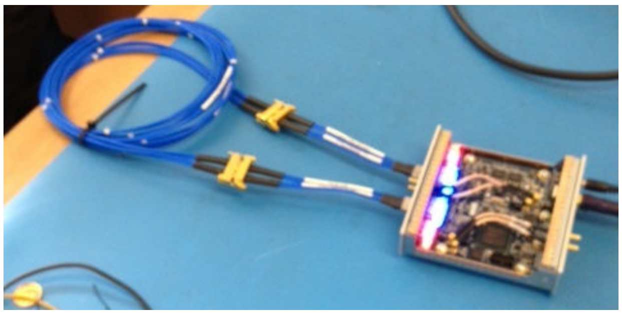


Fig. 6. AxoMach crossover 2 way link connected to a Star Dundee SpaceFibre test unit at 2.5Gb/s (with eSATA adaptors)

VI. CONCLUSIONS

A low mass alternative to the current SpaceWire cable now exists at approximately 50% of the standard weight, and can be used as a drop-in replacement with existing 9 way Micro-D connectors for compatibility with most systems. This cable is currently being added to the updated SpaceWire standard.

An ultra-low mass version, (70% weight saving) based on coaxial cable pairs also exists, but despite satisfactory test performance, is not ESA-endorsed for SpaceWire. User testing would be required to determine if this solution could really be used in actual LVDS applications.

The most common SpaceWire interface connector, the 9 way Micro-D, is neither impedance-matched nor EMC optimized, and there is no ready European solution for an improved SpaceWire connector. There is therefore scope to develop a matched impedance connector standard for SpaceWire, particularly if it can also provide forward compatibility for SpaceFibre operation.

A fully compatible space grade copper-based cable and connector solution already exists for SpaceFibre transmission, and indeed can support much higher data rates of up to 10Gb/s per channel.

ACKNOWLEDGMENT

Axon' would like to acknowledge, with thanks, the collaboration and support throughout the duration of the Low Mass SpaceWire project of J. Ilstad (ESA), S. Parkes (Star Dundee), C. Carron (EADS Astrium Toulouse) and their respective teams.

A Modular Connector for SpaceWire Backplanes

SpaceWire Components, Short Paper

Keir Boxshall
Smiths Connectors
London, U.K.
keir.boxshall@smithsconnectors.com

Sanjay Sharma
Systems Engineering and Assessment Ltd
Bristol, U.K.
sanjay.sharma@sea.co.uk

Alan Senior
Systems Engineering and Assessment Ltd
Bristol, U.K.
alan.senior@sea.co.uk

***Abstract*— SpaceWire can be considered a de-facto standard for onboard payload data systems to implement data links up to 400 Mbps. SpaceWire has been adopted by all the major space agencies throughout the world and most future missions will use it. ESA, NASA, JAXA and ROSCOSMOS, for example, are all specifying missions with a requirement to use SpaceWire standard for the interface links. The standards are maintained and issued formally as ECSS documents (e.g. ECSS-E-ST-50-12C) and this means that equipments designed by different agencies are interoperable, this is a significant advantage.**

Presently SpaceWire is mainly used between instrument units, however in order to facilitate a higher level of integration of onboard systems based on SpaceWire there is a need for a suitable connector for backplanes which offers a high number of power and discrete signal pins as well as impedance matched connectivity for high speed serial links.

An extensive market survey of available connector types has revealed a lack of space qualified connectors that offer the high I/O density required as well as matched 100 Ω impedance paths for SpaceWire links. Looking at the market perspective, such a connector would be useful for applications not only for SpaceWire but also for the next generation of high speed serial links such as SpaceFibre Copper that will have a minimum data rate of 2.7 Gbps.

This paper presents a new modular connector that Smiths Connectors, has designed and prototyped specifically for high integrity aerospace backplane applications, it incorporates a configurable set of power and discrete pins as well as controlled impedance pins that can operate at up to 10 Gbps.

***Index Terms*—SpaceWire, SpaceFiber, connector, backplane, high speed, controlled impedance.** *(key words)*

I. Introduction

The requirement for a robust, space qualified backplane capable of supporting 400 Mbps, 100 Ω impedance matched SpaceWire (SpW) links and capable of transmitting power and high density discrete signals was established. An extensive market study of pre-existing backplane connectors was performed yielding no suitable candidates for the application.

Initially three configurations were defined containing differing numbers of SpaceWire links, power and discrete contacts. The three configurations were estimated to cover a wide variety of, as yet undefined, applications for the connector standard within the SpaceWire community.

Smiths Connectors proposed a highly configurable connector system that satisfied all three defined configurations and numerous additional configurations by virtue of a connector with a singular, columnar modularity in terms of its contact architecture. The modular connector proposed will become a standard connector range for Smiths Connectors which would be intended to be sold into numerous other markets requiring robust backplane connectors.

The aim of this paper is to detail the features and explore the simulated performance of the connector system in various configurations and to demonstrate its ability to fulfill the requirements of numerous applications without requiring additional development and qualification.

II. Requirements

The minimum requirements for the connector were defined by Systems Engineering and Assessment (SEA) with reference to ECSS specifications where applicable. Table 1 contains the requirements as defined by SEA.

Requirements Table

Req. No.	***Requirement***	***Comment***
010	The SpW backplane connector shall fit 3U and 6U size cards.	To support the 2 proposed card sizes.
020	The SpW backplane connector shall permit 20mm pitch boards.	
030	The mass of the connector shall be minimised within the scope of environmental and material constraints.	
040	Both the connector body and contact materials shall be	ECSS-Q-ST-70C [REF 1] ECSS-Q-70-71 [REF 2]

Requirements Table		
Req. No.	***Requirement***	***Comment***
	compatible with ECSS standards for flight connectors.	ECSS-Q-ST-70-02 [REF 3]
050	The connector shall meet ECSS standards for durability.	ECSS-Q-ST-70C [REF 4]
060	The connector shall allow for through hole solder fitting to both the backplane and daughter-card.	
070	Alignment pins shall be provided on the connector.	To aid connector mating and prevent contact damage.
080	A minimum of 8 connector polarisation or keying options shall be provided.	To allow selective mating between connectors on different card types.
090	The connector shall support a minimum of 4 SpW links.	A SpW link is a full duplex communication path as supported by a standard SpW cable, (4 differential pairs).
100	Each differential pairs shall have an impedance of 100 Ω .	This is for all 4 differential pairs within a SpW link. To allow high speed communications.
101	Each SpaceWire differential pair shall be shielded from cross-talk.	
110	Each SpW pair shall support a data rate of at least 2.75Gbit/s.	To allow the use of SpFi Copper without changing the contact types. Ideally, the SpW pair should support a data rate of up to 10Gbit/s.
120	The SpW connector shall have 12 power pins and 12 returns (24 contacts) capable of 5A constant carry and a minimum of 85V rating for each contact.	Voltages envisaged to be +50V, +28V, +15V, +12V, -12V, +5V, -5V, +3.3V, 0V
130	The SpW connector shall have a minimum of 60 discrete contacts, each capable of 200mA (de-rated) constant carry and a minimum of 50V rating for each.	For carrying non-SpW signals.
140	There shall be a minimum of 12 contacts with a differential impedance of 100 Ω within the provided discrete contacts.	For use as LVDS clocks.

The connector will also meet the requirements of ECSS 3401 [REF 5], ECSS-Q-ST-70-08C [REF 6] & ECSS-Q-ST-70-38C [REF 7] where applicable. In addition to the requirements defined by SEA and ESA Smiths Connectors identified further requirements that were deemed attractive and incorporated those elements into the design of the connector.

III. Features & Performance

The connector is comprised of two main sub-assemblies; a socket contact housing backplane mounted receptacle and a 90º daughter-card mounted plug containing pin contacts. The connector (mated) is 97.5mm x 25 mm x 12.5mm enabling comfortable mounting of 1 connector on a 3U card or two connectors on a 6U card (see fig.1 & fig. 2).

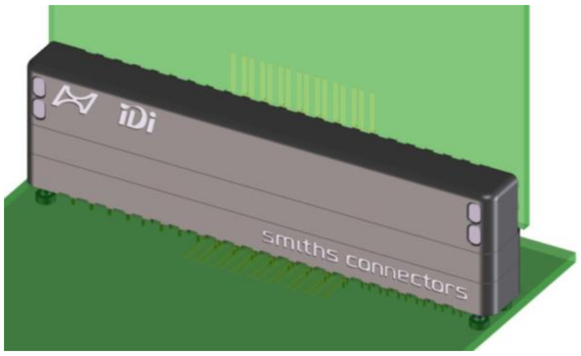


Fig. 1. Mated 3U connector, shown on 100mm PCBs.

The potential exists to expand the range to include a 6U all in one connector to increase the capacity of the overall solution. The 3U connector was designed with a capacity of 22 bays and can, therefore, house 22 modules. A single connector 6U solution would contain up to 49 modules. The 12.5 mm depth allows for adjacent connector stacking heights of ~ 12.7 mm in a surface mount (SMT) connector.

Located at either end of the plug connector are polarizing guide pins which also form part of the mounting system connector mounting system. Each of these pins can be positioned in one of 4 orientations giving a total number of 16 polarization permutations (see fig. 2).

Chassis and insulator components have been manufactured from a nonconductive Polyether Ether Ketone (PEEK) composite but can be supplied in a conductive or selectively conductive PEEK composite to improve electromagnetic compatibility (EMC). In the aforementioned screened state the cross-talk between modules is minimized. Mounting hardware is stainless steel.

The contacts located in both ‘halves’ of the connector are scoop proof, that is to say they are very well protected from accidental damage when unmated or during transport by virtue of being located wholly within small cavities (see figures 2, 3, 4 & 5).

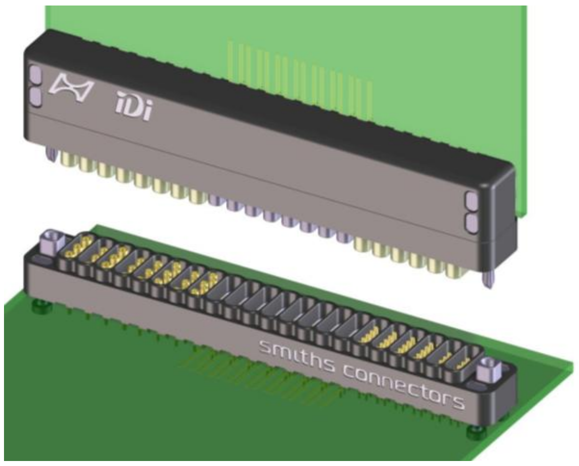


Fig. 2. Unmated 3U connector, shown on 100mm PCBs

A system for preferential mating has been devised and implemented for all module types, whereby early mate contacts will make electrical connections before late mate contacts (see figures 2, 3, 4 & 5). Mating interfaces of all contacts have a minimum of 1.27 µm of gold plating as per ECSS-3401.

All modules have been designed to maximise their dielectric withstand voltage (DWV) capability without the requirement for interfacial seals and therefore enabling a high DWV offering without the typically associated (and deemed to be undesirable) increase in mating force. Further features have been incorporated into the internals of the connector to reduce mating force, which, when coupled with the use of Hyperboloid sockets throughout, are intended to yield a backplane connector with industry leading low mating forces.

Both plated through hole (PTH) & SMT technologies are supported by the connector system. Connections to either and/or both printed circuit boards (PCBs) can be accomplished by three different methods, dependent upon application or customer preference; PC-tails suitable for plated through holes, surface mounted (soldered) and a solder-less, compliant, surface mounting method utilising spring probes.

A. Power3.1 module

Power3.1 modules contain 3 x 1 mm (size 20) contacts. These are rated at 7.5 Amps (A) nominally and are typically de-rated to 5 A when in a bunched configuration – i.e. when located in close proximity to one another whilst all are working at 5 A. Existing receptacle modules (see fig. 3) are configured with 3 x early mate contacts or with 3 x late mate contacts although a mixture of the two within one receptacle module is possible.

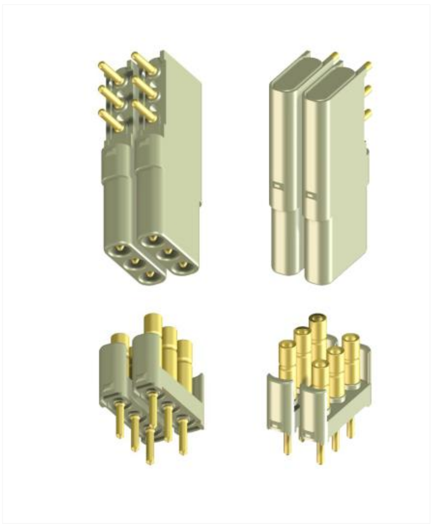


Fig. 3. Two Power3.1 daughter modules shown with early and late mate Power3.1 receptacle modules.

B. Power5.075 module

Power5.075 modules contain 5 x 0.75 mm (size 22) contacts. These are rated at 5 A nominally and are typically de-rated to 4 A when in a bunched configuration. Existing receptacle modules are configured with 5 x early mate contacts or with 5 x late mate contacts (see fig. 4) although a mixture of the two contact types within one receptacle module is possible.

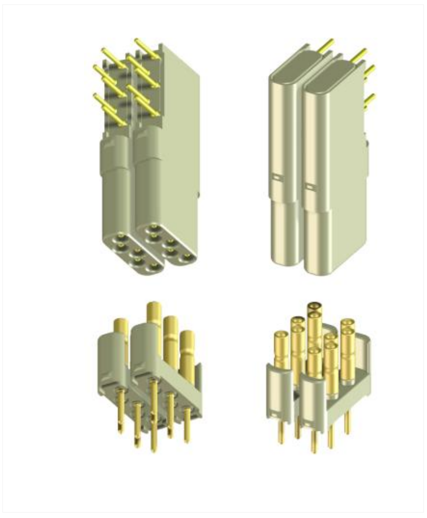


Fig. 4. Two Power5.075 daughter modules shown with early and late mate Power5.075 receptacle modules.

C. Signal10.04i module

Signal10.04*i* modules contain 10 x 0.4 mm contacts. These are rated at 400 mA nominally and are de-rated to 200 mA when in bunched configuration. Existing receptacle modules are configured as follows: early mate; containing 10 x early mate contacts, late mate; containing 10 x late mate contacts and mixed mate; containing 5 early mate and 5 late mate contacts, with the module exhibiting an early and late mate sidedness (see fig. 5). The module has been designed such that a higher rated contact can be used as a direct replacement to upgrade the current rating to 2 A nominally and 1 A de-rated whilst using the same mouldings (Signal10.04 module).

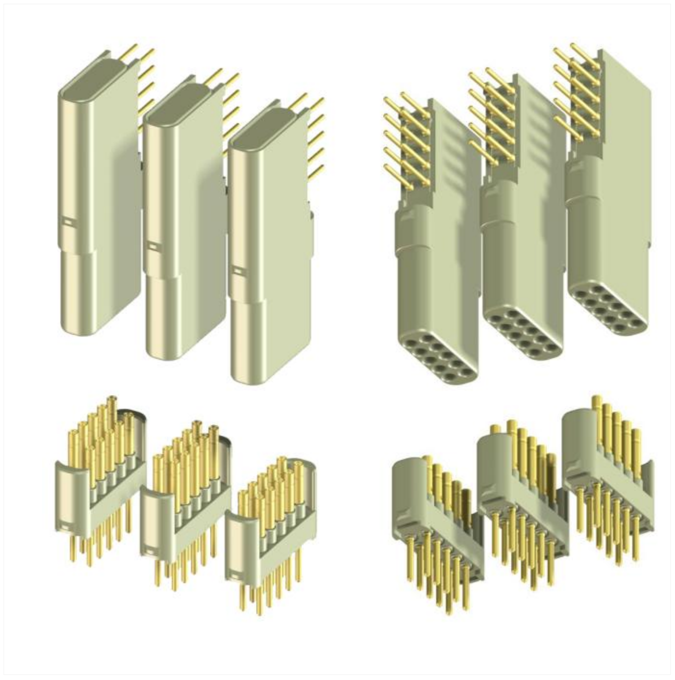


Fig. 5. Three Signal10.04*i* daughter modules shown with (from the left) mixed mate, early mate and late mate Signal10.04i receptacle modules.

The Signal10.04*i* module is impedance matched (100 Ω +/- 6 Ω differentially). Differential pairs are routed across the connector, i.e. there are potentially 5 rows of differential pairs in each Signal10.04i module. Within each differential pair there is no skew as each line in a differential pair is comprised of identical contact components to its partner. Figure 6 shows the return loss simulated from 0 to 8 GHz in CST microwave studio for each differential pair. The 1 ns rise time particular to the SpW transfer protocol is, for the sake of the interpretation of these results, considered to be approximately equivalent to the Gaussian pulse used for all simulations at 1 GHz. Ports were labelled sequentially, i.e. differential pair row 1 was port 1 to port 2, and row 5 was port 9 to port 10. Odd numbered ports were adjacent to one another, as were even numbered ports. Only the differential mode (mode 1) was simulated.

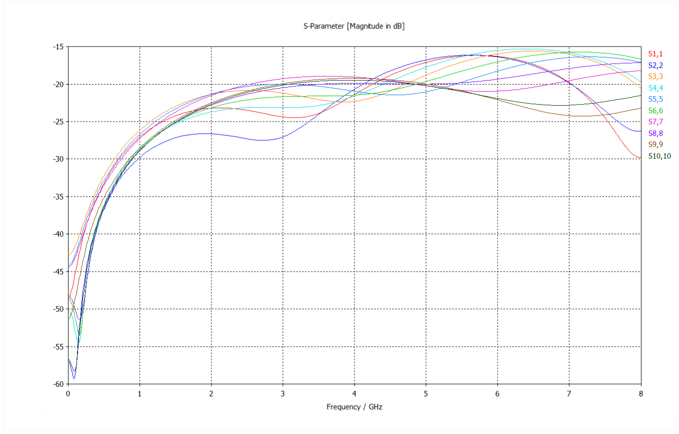


Fig. 6. Differential return loss for Signal10.04*i* module configured with 5 differential pairs.

If all contacts are to be utilised as differential pairs, cross talk will be at a maximum within the system. As the following figures show, cross talk (X-talk) between differential pairs progressively diminishes with distance from the excited pair. Figure 7 shows the near (s1,3) and far (s1,4) end cross talk (NEXT and FEXT), when port 1 is excited, and the NEXT (s2,4) and FEXT (s2,3) when port 2 is excited. Simulations were performed with a Gaussian pulse from 0 to 8 GHz. Simulated s-parameter data in figures 8, 9 & 10 follow the same nomenclature as used in figure 7.

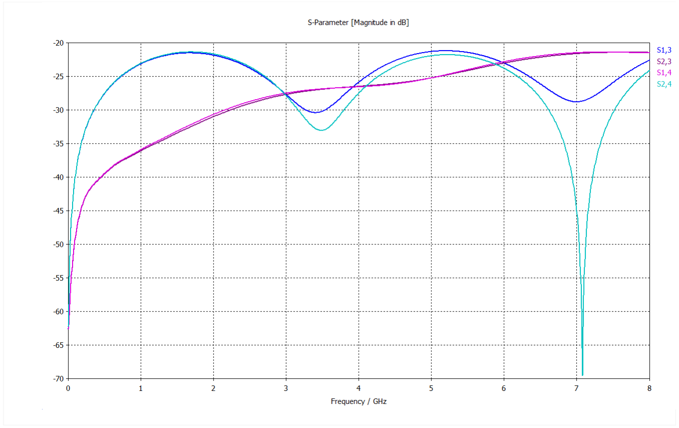


Fig. 7. Differential X-talk between pair 1 & pair 2.

Differential X-talk between pair 1 and 2 was approximately -23 decibels (dBs) for a 1 ns rise time pulse (see fig. 7).

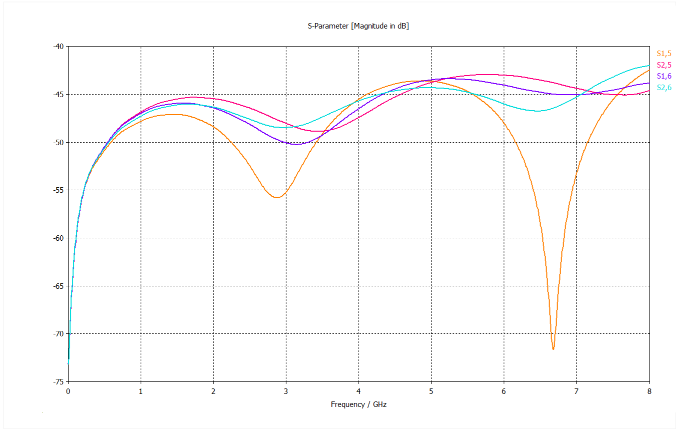


Fig. 8. Differential X-talk between pair 1 & pair 3.

Differential cross talk between row 1 and row 3 is approximately -47 dBs for a 1 ns rise time pulse (see fig 8).

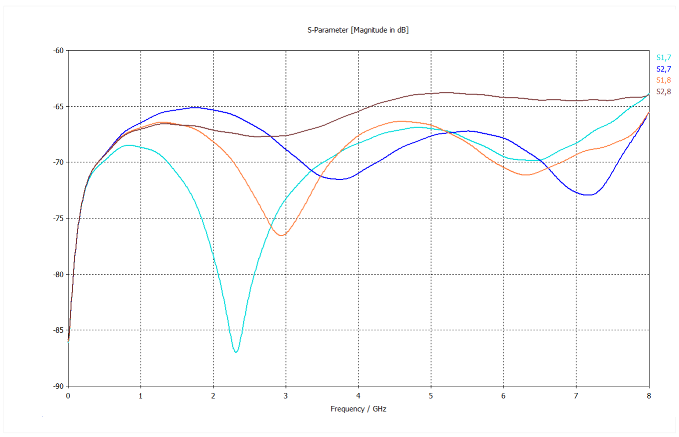


Fig. 9. Differential X-talk between pair 1 & pair 4.

Differential X-talk between row 1 and row 4 is approximately -66 dBs for a 1 ns rise time pulse (see fig 9).

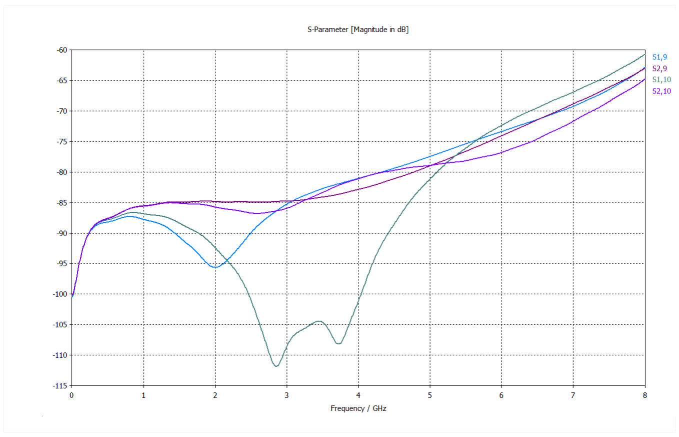


Fig. 10. Differential X-talk between pair 1 & pair 5.

Differential cross talk between row 1 and row 5 is approximately -85 dBs for a 1 ns rise time pulse (see fig 10).

The Signal10.04*i* module was also simulated with the intermediate pairs (rows 2 & 4) grounded to assess whether changes in configuration could be implemented to improve s-parameter performance. This configuration allows 3 differential pairs per module.

Differences in return loss performance were negligible between the two configurations and are not reported here.

The differential X-talk between pairs 1 & 3 with intermediate pair 2 grounded exhibited an improvement of approximately 20 dBs at 1 GHz (see figures 8 & 11).

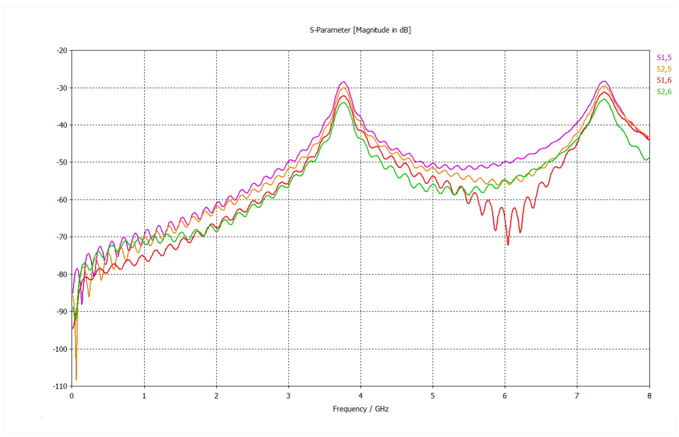


Fig. 11. Differential X-talk between pair 1 & pair 3 with pair 2 grounded.

The differential X-talk between pairs 1 & 5 with intermediate pairs 2 & 4 grounded exhibited an improvement of approximately 15 dBs at 1 GHz (see figures 9 & 12).

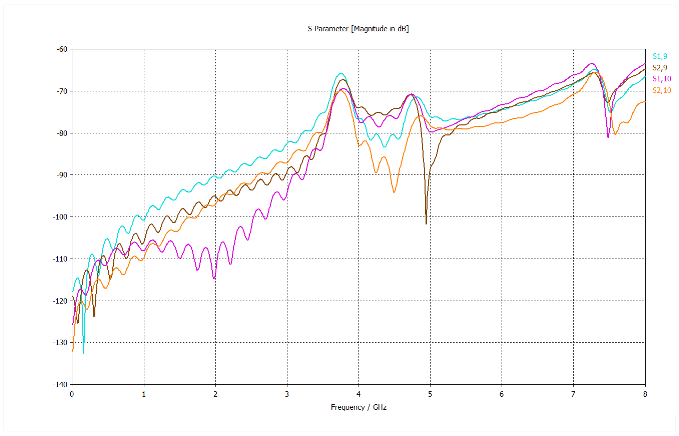


Fig. 12. Differential X-talk between pair 1 & pair 5 (pairs 2 & 4 grounded).

D. Quadrax Module.

The Quadrax module differs in its construction from all the previously described modules. The module housing is constructed from a conductive material (either manufactured from aluminium or a conductive composite) acting as a waveguide for each of the differential pairs and screening them from electromagnetic interference (EMI). The 90º transition is accomplished via a PCB minimising inter and intra differential pair skew; the connections to which are made compliantly with a robust methodology suitable and specifically designed for high speed transmission lines situated in high vibration environments. The configuration of the differential pairs within this module is columnar and therefore orthogonal to those located within the Signal10.04*i* module (see figure 13).

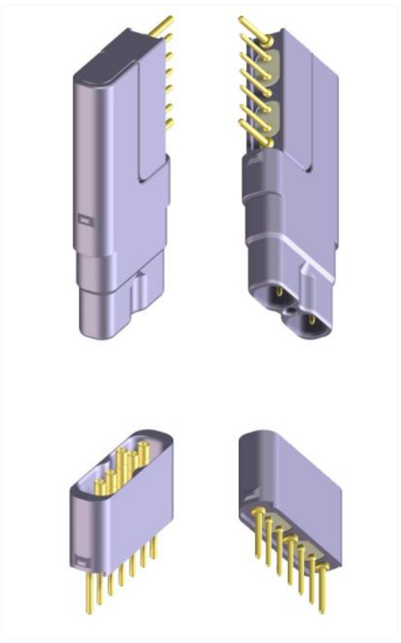


Fig. 13. Two Power3.1 daughter modules shown with early and late mate Power3.1 receptacle modules.

The transmission line characteristics of the Quadrax module differential pairs were simulated up to 20 GHz, corresponding to a digital pulse with a rise time of approximately 20ps. The simulated return loss s-parameters pertaining to the SpW protocol rise time were approximately -40 dBs in the worst case (see figure 14).

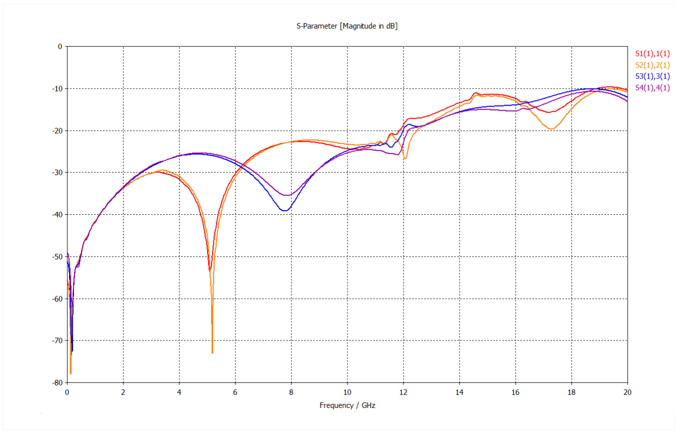


Fig. 14. Return loss of both differential pairs within the Quadrax module.

The simulated intra modular differential X-talk was approximately -80 dBs at the 1 GHz point under consideration in these analyses (see figure 15).

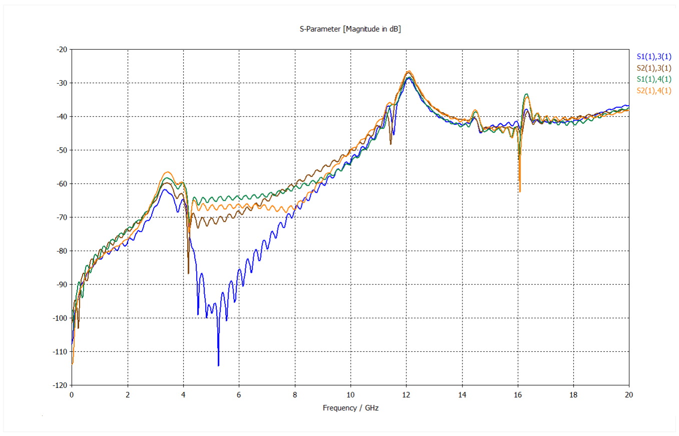


Fig. 15. Near and Far end X-talk between pairs within the Quadrax module.

IV. Conclusions

The goal of designing a robust highly configurable backplane connector system, capable of satisfying a large number of different signal types, was successfully achieved.

The 3U connector can be populated with up to 22 of the 4 types of module:

- Power3.1, containing 3 x 7.5 A contacts.
- Power5.075, containing 5 x 5 A contacts.
- Signal10.04(*i*), modules which can contain up to 5 differential pairs (400 mA) suitable for multi gigabit per second transmission rates or up to 10 discrete 2 A contacts.
- Quadrax, containing 2 differential pairs, suitable for multi gigabit per second transmission rates with exceptional screening performance.

The requirements set out by SEA and ESA have been designed for in full and in many cases exceeded.

The connector, in its simulated performance, is suitable for use in SpaceWire and is certainly a candidate for SpaceFibre Copper applications.

V. Further work

The prototype connectors currently being manufactured will be tested and their performance characterized at temperature and under vibration with respect to the following aspects:

- Return loss
- Insertion loss
- Near end cross talk (intra and inter modular)
- Far end cross talk (intra and inter modular)
- Insulation resistance
- Dielectric withstand voltage
- Electrical lengths of all signal paths.
- Mating forces (modular & connector)

Suitable test boards will be designed (& manufactured) in such a way as to enable the above outlined performance to be characterized in isolation.

Further development of the Quadrax module design will be performed to optimize all s-parameter performance characteristics at higher frequencies.

De-rating curves specific to the connector system will be derived from empirical data with respect to both bunching and temperature.

Assess the optimized, simulated and measured s-parameter performance against the requirements of SpaceFibre Copper.

Networks & Protocols (Long)

SpaceWire-RT/SpaceFibre Specification and Modeling

Session: SpaceWire networks and protocols

Long Paper

Valentin Olenev, Irina Lavrovskaya, Ilya Korobkov
Saint-Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russian Federation
Valentin.Olenev@guap.ru, Irina.Lavrovskaya@guap.ru, Ilya.Korobkov@guap.ru

***Abstract*— The growing autonomy of scientific missions to remote planets requires highly capable on-board networks that are robust and durable, able to recover from transitory errors and faults automatically. SpaceFibre is a very high-speed serial data-link being developed by ESA which is intended for use in data-handling networks for high data-rate payloads. SpaceWire-RT is then introduced as a logical development of SpaceWire and SpaceFibre, which aims to cover many on-board communications applications from low to very high data-rate networks. SpaceWire-RT standard is being developed under the scope of the 7th Framework programme. SpaceWire-RT uses the upper layers of SpaceFibre, which provide QoS, FDIR and multi-laning. The lower layers could be represented by both SpaceWire and SpaceFibre lower layers.**

The specification of the SpaceWire-RT standard, and its modeling is one of the key aspects of the proposed research programme. From the textual specification a formalised specification for the SpaceWire-RT was developed using the SDL language, which is itself a test of the specification for completeness and unambiguousness. As a result the consistent readable textual description and formalised specification in SDL were produced. The SystemC simulation language was used to model reconfigurable SpaceWire-RT networks with multiple nodes and routing switches. Such models can be used for investigating and proving the network level features and characteristics of the novel SpaceWire-RT technology: scalable performance, responsiveness, robustness, provision of quality of service, and ultimate low latency signalling.

The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013]) under grant agreement n° 263148. This paper gives and overview of the specification and modeling work that is done by SpaceWire-RT Consortium. The main focus of the paper is application of modeling for the protocol stack development purposes, which advantages could it give and what the implemented models could be used for.

***Index Terms*— SpaceWire-RT, SpaceFibre, modeling, specification, protocols, SDL, SystemC**

I. INTRODUCTION

Simulation plays a very important role in a process of communication protocols design. It is used to validate the specification and to find ambiguities and inconsistencies in it. This was one of the key tasks in the newest onboard communication standard developments flow. The SpaceWire-RT standard is developed in the scope of the 7th Framework Program project. The main project's task is to conceive and create communications network technology, suitable for a wide range of demanding space applications where responsiveness, determinism, robustness and durability are fundamental requirements. This is a critical component technology for future spacecraft avionics and payloads.

The project began with gathering of the requirements for the future network technology from Russian and European companies involved in the space and avionics industry. Based on these requirements the draft of the SpaceWire-RT specification was produced. So the next step was to validate the specification and check the standard for inconsistencies, ambiguities and errors in developed mechanisms. This was done by using the simulation models of SpaceWire-RT.

The current paper gives an overview of the developed SpaceWire-RT standard and presents the simulation and research results. Moreover, it provides the description of SDL point-to-point model and SystemC network model that were implemented for the SpaceWire-RT simulation and validation.

II. SPACEWIRE-RT

Aerospace industry is one of the most rapidly growing areas in terms of communication protocols development. Avionics and robotics impose requirements on network responsiveness and determinism. The increasing international collaboration on scientific and Earth observation spacecraft requires standard network technology where a component developed by one nation will interoperate effectively with equipment developed by another nation. SpaceWire-RT is the newest onboard communication protocol standard, which aims to fulfill these demanding requirements with a flexible, robust, responsive, deterministic and durable standard network technology that is able to support both avionics and payload data-handling applications [1].

SpaceWire-RT aims to cover many on board communication applications from low to very high data-rate networks. This is a critical component technology for future spacecraft avionics and payloads.

SpaceWire-RT will:

- use virtual channel concepts to provide a variety of QoS;

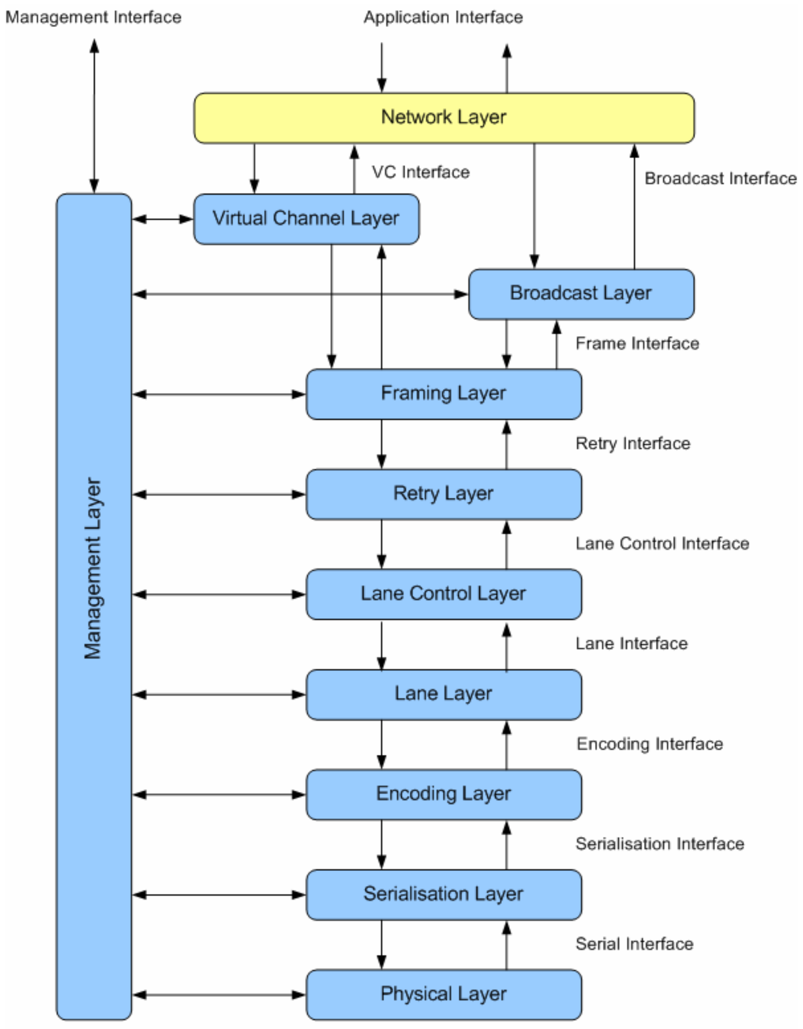


Figure. 1. Overview of SpaceWire-RT layered architecture

- provide broadcast and multicast capability;
- increase performance;
- provide low latency message delivery;
- include extremely low latency time and out-of band signalling mechanisms;
- incorporate novel fault detection, isolation and recovery methods;
- make the network fully responsible for information transfer;
- decouple application and data transfer;
- implement appropriate communication mechanisms in relatively simple hardware.

SpaceWire-RT standard is based on the SpaceFibre technology [1].

SpaceFibre is a very high-speed serial data-link, which is intended for use in data-handling networks for high data-rate payloads. SpaceFibre is able to operate over fibre optic and copper cable and support data rates of 2 Gbit/s in the near future and up to 20 Gbit/s long-term. SpaceFibre will provide a coherent quality of service mechanism able to support best effort, bandwidth reserved, scheduled and priority based qualities of service. It will substantially improve the fault detection, isolation and recovery (FDIR) capability of SpaceWire [11].

SpaceFibre provides robust, long distance communications for launcher applications and will support avionics applications with deterministic delivery constraints through the use of virtual channels. SpaceFibre enables a common onboard infrastructure to be used across many different mission applications resulting in cost reduction and design reusability. SpaceFibre can run over fibre optic or copper cables [11].

The CODEC design for SpaceFibre has many advantages in comparison with SpaceWire [5,11]:

- It uses fewer wires reducing cable mass;
- It operates at data rates of 2 Gbits/s and potentially higher;
- It uses matched impedance connectors;
- The size of all the characters are the same (32-bits);
- Parity coverage is per character;
- It uses a DC balanced encoding scheme;
- It provides simple capacitive, magnetic, or optical galvanic isolation;
- The initialisation protocol is base on a double handshake [1].

An overview of the SpaceWire-RT architecture is provided in Fig. 1.

Although the SpaceWire-RT is primarily based on the SpaceFibre, it has an additional functionality which is necessary for network operation. Therefore, the SpaceWire-RT comprises SpaceFibre protocol layers which are complemented by the Network Layer. The *Network Layer* is responsible for routing SpaceWire-RT packets over a SpaceWire-RT network, comprising SpaceWire-RT routing switches, SpaceWire-RT links, and SpaceWire-RT nodes. Moreover, it is also responsible for validating and broadcasting SpaceWire-RT broadcast messages over a SpaceWire-RT network.

In turn SpaceFibre defines ten conceptual layers [11]:

- *Virtual Channel Layer* which *is* responsible for quality of service and flow control over the SpaceFibre link.
- *Broadcast Layer* which is responsible for sending short broadcast messages across a SpaceFibre link and for receiving those messages.
- *Framing Layer* which is responsible for framing SpaceWire-RT packet data, broadcast messages and FCTs to be sent over the SpaceFibre link. It is also responsible for scrambling SpaceWire-RT packet data for EMC mitigation purposes.
- *Retry Layer* which is responsible for recovering from transient errors on the SpaceFibre link, and for reporting errors and link failure. It also detects missing and out of sequence frames.
- *Lane Control Layer* which is responsible for managing the operation of several SpaceFibre lanes in parallel to provide a higher data throughput and to provide redundancy with graceful degradation.
- *Lane Layer* which is responsible for initialising the lane, detecting lane errors and re-initialising the lane after an error has been detected.
- *Encoding Layer* which is responsible for encoding the data and control words into a suitable form for sending over the SpaceWire-RT link and decoding received data and control words. Uses 8B/10B encoding.
- *Serialisation Layer* which is responsible for serialising and de-serialising encoded data and control words for sending and receiving over the serial interfaces.
- *Physical Layer* which is responsible for sending the SpaceWire-RT information over the physical media used in SpaceWire-RT: fibre optic, Current Mode Logic (CML) and Low Voltage Differential Signalling (LVDS).
- *Management Layer* which is responsible for configuring, controlling and monitoring the status of the various layers of the SpaceWire-RT protocol stack. This can be done by a local or remote network management application [1].

From the textual specification a formalised SDL model of the SpaceWire-RT was developed, which is itself a test of the specification for completeness and unambiguousness. In turn, the SystemC simulation language was used to develop reconfigurable SpaceWire-RT networks model with multiple nodes and routing switches. The following sections give the description of SDL and SystemC models and provide with the results of the simulation.

III. SpaceWire-RT P2P Model in SDL

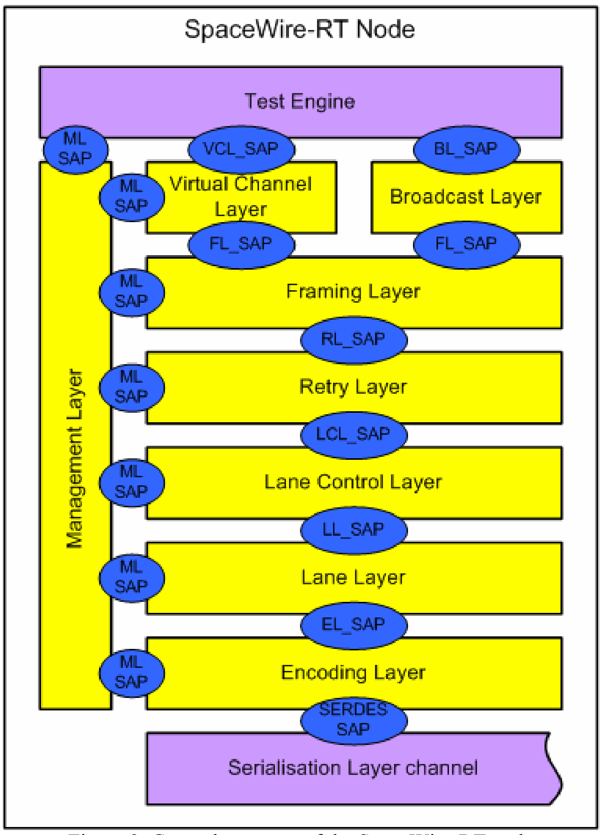


Figure 2. General structure of the SpaceWire-RT node

SDL (Specification and Description Language) is a language for unambiguous specification and description of the telecommunication systems behavior. The SDL model covers the following five main aspects: structure, communication, behavior, data and inheritance. SDL language is intended for description of structure and operation of the distributed real-time systems. Writing an SDL model on the basis of the specification is itself a test of the specification for completeness and unambiguousness. As a result the consistent readable textual description and formalised specification in SDL are produced [7].

SDL language was used for SpaceWire-RT specification and simulation on a per layer basis as the most reasonable solution. The SDL model formally describes all mechanisms, interactions and functionality which are stated in the SpaceWire-RT specification [7].

The SDL model implements all layers of the SpaceWire-RT protocol stack (excepting Serialisation Layer). Figure 2 shows the general structure of the SpaceWire-RT node in SDL.

According to the figure above the SDL model describes the internal mechanisms and functionality of the layers starting from the Encoding Layer and up to the Virtual Channel Layer. Each pair of adjacent layers communicates via a special interface between them, which is called a Service Access Point

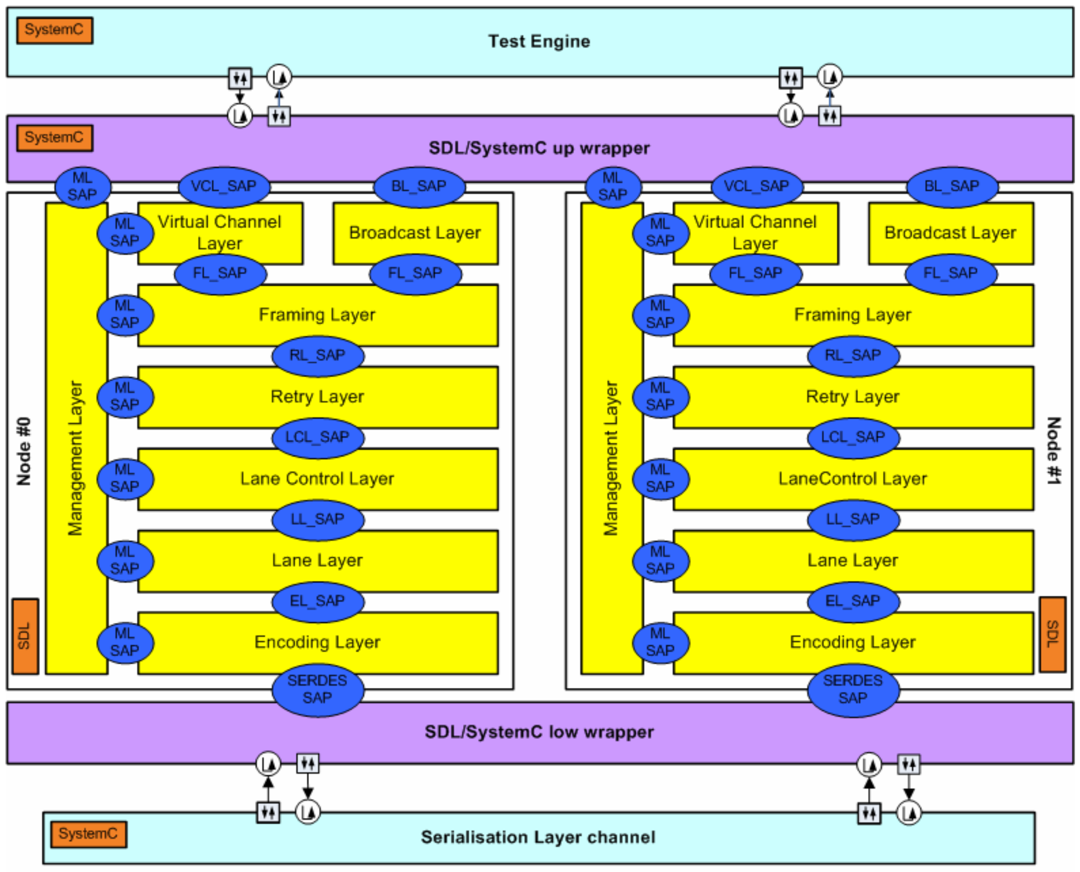


Figure 3. General structure of the SpaceWire-RT SDL/SystemC tester

(SAP). All SAPs are defined as sets of service primitives which are specific for each layer.

Simulation and investigation was done in two steps. First of all we performed verification of the SpaceWire-RT protocol stack by simulation in IBM Rational SDL Suite. The test system was represented by the two SpaceWire-RT nodes communicating through the Serialisation Layer channel. Configuration and generation of test sequences was performed by a special Test Engine. This simulation gave an ability to check all internal mechanisms of investigated layers and verify them.

The second step was validation of the SpaceWire-RT protocol stack by load testing by means of simulation within an SDL/SystemC tester. The SDL/SystemC tester provides a possibility for simulation of a point-to-point interconnection between two nodes, implemented in SDL and communicating via a channel. The tester is a flexible tool for setting different configurations, generating various test sequences and gathering statistics.

The general structure of the SpaceWire-RT SDL/SystemC tester is given in Fig. 3.

The SystemC Test Engine provides facilities for creation of different complicated test sequences with different configurations and efficient logging of the events in the model.

In order to implement interconnection between the SpaceWire-RT SDL model and the test environment an SDL/SystemC co-modeling approach was used [6]. This approach assumes that special wrappers (SDL/SystemC up wrapper and SDL/SystemC low wrapper) should be implemented for conversion of data from the SDL representation to the SystemC representation and vice-versa. The wrapper receives SystemC data, converts it into the SDL signals and sends to the SDL model via the correspondent SAP. Thus, the up wrapper is responsible for communication of test engine and SDL model and the low wrapper is responsible for communication of the SDL model and the channel.

The target SDL model of the whole SpaceWire-RT protocol stack can be used for checking, how all mechanisms operate in common in one node by means of simulation in the IBM Rational Tool and by means of simulation within the SDL/SystemC tester. This way, the SDL layered SpaceWire-RT model can be used for validation of consistency of the specification and checking of functional requirements, defined for the standard. The main advantage of this model is that it is implemented in a formal high-level language. This model can be used for further investigation of SpaceWire-RT technology because any changes in new versions of the standard could be applied to the SDL specification without any changes in the test environment.

IV. SpaceWire-RT Network Model in SystemC

The SystemC modeling is one of the most efficient and widely used methods for studying, analysis and constructing multi-component systems, such as stacks of protocols, embedded networks of a large number of nodes, systems-on-chip, networks-on-chip, etc.

SystemC is a set of C++ classes and macros that provide an event-driven simulation engine. It is specifically designed for modeling parallel systems. This library allows describing multi-component systems and program components, and modeling their operation. By using the internal mechanism of events it allows to model operations distributed in time of the modeled system [3].

The aim of the network SpaceWire-RT model development is to simulate communication of devices (switches and nodes) via the SpaceWire-RT links. In the SystemC network model some interactions of components and processes inside the device (e.g. between levels of a stack) could be not considered, because the model is primarily focused on the mechanism of devices' communications, such as transfer of packets, routing and performance characteristics of the network [7].

The SpaceWire-RT network model consists of the following SystemC modules:

- SpaceWire-RT stack model, which provides main functions of SpaceWire-RT;
- SpaceWire-RT node model;
- SpaceWire-RT switch model.

The SpaceWire-RT stack model is a part of the node and the switch models. For these models it is possible to set different parameters like: a data transmission speed (Gbps), a number of nodes and switches, size and amount of packets, a destination address for a particular packet, a time delay and a routing table for the switch, a number of ports in the switch, etc.

The SpaceWire-RT network model contains a number of nodes and switches. It could contain no switches so the model would be point-to-point. It gave an ability to simulate operation of the various number of devices in a network with different topologies: point-to-point, tree and circular.

Using a point-to-point configuration we had an ability to check correctness and consistency of the SpaceWire-RT stack specification. An example of the point-to-point configuration is shown in Fig. 4.

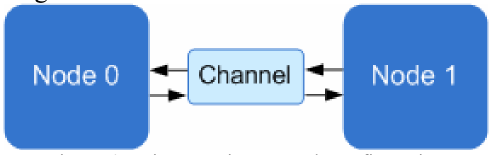


Figure 4. Point-to-point network configuration

For testing of network mechanisms we used the mixed configuration, which is a combination of tree and circular topologies. Mixed configuration gave an opportunity to check the following network parameters: latency for different packet sizes, reliability of data transfer with specific BER (Bit Error Rate), various QoS (Quality of Service), fault packet detection and identification, failure and fault tolerance of a network (deadlock and babbling idiot), broadcast and multi-cast, path and logical addressing. An example of the used mixed network configuration is given in Fig. 5.

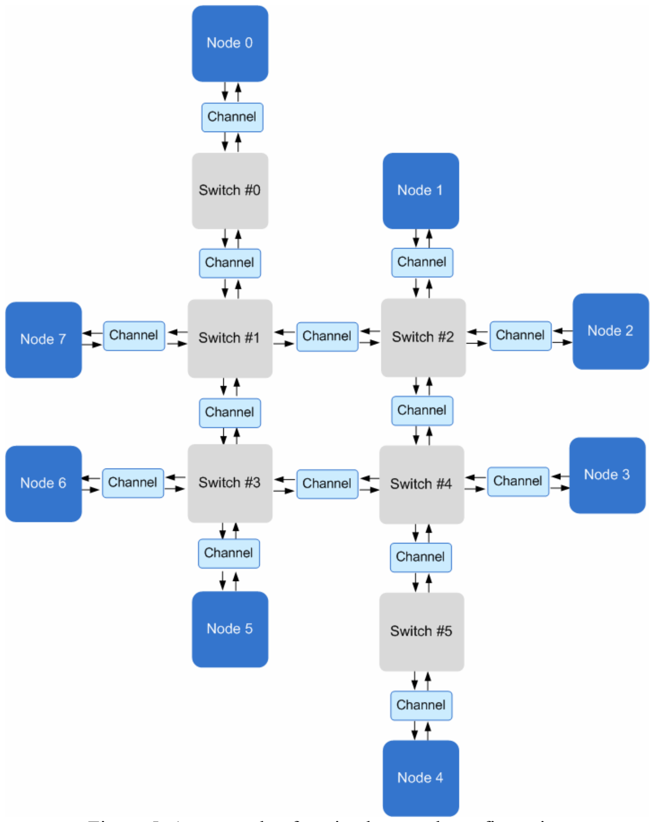


Figure 5. An example of a mixed network configuration

V. Simulation Results

Beforementioned SDL and SystemC models were used for the scientific studies and research. SpaceWire-RT standard was checked on conformance to the Russian and European industry requirements and also on existence of inconsistencies and ambiguities. Some of the most important results of our research are given below.

A. SpaceWire-RT packet length

The SDL model was used for testing and proving a possibility of transmission of a 32 Mbytes packet over a single link (as the SDL model provides only point-to-point simulation). Simulation proved that a 32 Mbytes packet can be successfully transmitted over a link. Transmission took 160 ms at a data rate 2 Gbit/s. The SystemC network model also proved the possibility of a 32 Mbytes packet transmission. Moreover, SystemC simulation provided the results for the packet lengths of 32 Mbytes to 8 bytes.

For such kind of test we used the network configuration shown in Figure 5. During simulation we measured latencies for packet delivery between Node 2 and Node 8.

The latency results for each packet length are given in Table 1.

TABLE I. Latency Results

Packet length	Latency
32 Mbytes	247,84 ms
16 Mbytes	123,92 ms
8 Mbytes	61,95 ms
512 Kbytes	3,86 ms

Packet length	Latency
64 Kbytes	474,2 µs
32 Kbytes	239,9 µs
16 Kbytes	124,9 µs
8 Kbytes	67,9 µs
512 bytes	14,8 µs
256 bytes	13,5 µs
8 byte	665,352 ns

B. Broadcast data transfer

SpaceWire-RT standard supports broadcast data transfer mechanism which is described in SpaceFibre. The broadcast mechanisms were tested on three different topologies: tree configuration, circular configuration and mixed configuration. The broadcast messages were successfully generated, sent and delivered to all destination Nodes in the SpaceWire-RT network. However, the latency for the broadcast messages does not fit the requirement of 100 ns even if we have only one switch between the nodes. So the requirement is too strict for the current technology.

Moreover, the network simulation shown that the current version of the SpaceWire-RT standard does not provide the mechanism for the broadcast messages to be discarded in Switches during the repeated transmission over a network with a circular structure.

C. Reliability

SDL and SystemC simulation shown that SpaceWire-RT, based on SpaceFibre, provides a capability for reliable data delivery. It provides a mechanism of automatic acknowledgements (ACK) and negative acknowledgements (NACK), which are used for indication about the validity of the received data in the Retry Layer. To check this mechanism in SDL a channel with a capability of errors insertion was used. For the purpose of acceleration of getting testing results we increased BER for the channel and assumed it equal 10^{-6}. Simulation shown, that all transmitted data was delivered correctly. The increased BER (e.g. 10^{-5}) leads to a corruption of a large amount of data, so that the connection cannot be established or re-established in some cases.

The same test was used for the SystemC point-to-point topology. The ACK/NACK mechanism works and the remote Node successfully got the data.

D. Determinism

The determinism requirement was checked on SystemC model. For proving this requirement the scheduling QoS was tested on the network model. The topology used for testing this requirement is shown in Figure 5. Testing was done using 2 different schedules. Both of them used three virtual channels for the transmission of data packets.

The SystemC SpaceWire-RT network successfully operated using both test schedules. So SpaceWire-RT provides deterministic data delivery using scheduling mechanism.

E. Automatic acknowledgement

SpaceFibre is the basis for the SpaceWire-RT and one of its main features is reliable data delivery. This is achieved by the automatic acknowledgements on the Retry Layer of SpaceFibre. This mechanism was tested by the simulation of SDL model of the Retry Layer. During the simulation, data was transmitted via a channel. This channel corrupted transmitted data with the BER = 10^{-6}. All transmitted data was delivered correctly.

Simulation shown that SpaceWire-RT, based on SpaceFibre, provides automatic acknowledgement mechanism at the Retry Layer which is not configurable. However, this requirement relates to the end to end acknowledgement and not to a link level function like the SpaceFibre retry. So an automatic acknowledgement should be implemented above the network layer.

F. Failure and fault tolerance

The failure and fault tolerance of network had been checked on the SystemC network model. The Network Layer of the SpaceWire-RT provides mechanisms for failure and fault tolerance of a network. These mechanisms were checked by testing the following situations:

- Deadlock. Some kind of deadlock situation was modeled by a special configuration of Switches and Channels. The Channel between a Switch and a Node was made full so the data transmission for the particular VC stopped. All the other data transmissions for this VC to this Node also stopped. The model started to work slower, but the transmission of the data to the Node via the other virtual channels did not stop. This test shown that even in a deadlock situation for one virtual channel, the other virtual channels will continue to send data.
- Babbling idiot. The traffic generator of a Node continuously generated 8 bytes packets to all the network addresses using all available virtual channels. The most part of the packets were discarded during the transmission through Switches but some packets were transmitted to Nodes. This made the transmission of the data in the whole network much slower. But anyway the other data traffic from other Nodes was successfully delivered to the destinations.

VI. SpaceWire-RT Models Future Use

Currently, the SpaceFibre standard, which is the basis for the SpaceWire-RT standard, is in process of development. Consequently, any changes in SpaceFibre will result in changes of the SpaceWire-RT specification. New mechanisms and updated old ones can be successfully simulated on the SpaceWire-RT SDL and SystemC models by changing necessary parts of the models. Although the specification of the standard is updated, the test environment will remain the same.

Any changes in the new releases of the specification can be applied to the SpaceWire-RT SDL model. This would not cause any difficulties, as local changes in one layer will not affect the other layers. Therefore, the SDL model can be used for verification of the new mechanisms in a stack. Moreover, the SDL/SystemC tester gives opportunities for creation of complicated test sequences and for non-nominal testing.

In its turn, the SystemC model can be effectively used for obtaining network performance characteristics such as latencies, QoS mechanisms operation, etc. Since the SpaceWire-RT Network Layer is currently under development the SystemC model can be applied for its further validation. Moreover, SpaceWire-RT SystemC network model can give an opportunity for investigation of Transport Layer protocols operation over the SpaceWire-RT network (e.g. RMAP, STP, etc.).

Finally, SDL and SystemC models can be used for development of applications and drivers for future SpaceWire-RT devices.

VII. Conclusion

The paper describes the new SpaceWire-RT technology. SpaceWire-RT standard was validated by means of SDL and SystemC models and was checked on conformance to the Russian and European industry requirements. A number of inconsistencies in the specification were found during the simulation and some solutions and additional mechanisms were proposed to solve them. The new version of the SpaceWire-RT standard is produced and it is based on the simulation impact. The latest news and results of the project are available on our website http://www.spacewire-rt.org.

References

[1] S. Parkes, “SpaceWire-RT Outline Specification, version 2.1”, University of Dundee, 6th September 2012.

[2] International Telecommunication Union, “Recommendation Z100: Specification and Description Language (SDL)”, 2007.

[3] Open SystemC Initiative (OSCI), “IEEE 1666™-2005 Standard for SystemC”, 2005.

[4] J. Gipper, “SystemC the SoC system-level modeling language. Embedded computing Design”. 2007.

[5] ESA (European Space Agency), standard ECSS-E-50-12A, “Space engineering. SpaceWire – Links, nodes, routers and networks. European cooperation for space standardization”, ESA Publications Division ESTEC, Noordwijk, The Netherlands, 2003.

[6] S. Balandin, M. Gillet, I. Lavrovskaya, V. Olenev, A. Rabin, A. Stepanov, “Co-Modeling of Embedded Networks Using SystemC and SDL”, International Journal of Embedded and Real-Time Communication Systems (IJERTCS), IGI Global, pp. 24-49, 2011.

[7] Y. Sheynin, E. Suvorova, V. Olenev, I. Lavrovskaya, “D3.1 SpaceWire-RT Simulation and Validation Plan”, Saint-Petersburg State University of Aerospace Instrumentation, 3rd October 2012.

[8] V. Olenev, “Different approaches for the stacks of protocols SystemC modelling analysis”, Proceedings of the Saint-Petersburg University of Aerospace Instrumentation scientific conference, Saint-Petersburg University of Aerospace Instrumentation (SUAI), Saint-Petersburg, pp. 112-113, 2009.

[9] A. Jantsch, “Modeling Embedded Systems and SoCs”, Morgan Kaufmann Publishers, Stockholm, 2004.

[10] Y. Sheynin, T. Solokhina, Y. Petrichkovitch “SpaceWire technology for the parallel systems and onboard distributed systems”, ELVEES, 2006, http://multicore.ru/fileadmin/user_upload/mc/publish/SpW-part1.pdf.

[11] Parkes SM, Ferrer Florit A, Gonzalez A, and McClements C, “SpaceFibre”, Draft E1, Space Technology Centre, University of Dundee, 28th September 2012.

OPNET Modeler® Co-Simulation for Modeling SpaceWire Plug-and-Play Protocols

SpaceWire Networks and Protocols, Long Paper

Sandra G. Dykes, Carlos Quiroz, Paul Wood, and Allison Bertrand
Communications and Embedded Systems Department
Southwest Research Institute®
San Antonio, TX, USA
sdykes@senomedical.com, cquiroz@swri.org, pwood@swri.org, abertrand@swri.org

***Abstract*—Co-Simulation is a method for integrating external hardware and/or software with a simulation model, commonly referred to as "in-the-loop" simulation. We describe a co-simulation investigation that combines the Space Plug-and-Play Architecture (SPA) Services Manager (SSM) software with a newly developed OPNET Modeler SpaceWire model. The advantage of this approach is that spacecraft designers can rapidly and accurately verify behavior for a variety of topologies and use cases. To simplify tool use, we are building a library of spacecraft components with common traffic generation characteristics. This paper describes the co-simulation implementation, the OPNET SpaceWire model, our spacecraft component library, and a set of simulation studies.**

***Index Terms*—SpaceWire simulation, OPNET Modeler, SpaceWire model, SPA, SSM, Space Plug-and-Play Architecture.**

I. Introduction

Simulation is a well-established practice used in the design of network infrastructure. SpaceWire (SpW) networks are no exception and simulation network designers benefit from the ability to experiment with various hardware configurations and topologies before committing to a specific design.

OPNET Modeler is a commercial, dynamic discrete event simulator widely used for analyzing and designing communication networks, devices, protocols, and applications. Its co-simulation interface allows the simulation model to be integrated with external hardware and software systems. This "in-the-loop" approach enables a simulation to measure the network contention and delays incurred by actual components under various scenarios.

This paper describes a co-simulation approach that combines an OPNET Modeler SpaceWire model with a software implementation of the Space Plug-and-Play Architecture (SPA-S) middleware, the SPA Services Manager (SSM). The SSM is a software layer between the application and the SpaceWire interface that provides network discovery and message delivery services. In our approach, one or more external hosts run the SSM and application processes that model spacecraft components, such as imagers, StarTrackers, or data recorders. These external host processes generate traffic that is passed through the OPNET model. Inside the simulated network, the system measures congestion on links and at router ports that result from a combination of network topology and traffic load. Congestion delays may cause messages to miss deadlines or cause retransmissions that further increase congestion. Moreover, port blocking can delay high priority messages, potentially causing serious faults. Our simulation tool reports statistics on delays, missed deadlines, and other quality of service (QoS) metrics, in addition to logging data that can be used to understand behavior.

We describe the process of building and operating the co-simulation, the implementation of the SSM, and the results of a series of simulation studies. Finally, we analyze the results to expose factors that influence performing a simulation in this way and look at benefits/disadvantages and possible limitations of co-simulation with external software.

II. Background

The Air Force Research Laboratory (AFRL) is pursuing the goals of standardizing the Space Plug-and-Play Architecture (SPA) standards along with supporting the development of various tools to aid network designers and developers in the successful application of the standards.

To achieve the first goal, AFRL has been working to have the SPA standards adopted and published by American Institute of Aeronautics and Astronautics (AIAA). Although the SPA has been recently renamed the Modular Open Network Architecture (MONARCH), we have used SPA throughout this paper.

The Space Dynamics Lab (SDL), under AFRL contract, has been working on the development the SPA software middleware known as SPA Services Manager (SSM). The SSM implements the functionality described in the SPA standards.

AFRL's second thrust has been in the area of SpW network simulation. AFRL has contracted with OPNET Technologies, Inc. to produce a low-level network simulation module for OPNET Modeler. This simulation captures SpW interactions between nodes and routers at the character level and includes flow control characteristics such as the credit counts and flow control tokens that are part of the SpW standard.

In addition to the OPNET Modeler SpW module, simulation of the SPA is desired. The SPA behavior is complex, and direct integration into OPNET Modeler is not practical. Although OPNET Modeler includes a mechanism to create custom behaviors for network components including nodes, the multi-process implementation of the SPA is not conducive to direct integration into OPNET Modeler. Also, SPA complexity is such that rewriting the behavior directly in OPNET Modeler would be excessively time consuming and costly.

Thus, an alternative approach was developed that takes advantage of SSM, which implements SPA. This approach uses the "co-simulation" capability of OPNET modeler to link externally running SSM processes (i.e., producers and consumers) into the OPNET Modeler SpW simulation in order to give a high fidelity result that incorporates the complex behavior of the SPA middleware into a network simulation.

The OPNET Modeler co-simulation facility provides an Application Program Interface (API) that allows an external program to interact with and control the OPNET simulation. This external program provides the interface that links the simulation to independent devices and applications. This interface forwards data between nodes in the simulation model and the physical nodes they represent. For example, the interface can receive SpaceWire packets from a physical producer node and forward them to the corresponding producer node in the simulation. This approach allows for the full fidelity of the external programs to be included in the simulation model.

Co-simulation raises a unique problem in synchronizing simulation time to the wall clock time on the physical devices. In some cases, the physical hosts must use simulation time. For example, a node that sends a network discovery probe should use simulation time for the timeout for the reply. However, there are other cases when the physical host should use its local clock time. An example is the timeout for acquiring a mutex or semaphore. In this case, the events are local to the physical host and should use wall clock time rather than simulation time.

If the simulations ran faster than real time, the time could easily be synchronized by slowing the simulation. However, the OPNET SpW simulation, like most simulations, runs slower than real time. Time synchronization therefore requires adjusting event times on the physical systems. The balance of this paper describes details of the simulation approach with a focus on describing how this issue of time synchronization was accomplished.

III. Design of SPA Co-Simulator

A. Co-simulator System Design

The OPNET/SSM Co-Simulation system is composed of external hosts which generate and consume traffic used as input and outputs of an OPNET simulation, creating a "system-in-the-loop" environment (see Figure I). The system can be broken down into the following three pieces:

1. External hosts running SSM on Linux PC's
2. OPNET/SSM Co-Simulation Controller
3. OPNET simulation with custom SpaceWire models

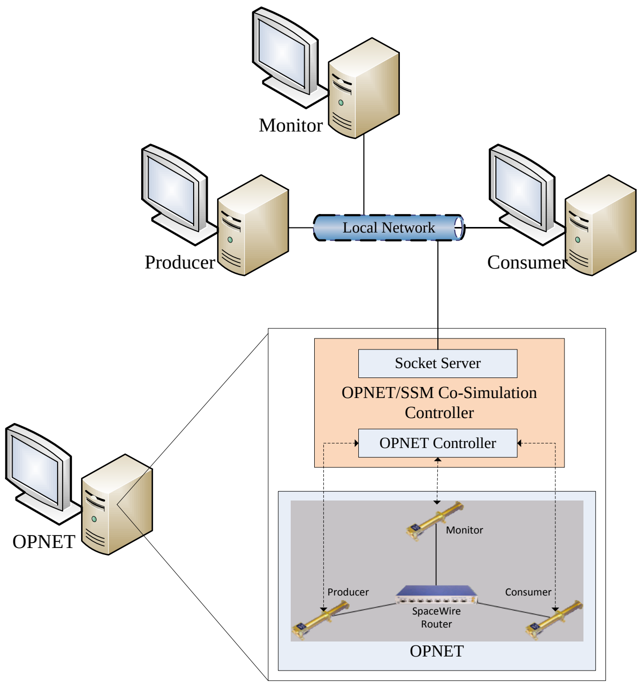


Fig. 1. OPNET/SSM Co-Simulation System Overview

In an SSM system, several processes execute on each node, coordinating via shared memory that is protected by semaphores. Table I contains a list of the SSM components and a description for each.

TABLE I. SSM Components

Service	Description
Producer Application	The Producer Application publishes a notification message and accepts a command which alters the data being published.
Consumer Application	The Consumer Application queries for and subscribes to data provided by the Producer Application.
SPA Local Manager	The SPA Local Manager allows SPA components to interoperate on a local processing node.
SPA SpW Manager	The SPA SpW Manager is responsible for performing discovery for a particular subnet. It maps incoming packets to the correct SPA endpoint on the subnet, encapsulating the SPA packet with the correct protocol header. In the reverse direction, it removes the protocol header and possibly adds a new header conforming to the subnet the packet is about to enter. It is also responsible for topology discovery and reporting within the subnet. The SPA SpW implements the interface that transmits and receives SPA messages over a network.
Central Addressing Service	The Central Addressing Service (CAS) is responsible for providing logical address blocks to be assigned to each hardware or software component. The CAS stores the logical address block and logical address for each SPA Manager in the SPA Network.
SPA Lookup Service	The SPA Lookup Service is responsible for accepting component registration and providing data source route information for components requesting a particular type of service.

For the studies presented in this paper, three external SSM hosts were used: a Producer, a Consumer, and a Monitor, as seen in Figure 1. The Producer generates data that is transmitted over the network to the Consumer. The Consumer receives and logs the data. The Monitor contains the Central Addressing and Lookup services, which the Producer and Consumer discover by sending network probes. In addition, each node runs the SPA Local Manager and SpW Manager processes. Table II summarizes the SSM services running on each of the external hosts.

TABLE II. SERVICES ON EXTERNAL HOSTS

Producer	Consumer	Monitor
Producer Application SPA Local Manager SPA Spw Manager	Consumer Application SPA Local Manager SPA Spw Manager	SPA Local Manager SPA SpW Manager Central Addressing Service SPA Lookup Service

In order to integrate the external hosts with the OPNET simulation, an additional interface was added to the SPA SpW Manager service. Typically, the service is intended to use interfaces to physical devices such as SpW or Universal Serial Bus (USB). In our co-simulation, we replace the SpW interface with a *SpW-Sim* interface. Rather than connecting to a physical SpW network, the SpW-Sim connects to the OPNET simulation model. It does this by wrapping the SpW message in a Transmission Control Protocol/Internet Protocol (TCP/IP) packet and sending that packet over an Ethernet network to the OPNET computer. This TCP/IP connection is part of the co-simulation controller and is transparent to both the SSM processes and to the simulation model.

The OPNET/SSM Co-Simulation Controller bridges the external hosts to the OPNET simulation. Its external interface is a TCP/IP Socket Server that connects to the SpW-Sim interfaces on external hosts (see Figure 2). When a TCP/IP message is received, the server extracts the SpW message and stores it in a buffer, where it waits to be forwarded to the appropriate node within the OPNET simulation.

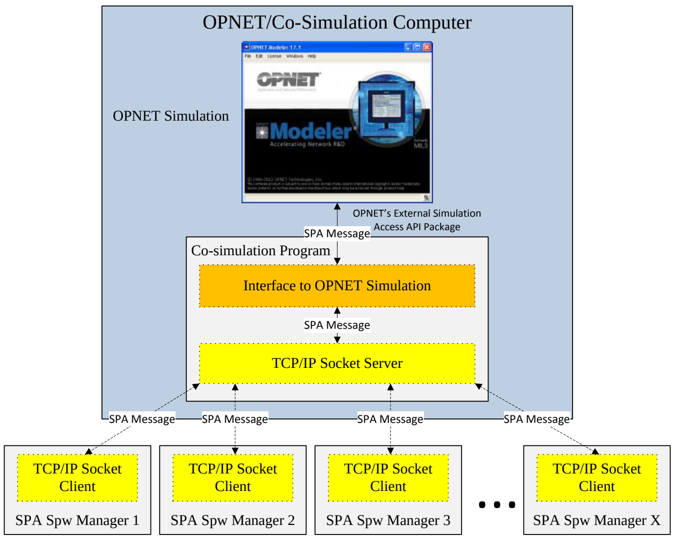


Fig. 2. OPNET TCP/IP Socket Interface

The Controller is also responsible for driving the OPNET simulation forward. Using the OPNET External Simulation Access (ESA) API package, the Controller specifies a simulation time and hands over the thread of execution to the OPNET simulation. At this time, the buffered messages received by the Socket Server are forwarded to their corresponding simulation nodes where they become traffic generators for the simulation. As seen in Figure 3, the thread of execution is not returned back to the Controller until the simulation has advanced to the specified simulation time.

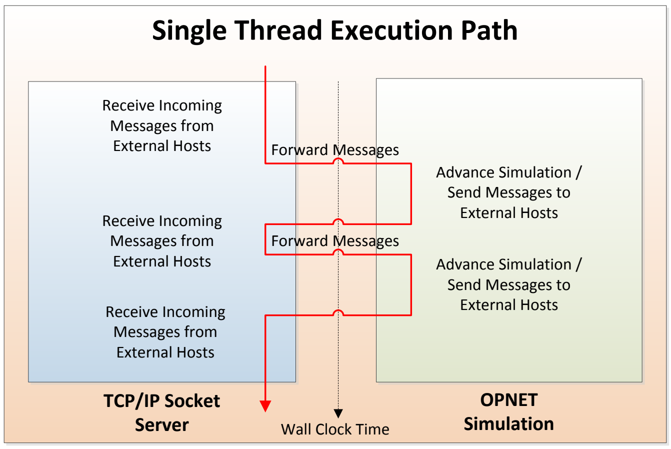

Single Thread Execution Path
Receive Incoming Messages from External Hosts
Forward Messages
Advance Simulation / Send Messages to External Hosts
Receive Incoming Messages from External Hosts
Forward Messages
Advance Simulation / Send Messages to External Hosts
Receive Incoming Messages from External Hosts
TCP/IP Socket Server
Wall Clock Time
OPNET Simulation

Fig. 3. Co-Simulation Controller Execution Path

Data flow in the opposite direction is achieved by a callback function implemented in the Controller program. The callback function is registered using the ESA APIs when the Controller program is initialized. As SPA messages traverse the simulated OPNET network, eventually they reach the destination node. When a simulation node receives a message, a callback function is triggered. There, the message is forwarded to the corresponding external host using the established TCP/IP socket connection.

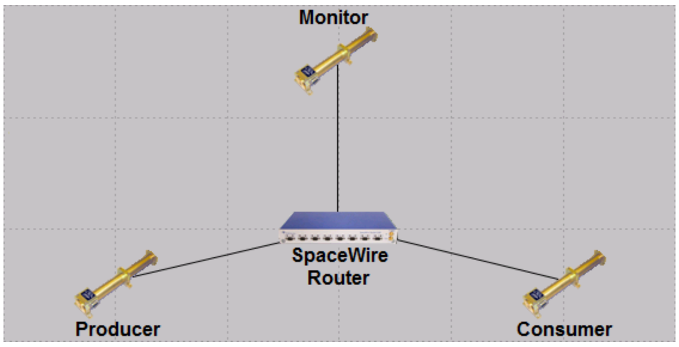


Fig. 4. OPNET Simulated Network Topology

The OPNET simulated network topology, as seen in Figure 4, was configured to have the same number of simulated nodes as the number of external hosts in order to have one-to-one correspondence. The nodes are interconnected using a simulated SpW router. Two models were created for the OPNET simulation: a SpW Node model and a SpW Router model. The models were developed as means to feed traffic generated from external SSM hosts into the OPNET simulation. The SpW Node model uses OPNET's External System (Esys) package to establish a link to the Co-Simulation Controller. The Esys package is similar to the ESA API described earlier in that they both allow communication between an OPNET simulation and an external program. However, the Esys package is used in OPNET models within the simulation while the ESA package is used in external programs. Together, the Esys and ESA packages allow the SpW Node model to establish a bidirectional link with an external SSM host through the Co-Simulation Controller. The model receives SpW messages from an external host and forwards the messages into the simulated network. When messages are received from within the simulation, the model forwards them to the external hosts. The SpaceWire Router model simply uses SpW path addressing to forward messages out the appropriate port.

B. SSM Time Implementation Description

The SSM uses two classes to implement time-related calls: SpaTimers and SpaTimingUtils. The SpaTimers class is based on an operating system timer. For Linux, these are Portable Operating System Interface (POSIX) timer calls. When the operating system (OS) timer expires, the SpaTimers class uses a dispatcher method to invoke a specified handler function. As such, the SpaTimers class allows a given handler function to run at a specified frequency, and, therefore, implements a periodic callback functionality. In this way, the SSM periodically publishes messages to subscribers on the network.

The SpaTimingUtils class manages a high resolution time known as SPA Time that can be synchronized with an external time source such as the global positioning system (GPS). Upon receiving an update, the time is stored in shared memory. Between updates, the elapsed time since the last update is added to the stored time to create a SPA Time with up to one nanosecond resolution (see Figure 5).

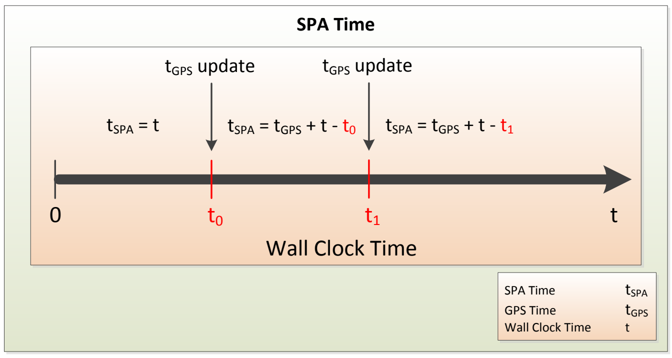


Fig. 5. SPA Time Synchronization and Calculation

C. Solution to Time Synchronization Problem

The Co-simulation Controller program is responsible for advancing the OPNET simulation, and the interface between it and the OPNET process is provided by the OPNET ESA API library. The co-simulation controller calls the ESA function Esa_Execute_Until() with an absolute simulation time. This causes the co-simulation process to block and the OPNET simulation process to execute until certain conditions are met (see OPNET ESA library documentation for details). At that point, the OPNET simulation pauses and hands control back to the Co-Simulation Controller process.

Esa_Execute_Until() returns the current simulation time to the Controller. The Controller then sends a User Datagram Protocol (UDP) multicast broadcast with the current simulation time. This functionality in the Co-Simulation Controller is known as the SimTime Server. Figure 6 illustrates the interface between the OPNET simulation process and the SimTime Server in the Co-Simulation Controller. Table III describes fields and format for the SimTime multicast message.

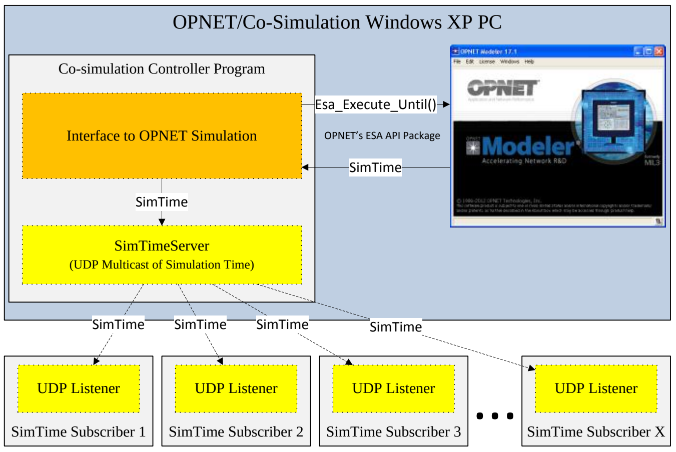


Fig. 6. Simulation Time Distribution Interface

TABLE III. SIM TIME MESSAGE FIELDS AND FORMAT

Offset	0	1	2	3	4	5	6	7
0x0000	**Sequence Number unsigned short – 2 octets**							
0x0008								
0x0010	**Reserved N/A – 2 octets**							
0x0018								
0x0020	**Sim Time Seconds unsigned int – 4 octets**							
0x0028								
0x0030								
0x0038								
0x0040	**Sim Time Microseconds unsigned int – 4 octets**							
0x0048								
0x0050								
0x0058								

Our initial approach to the time synchronization problem was to treat simulation time as the external time source, using the existing SSM "SPA Time" utility. This would solve the problem for cases where the external host uses simulation time, such as timeouts for network discovery. However, SPA time is also used for local events where simulation time is not appropriate, such as timeouts on acquiring a mutex for local shared memory. Consider that a mutex is used to protect the SPA time in shared memory – it makes little sense to use the shared memory time as the timeout for acquiring that shared memory.

We were faced with two choices; either modify the timers in the OS kernel to use simulation time, or implement our own versions of these timers. Because the SSM is designed to execute on Windows, Linux, and VMWorks, modifying the kernel is not practical. We therefore decided to implement simulation time replacements for the OS timers. The remainder of this section describes our implementation.

In the existing SSM code for Linux, SPA timers are implemented using POSIX timers based on the wall clock. When a timer expires, the OS kernel runs the SpaTimer dispatch function, which in turn invokes the callback function for that timer ID. This callback function executes on a newly created thread. Since POSIX timers cannot be converted to simulation time without modifying the OS kernel, an application-level timer mechanism was implemented instead. This mechanism consists of three components, as shown in Figure 7.

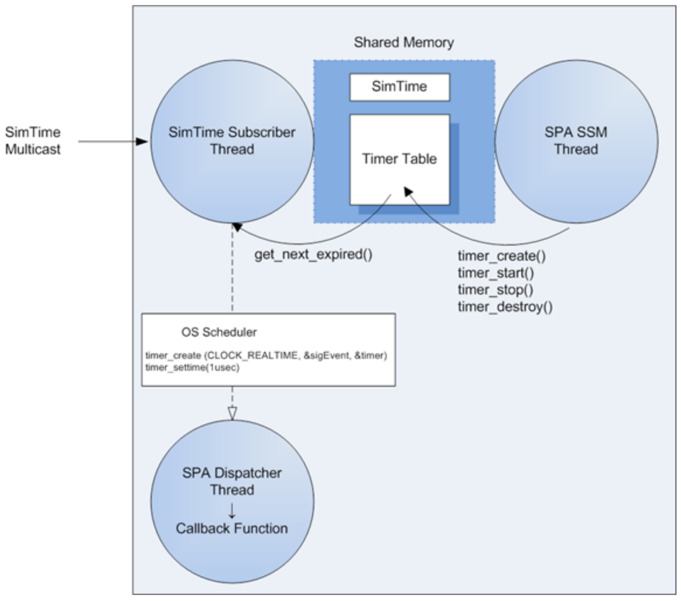


Fig. 7. Simulation Time SSM Design

- Shared Memory
  The Shared Memory contains the local simulation time variable and a table of timers. This memory is shared by the SimTime Subscriber thread and SSM thread within a single process. The fields included in the timer table are shown in Table IV.

TABLE IV. SIM TIMER TABLE FIELDS

Field	Description
State	Possible states include: EMPTY STOPPED RUNNING
Expiration Time	Timer expiration time (timespec)
Interval Time	Timer interval time (timespec)
Timer Args	Timer arguments which include a pointer to the callback handler function.

- SimTime Subscriber
  The subscriber runs as a thread within each SSM process. It is created when the process starts and is deleted when the process exits. The subscriber blocks waiting for a SimTime multicast. When a message arrives, the subscriber updates the local simulation time, checks the timer table, and invokes callback functions for expired timers. The dispatch functionality of the SpaTimer class is moved to this thread.
- Extensions to the SpaTimer Class
  The SpaTimer class contains methods for managing timers, including create, start, stop, and destroy. For system (wall clock) timers, these methods use the corresponding POSIX timer functions. For simulation timers, these methods modify the timer table in shared memory as shown in Table V.

TABLE V. SPATIMER MODIFIED METHODS

Method	Description
timer_create ( )	Adds a new timer to the timer table. The function returns a value of -1 if the table is full. Otherwise, it returns 0, sets the state to STOPPED, and stores the callback function pointer. The timer is stored in the first empty slot in the table, which is found by a linear search from index 0 to last_entry. If no empty slot is found in this range, last_entry is incremented and the timer is stored at this location. An empty slot is denoted by state = EMPTY.
timer_start ( )	Sets timer state to RUNNING and updates the timer table entry expiration and interval times.
timer_stop ( )	Sets the timer state to STOPPED.
timer_stop ( )	Sets the timer state to EMPTY, effectively removing it from the table.

In the SpaTimingUtils class, the SPA Time functionality was modified to use simulation time. Instead of adding elapsed wall clock time to SPA Time, the elapsed simulation time acquired by the SimTime Subscriber is used.

IV. RESULTS

Upon analyzing the experimental results for our studies, we were able to perform the following verifications.

The OPNET Co-Simulation with SSM was intentionally slowed down to run at speeds well below real-time. This was done to verify the SSM interface to the OPNET simulation, along with the changes to utilize simulation time. The co-simulation ran successfully with the described Producer, Consumer, and Monitor setup. Each of the external hosts was able to perform network discovery through the OPNET simulation and establish connections among each other. The Consumer was able to successfully acquire messages and utilize the data sent by the Producer. The overall functionality of the SSM with the OPNET Co-Simulation was validated against the same SSM setup over a physical SpW interface which included three SpW cards and a SpW router. With the exception of the slowed time, the two systems displayed nearly identical characteristics.

Secondly, we wanted to verify that the simulation time increased in a sequential manner as traffic flowed through the co-simulation system. Messages were logged as they went through every step in the system along with the simulation time at the time of arrival/departure. Figure 8 shows an example of the flow of traffic and the points in the co-simulation at which messages were logged.

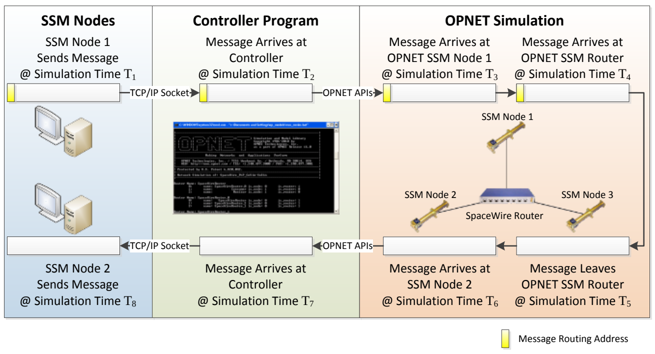

SSM Nodes
SSM Node 1 Sends Message @ Simulation Time T_1
Controller Program
Message Arrives at Controller @ Simulation Time T_2
OPNET Simulation
Message Arrives at OPNET SSM Node 1 @ Simulation Time T_3
Message Arrives at OPNET SSM Router @ Simulation Time T_4
TCP/IP Socket
OPNET APIs
SSM Node 1
SSM Node 2
SSM Node 3
SpaceWire Router
TCP/IP Socket
OPNET APIs
SSM Node 2 Sends Message @ Simulation Time T_8
Message Arrives at Controller @ Simulation Time T_7
Message Arrives at SSM Node 2 @ Simulation Time T_6
Message Leaves OPNET SSM Router @ Simulation Time T_5
Message Routing Address

Fig. 8. Expected Simulation Time and SPA Message Flow

We were able to verify that simulation time increased as SPA messages traversed the SSM Co-Simulation. Table VI shows the sequential simulation time at time of arrival for different points in the co-simulation for one SPA message. No simulation time discrepancies were found in the experimental results.

TABLE VI. SINGLE MESSAGE SIMULATION TIME LOG

Message Location	SimTime (s)	Delta SimTime (s)
Monitor Sent	2066.715	-
Controller Received	2066.715	0
OPNET Monitor Received	2066.715	0
OPNET SpW Router Received	2066.715001	0.000001
OPNET SpW Router Sent	2066.72	0.004999
OPNET Consumer Received	2066.720002	0.000002
Controller Sent	2066.720002	0
Consumer Received	2066.720002	0

V. CONCLUSIONS AND FUTURE WORK

A hybrid simulation that integrates a traditional discrete event simulation with a software in the loop simulation for SpW networks and the SSM SPA middleware has been successfully demonstrated. The work has shown that through small modificatiosn to the SSM, time synchronization to the simulation time can be achieved. Simulations can be achieved that are suitable to investigate networks of typical complexity for SpW networks.

Several areas remain to be investigated. First, the Co-Simulation must be integrated with the SpW modules created by OPNET Technologies, Inc. to ensure high fidelity of SpW within the OPNET simulation. As described earlier, the OPNET models created for this experiment were developed as a means to demonstrate a proof-of-concept for the OPNET Co-Simulation with SSM but do not accurately represent the SpW standard. Second, a validation of the simulation against a physical network implementation is needed to confirm fidelity of the simulation with reality. SwRI and AFRL plan to work together to build a configuration AFRL can realize in the laboratory to demonstrate strong coherence between the simulation and a real world implementation. Third, several network topologies representing realistic potential spacecraft SpW networks need to be generated to demonstrate that the process can be scaled. Last, real networks experience data loss, and the OPNET Modeler SpW module includes facilities to generate a variety of faults in the communication process. Experimenting with fault injection in the network simulation can be used to verify acceptable SPA behavior, identify weaknesses, and test modifications that increase SPA robustness.

VI. ACKNOWLEDGEMENTS

This was partially supported under AFRL contract number P010213604. We want to thank Dr. James Lyke of AFRL and Robert Vick of SAIC for their support. We also want to acknowledge the efforts of SDL on the SSM and the OPNET Technologies team.

VII. REFERENCES

[1] Lyke, James C., "U.S. Air Force's Plug-and-Play Satellites," IEEE Spectrum Inside Technology, August 2012. Reprint from Plug-and-Play Satellites.

[2] "Space Plug-and-Play Architecture Standards Development Guide" (draft), AIAA, August 2011.

[3] "Space Plug-and-Play Architecture Standard Networking" (Draft), AIAA.

[4] "Space Engineering SpaceWire – Links, Nodes, Routers and Networks," ECSS, July 2008.

[5] SDL's Modular Software Redmine Website, pnpsoftware.sdl.usu.edu/redmine/

[6] OPNET Website, www.opnet.com

[7] Lu, Zheng, H. Yang, "Unlocking the Power of OPNET Modeler," Cambridge, 2012.

SpaceWire Time Code Latency and Jitter

Session SpaceWire networks and protocols, Long Paper

Martin Suess
European Space Agency / ESTEC
Noordwijk, the Netherlands
martin.suess@esa.int

Felix Siegle
University of Leicester
Leicester LE1 7RH, UK)
fs131@leicester.ac.uk

***Abstract*—SpaceWire Time-Codes are intended to be used for time synchronization on board of spacecraft. So far not many reports are available on the synchronization accuracy that can be actually achieved with this mechanism. This paper describes a series of Time-Code latency measurements that have been made using a SpaceWire network built up from SpW-10X routers (AT7910E). During these measurements not only the link speed but also the length of the path through the network and data traffic load is varied. The obtained time delay data are statistically analyzed in terms of mean Time-Code latency and jitter. A simple linear model is fitted to the data to allow the prediction of the mean latency for data rates and network sizes which have been not directly covered by the measurements. The size of the observed Time-Code jitter is reported as well.**

***Index Terms*—SpaceWire, network, time-code, jitter, latency, measurement, distributed interrupt, SpW-10X, AT7910E.**

INTRODUCTION

Space systems frequently have the need to synchronize time within the distributed avionics system or between the spacecraft platform and the instruments. Today a separate time distribution network is normally used for distributing a pulse per second time synchronization signal. For platform applications the required time synchronization precision is typically ranges from 10µsec to 1msec. Within some specific instruments the required time synchronization precision can be significantly higher from several 10nsec to several 100nsec [1]. With the time-codes the SpaceWire standard offers a time synchronization mechanism which could be used as an alternative. This would allow to remove the need for the separated dedicated time synchronization network.

The time-codes are specified and the mechanism is described in the SpaceWire standard [2] but there is not much information available about which time synchronization performance can be reached in practice. The expected performance is not only dependent on the actual implementation of the SpaceWire routing switches but also on a number of other parameters like the link speed and the network topology, i.e. how many routing switches have to be passed by the Time-Code on its way from the source to the destination. The same is the case for the mean time-code latency. While the latency may not be so important for clock synchronization as it results only in a fixed time off-set its knowledge is very important for the safe operation of the distributed interrupt mechanism which is going to be introduced in the next revision of the SpaceWire standard.

MEASUREMENT TEST SETUP

In order to better assess the real achievable time synchronization performance of SpaceWire Time-Codes and its dependence on various system parameters a series of measurements have been conducted in a network based on the SpW-10X SpaceWire router (AT7910E). For the time measurements a high precision Time Interval Counter SR620 from Stanford Research Systems was used with a relative time measurement error that is specified to be less than 100psec.

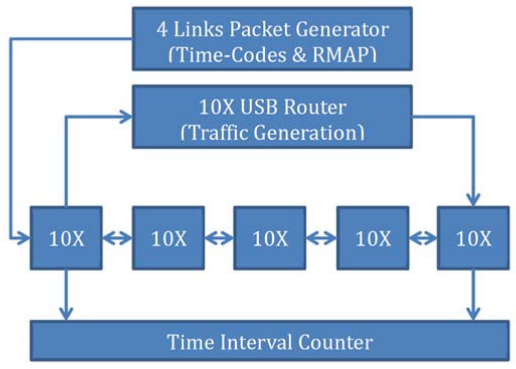


Fig. 1: Time-Code latency measurement setup with 5 SpW-10X routers

A block diagram of this measurement set-up is depicted in Fig. 1. The SpaceWire network consists of a chain of 5 SpW-10X routers, one packet generator from 4Links is used for the injection of Time-Codes and a SpW-10X USB Router with deactivated Time-Code forwarding is used for the generation of auxiliary traffic in the network. The SR620 Time Interval Counter measures with high accuracy the time interval between the start and the stop pulses at its two trigger inputs. The start pulse is taken from the TICK_OUT pin of time-code interface of first router in the chain. The stop pulse is taken from the TICK_OUT pin of one of the later routers in the chain dependent on how many hops in the network shall be taken into account for the measurement. Each of the SpW-10X routers is supplied with its own, independent 30 MHz system clock. The routers in the chain are interconnected with 0.5m long SpaceWire cables. This arrangement has been chosen to measure the Time-Code latency due to the routers and the connecting links and to avoid any latency variation due to the Time-Code injection or generation.

A picture of the actual Time-Code latency measurement setup is shown in Fig. 2. It was used to measure the following combination of parametric test cases:

- The number links between the routers in the chain is varied between 1 to 4,
- The data rate of all links is set to one of the following 2.73, 3, 30, 60, 120 and 200 Mbps,
- The influence of data traffic on the links is evaluated by injecting no traffic or 100% in the direction of the time code propagation and 100% traffic in the opposite direction.

In each of these 72 combinations of conditions the latency between the TICK_OUT signal from the first router to the TICK_OUT signal of the last router in the chain was measured 15000 times in order to get a good statistical basis.

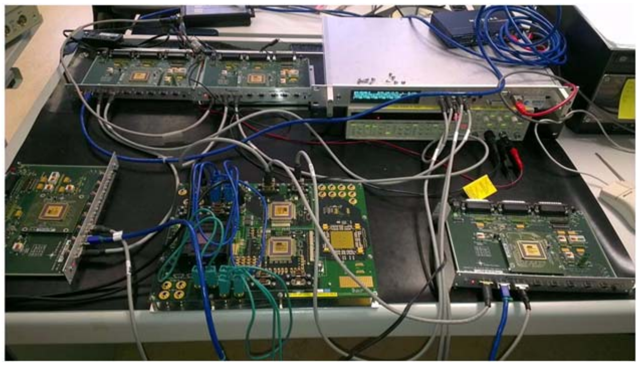


Fig. 2: Picture of actual measurement set-up

CAUSES FOR TIME CODE JITTER

There are two fundamental mechanisms that are the cause for Time-Code jitter. The Time-Code jitter is defined here as the maximum time difference observed between the 15000 individual Time-Code latencies.

a) The first cause of jitter is described in NOTE 2 of clause 8.12.2 p. in [2]. Before the a Time-Code can be transmitted on a link interface the transmitter has to finish the current data character, control character or control code. If no other traffic is on the network NULL characters with a length of 8 bits are sent over the link. In this case the jitter at the link interface is in between 0 and 8 transmit bit periods.

b) The second origin of jitter is the fact that the Time-Code signal has to cross the boundaries of incoherent clock domains inside the router. The signal arriving at the receive port of the router contains the clock from the transmitting side which is recovered from the data/strobe encoding used to sample the signal and to decode the Time-Code. The Time-Code arrival needs then to be synchronised into the system clock domain of the router as it is also the case for an external TICK_IN signal. Similar the Time-Code signal needs to be synchronised when crossing from the system to the transmit clock domain. In the presented measurement setup the transmit clock is derived from the same oscillator as the system clock. The presence of synchronisation jitter depends therefor on the ratio between the transmit and the system clock.

For low link data rates the overall jitter is dominated by the first cause while for high data rates the second cause for jitter can reach a similar order of magnitude as the first one. The methods to improve the Time-Code jitter published in [1] and [3] mainly aim to reduce the jitter contribution described in a).

The clock domain transitions on the path of the of the Time-Code signal in the SpW-10X for the situation of idle links is shown in Fig. 3.

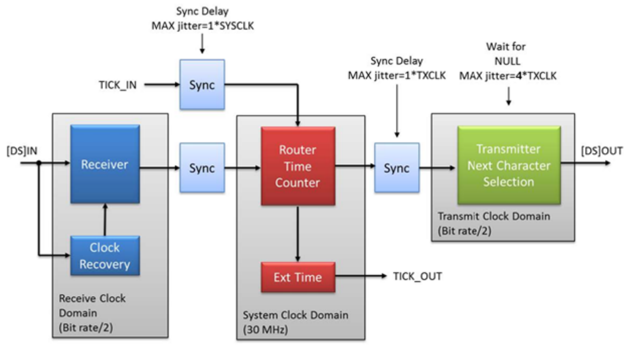


Fig. 3: Clock domain transitions on the path of the Time-Code signal in the SpW-10X router

LATENCY AND JITTER MEASUREMENTS

The first set of measurements shown in Fig. 4 has been acquired with a link data rate of 3 Mbps and no additional traffic being injected in the network. The plots a) to d) show the histograms of the measured Time-Code latency for 1 to 4 links respectively. As each histogram contains 15000 statistically independent latency measurements it can be interpreted as approximation for the random distribution of the Time-Code latency.

For this test case the mean latency is 11.07µsec per link and the worst case jitter increases by 2.665µsec per link. This worst case jitter corresponds very well to the 8 transmit bit periods of jitter introduced by the link transmit interface at a data rate of 3Mbps as described in 0 a). As expected this case results in a uniform random distribution of the Time-Code latency as shown in Fig. 4a).

All the random distributions shown appear to be very well symmetric and the mean and the median value agree with a remaining difference of less than 0.05%.

The development of the Time-Codes latency random distribution when passing over several links can serve as a demonstration of the Central Limit Theorem [4] as known in statistics. The jitter introduced when a Time-Code passes over a single link results in a uniformly distributed Time-Code latency as shown in Fig. 4a). When passing over a second link which adds a second statistically independent uniformly distributed jitter causes the Time-Code latency random distribution to become triangular as shown in Fig. 4b). According to the Central Limit Theorem sum of a sufficiently large number of independent random variables of similar shape will result in a normal (or Gaussian) random distribution which is nicely visible in Fig. 4c) and d).

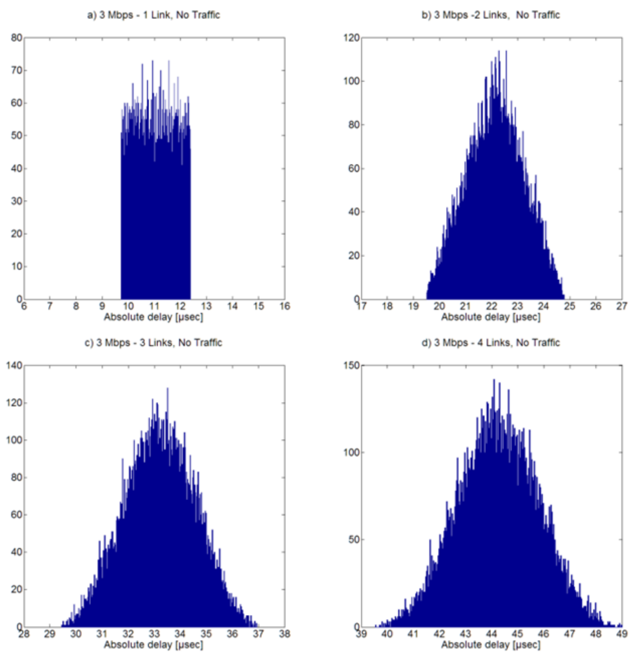


Fig. 4: Histograms of the measured Time-Code latency for 1 to 4 links at a data rate of 3Mbps with no additional traffic on the network.

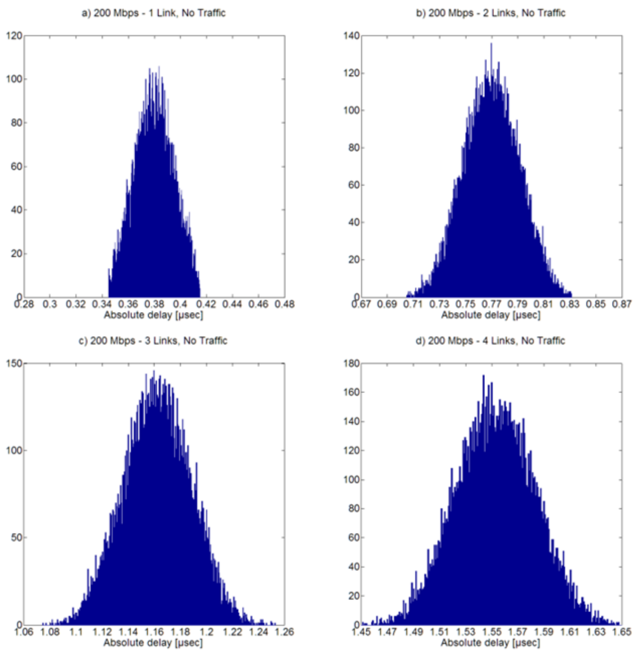


Fig. 5: Histograms of the measured Time-Code latency for 1 to 4 links at a data rate of 200Mbps with no additional traffic on the network

The second set of measurements shown in Fig. 5 has been acquired with a link data rate of 200 Mbps and no additional traffic being injected in the network.

For this test case the mean latency is 0.385 µsec per link and the worst case jitter increases by 70.2 nsec per link. As expected the significant increase of the link data rate has drastically reduced the Time-Code latency and the jitter. The worst case jitter introduced per link corresponds to about 14 transmit bit periods which shows that there must be an additional significant cause of jitter. As described in 0 b) this additional jitter is due to the clock domain crossings of the Time-Code signal on its path within the router. This additional statistically independent random variable makes the latency distribution to look somewhat triangular already for a single link case as shown in Fig. 5a). When passing through additional routers and over additional links the Time-Code latency random distribution becomes a more and more Gaussian shape as visible in Fig. 5b) to d).

Model for the mean Time-Code latency

The mean Time-Code latency as measured for the different link speeds over a distance of 1 to 4 links is shown in Fig. 6. It shows nicely a linear increase of the mean Time-Code latency with link distance.

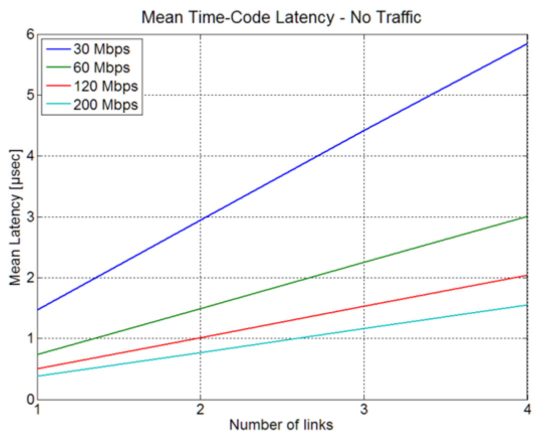


Fig. 6: Mean Time-Code Latency for different link speeds and no traffic

Based on these data a generalised parametric model for the mean Time-Code latency as a function of link speed and the number of links is derived in the following.

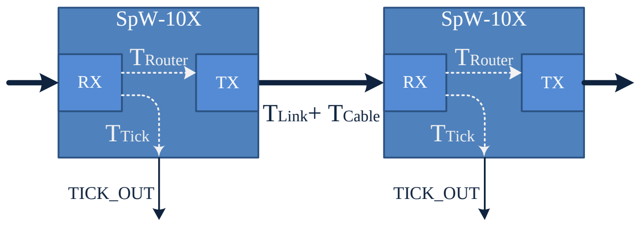


Fig. 7 Time-Code latency measurement setup with 5 SpW-10X routers

As a first step all the elements contributing to the measured Time-Code latency are analysed with the help of Fig. 7.

The boxes marked with RX and TX belong to receiver and the transmitter clock domain while the rest of the router belongs to the system clock domain.

The time delay T_m measured with the time counter is the time between the TICK_OUT signal of the first and the N^{th} router in the path of the Time-Code. There is a time delay T_{Tick} between the reception of the Time-Code in one of the ports of the router and the raising of the TICK_OUT signal. The time delay inside the router T_{Router} between the reception of the Time-Code in one of its ports and its retransmission through the other ports which can be expressed in terms of system clock periods T_{sys}. The time delay T_{Link} over the link between the transmission of the Time-Code at one router and its reception at the next router is assumed to be proportional to the link bit period T_{bit} = 1/Link_Rate. In addition to this there is the cable delay T_{Cable} which is proportional to the cable length and assumed to be 1/(0.59·c) or 5.6 nsec/m for twisted pair cables [5]. For the 0.5m long cables used in between the routers the parameter T_{Cable} is set to 2.8 nsec. In the mean all the time delays listed above are assumed to be the same for all routers and all links.

The measured time delay can be therefore modeled as follows.

$$\begin{aligned} T_m(N, T_{bit}) &= \text{TICK\_OUT}_N - \text{TICK\_OUT}_1 \qquad \text{Eq. 1} \\ &= N\cdot T_{Router} + N\cdot T_{Link} + N\cdot T_{Cable} + T_{Tick} - T_{Tick} \\ &= N\cdot A\cdot T_{Sys} + N\cdot B\cdot T_{bit} + N\cdot T_{Cable} \end{aligned}$$

The resulting equation which is a function of the number of links N and the transmit bit period has two unknowns A and B. Only two independent measurements with different link rates could be sufficient to determine the unknowns. In order to be able to take benefit of all 24 measurements available the over determined system of equation is written in matrix form. The Moore Penrose pseudo inverse of the matrix ***M*** can be calculated and used to solve the equation. The use of the Moore Penrose pseudo inverse minimizes the global error between the model and the measured data in the least square sense.

$$\vec{T} - T_c\vec{N} = \boldsymbol{M} \cdot \begin{bmatrix} A \\ B \end{bmatrix} \qquad \text{Eq. 2}$$

$$\begin{bmatrix} T_{m\,1} \\ \vdots \\ T_{m\,24} \end{bmatrix} - \begin{bmatrix} T_c N_1 \\ \vdots \\ T_c N_{24} \end{bmatrix} = \begin{bmatrix} N_1 T_{Sys} & N_1 T_{bit} \\ \vdots & \vdots \\ N_{24} T_{Sys} & N_{24} T_{bit} \end{bmatrix} \cdot \begin{bmatrix} A \\ B \end{bmatrix}$$

$$\begin{bmatrix} A \\ B \end{bmatrix} = pinv(\boldsymbol{M}) \left(\vec{T} - T_c\vec{N}\right) \qquad \text{Eq. 3}$$

The time delay inside the router T_{Router} results to 260.2 nsec or A=7.806 system clock periods. The mean delay introduced by a link is calculated to be B=32.42 transmit bit periods. By inserting the derived values for A and B in Eq. 1 the expected mean time delay can be calculated for any N number of links and any link data rate $1/T_{Link}$. This very simple linear model fits well to the measurements in particular for low data rates and has still less than 10% error at the maximum data rate.

Influence of Data Traffic on Time-Code Latency

The measurements presented so far have all been obtained with no auxiliary data traffic in the network. If no other data traffic is waiting for transmission SpaceWire inserts NULL control codes (ESC + FCT) of 8 bits length to keep the link active. Data is transmitted with data characters of 10 bits length and one FCT (flow control token) of 4 bits length is transmitted for every 4 data characters received. These two kind of characters together with the NULL control code are normally the most frequent characters present in a SpaceWire network. Due to their different length they are expected according to 0 a) to have an influence on the mean Time-Code latency and the jitter introduced by each SpaceWire link. In order to quantify this influence the Time-Code latency has been measured with two additional traffic load cases where in the first case the Time-Code transmitting side is sending data at maximum rate and in the second it is receiving data at maximum rate.

The changes in latency and jitter are nicely visible in the measured data. In the first case where the Time-Code transmitting side is sending at 3 Mbps data characters of 10 bit length the mean latency increases by 338.5 nsec which corresponds to an increase by 1.015 transmit bit periods. For the same data rate the maximum observed jitter over a single link increase by 666.9 nsec corresponding to 2.001 transmit bit periods. The same increase of jitter is also found for the other data rates which were measured.

In the second case the Time-Code transmitting side is transmitting an FCT control code for every four data characters received. This means at maximum data rate 9 NULL and 2 FCT control codes are transmitted for every 8 data characters received. Considering the length of the two characters the probability that the transmission of a Time-Code has to wait for the completion of a FCT instead of a NULL control code is only 1/9. If the received data rate is lower than 100% this probability reduces proportionally. Accordingly the changes in latency and jitter are much less pronounced. At a data rate of 3 Mbps the mean latency decreases by 35.85 nsec corresponding to a reduction by 0.1076 transmit bit periods. For the same data rate the maximum jitter over a single link remains the same within the measurement accuracy.

Also for the two test cases with auxiliary data traffic on the network the linear model for the mean Time-Code latency given in Eq. 1 can be calculated.

In the first case where the Time-Code transmitting side is also sending data at maximum data rate the modeled time delay inside the router T_{Router} results to 262.6 nsec or A=7.879 system clock periods. The mean delay introduced by a link is calculated to be B=33.04 transmit bit periods. This is the worst case traffic load situation which causes the maximum mean Time-Code latency.

In the second case where the Time-Code transmitting side is receiving data at maximum data rate the modeled time delay inside the router T_{Router} results to 261.7 nsec or A=7.852 system clock periods. The mean delay introduced by a link is calculated to be B=32.33 transmit bit periods.

STANDARD DEVIATION OF THE TIME-CODE JITTER

The following maximum Time-Code jitter has been measured for a single link at different data rates when no auxiliary data traffic is present.

MAXIMUM TIME-CODE JITTER WITH NO TRAFFIC

Link data rate	Maximum jitter in nsec	Maximum jitter in transmit bit periods
3 Mbps	2665	7.996
30 Mbps	266.7	8.002
60 Mbps	133.5	8.015
120 Mbps	83.41	10.01
200 Mbps	70.24	14.05

For the link data rates 3, 30 and 60 Mbps the maximum jitter corresponds nicely to the 8 transmit bit periods expected from the length of the NULL characters present on the link. For the link data rates 120 and 200 Mbps this jitter is increased through an additional jitter source internal to the router.

MAXIMUM TIME-CODE JITTER WHEN TRANSMITTING DATA AT MAXIMUM DATA RATE

Link data rate	Maximum jitter in nsec	Maximum jitter in transmit bit periods
3 Mbps	3332	9.997
30 Mbps	333.3	10.00
60 Mbps	166.9	10.02
120 Mbps	100.1	12.01
200 Mbps	80.16	16.03

The maximum jitter in the maximum transmit data rate case is consistent with the jitter values provided in 0but extended by two transmit bit periods as expected from the two bit longer data characters now being present on the network. This worst case jitter data apply also when the data is transmitted at lower than maximum data rate.

MAXIMUM TIME-CODE JITTER WHEN RECEIVING DATA AT MAXIMUM DATA RATE

Link data rate	Maximum jitter in nsec	Maximum jitter in transmit bit periods
3 Mbps	2666	7.999
30 Mbps	266.8	8.005
60 Mbps	133.5	8.012
120 Mbps	83.73	10.05
200 Mbps	69.98	14.00

The maximum jitter in the case where auxiliary data is received at maximum rate matches very well the data obtained for the no traffic case. This is not surprising as the presence of the shorter FCT characters does not influence the worst case situation.

In order to calculate of the worst case Time-Code jitter in the network these values provided in TABLE I to III have to be multiplied with the number of links to the destination.

For some applications it is not the worst case but the standard deviation of the jitter is needed to assess the impact. For this case the standard deviation of the time code latency is provide in Fig. 8 for the no traffic case and in Fig. 9 for the maximum transmit data rate case for a network distance of 1 to 4 links.

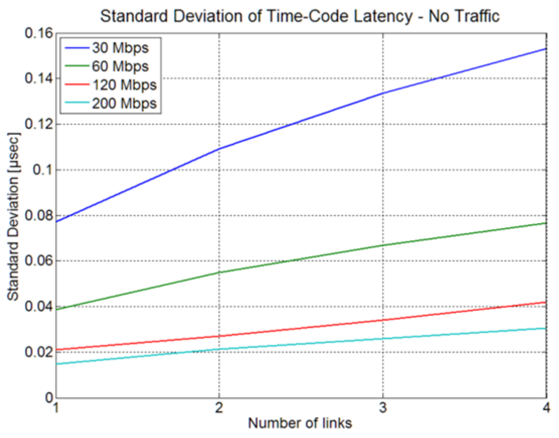


Fig. 8: Standard Deviation of Time-Code Latency for different link speeds in the no traffic case

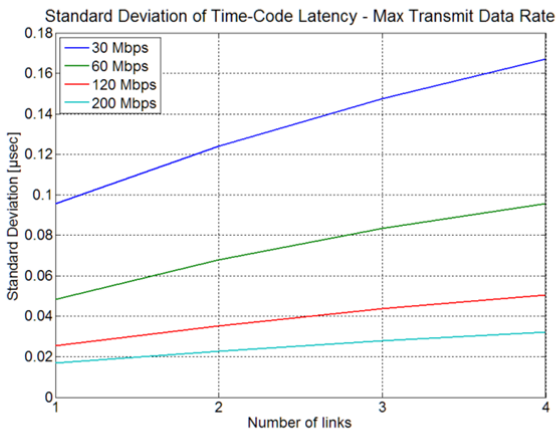


Fig. 9: Standard Deviation of Time-Code Latency for different link speeds in the maximum transmit data rate case

CONCLUSIONS

In this paper a series of Time-Code latency measurements for a set of different network parameter settings has been reported. The statistical behaviour of the measurements has been analysed in terms of mean value and jitter. This gives an indication on the synchronisation accuracy that can be reached with the SpaceWire time code mechanism. A simple linear model for the mean Time-Code latency dependent on the number of links and the data rate has been fitted to be able to predict the mean latency also for cases not directly covered by the measurements. This information may give a first indication about the guard times and time-out values to be used for the safe operation of the distributed interrupt mechanism which is going to be introduced in the next revision of the SpaceWire standard.

Acknowledgment

The authors want to thank Chris McClements from the University of Dundee for the discussion of the clock domain boundaries and synchronization inside the SpW-10X router and for providing Fig. 3.

References

[1] F. Pinsard and C. Cara "High resolution time synchronization over SpaceWire links", Aerospace Conference 2008, IEEEAC paper#1158, 10.1109/AERO.2008.4526462

[2] SpaceWire - Links, nodes, routers and networks - ECSS-E-ST-50-12C, 31 July 2008

[3] Barry M Cook, "Reducing Time Code Jitter on SpaceWire", http://www.4links.co.uk/bibliography/Reducing-Time-Code-Jitter-on-SpaceWire.pdf

[4] John A. Rice, Mathematical Statistics and Data Analysis (Second ed.), Duxbury Press, 1995

[5] http://stason.org/TULARC/networking/lans-ethernet/3-11-What-is-propagation-delay-Ethernet-Physical-Layer.html

SpaceWire – Time Distribution Protocol

SpaceWire Networks & Protocols
Long Paper

Sandi Habinc, Anandhavel Sakthivel
Aeroflex Gaisler AB
Gothenburg, Sweden
info@gaisler.com

Martin Suess
European Space Agency
Noordwijk, The Netherlands

***Abstract*— Aeroflex Gaisler has developed, under European Space Agency (ESA) contract 4000104519, a draft ECSS protocol for the transmission and synchronization of CCSDS Unsegmented Code (CUC) time in SpaceWire networks. The working name of the protocol is "Time Distribution Protocol“.**

***Index Terms*—SpaceWire, Networking**

I. Introduction

The objective of the referenced “High Accuracy Time Synchronization over SpaceWire Networks” ESA activity is to:

- Establish a time message distribution mechanism over SpaceWire
- Establish an offset correction mechanism between local times which is correcting for the time distribution latency in SpaceWire networks
- Establish a method for clock synchronization by correcting the drift between clocks as well as the jitter experienced in SpaceWire networks

The first point above is fully covered in the Time Distribution Protocol, support is provided in the draft protocol. The implementation of the second point and the third point is actually outside of the draft protocol since it does not affect the protocol itself (see discussion further down).

The target is for example SpaceWire networks for critical on-board control applications where current on-board buses such as Mil-Std-1553 and CAN 2.0B can be replaced. For this type of applications the time accuracy is important to allow implementation of isochronous communication over the inherently asynchronous SpaceWire network.

Also SpaceWire networks for science payloads where time synchronization is an important factor are targeted. An example could be multiple distributed sensors in an antenna that communicate via SpaceWire and need be synchronized for coherent measurements, or when two instruments exchange data to correlate their results.

II. Draft Protocol

The Time Distribution Protocol provides the capability to transfer CCSDS Time Codes (i.e. time message) between onboard users of a SpaceWire network. The CCSDS Time Codes may be of variable length or fixed size at the discretion of the user and may be submitted for transmission at variable time intervals, providing a communication service.

The Time Distribution Protocol provides the capability to synchronize nodes in a SpaceWire network by using SpaceWire time control codes (Time-Codes), providing a timing service.

An Initiator is a SpaceWire node distributing CCSDS Time Codes and SpaceWire time-control codes (Time-Code). An Initiator is also an RMAP initiator, capable of transmitting RMAP commands and receiving RMAP replies. There is only one active Initiator in a SpaceWire network during a mission phase.

A Target is a SpaceWire node receiving CCSDS Time Codes and SpaceWire time-control codes (Time-Codes). A Target is also an RMAP target, capable of receiving RMAP commands and transmitting RMAP replies. There can be one or more Targets in a SpaceWire network.

The protocol also provides means for time-stamping of incoming and outgoing Distributed Interrupts in the Target and makes this information accessible to an Initiator by means of RMAP accesses. Note that Distributed Interrupts are currently being defined in ECSS‐E‐ST‐50‐12C Rev.1 [3].

The protocol also provides means for transferring latency correction information (which can be calculated from the above time-stamp information) from an Initiator to a Target by means of RMAP accesses.

III. Latency Measurement and Correction

Due to the natural latency of transferring time control codes (Time-Codes) in a SpaceWire network, the clock state correction accuracy discussed in [2] will be limited by an offset difference between the initiator and the target (or between targets).

The proposed protocol utilizes the new Distributed Interrupts defined in [3] which are distributed using similar methods as time-control codes and can therefore be used a means for measuring the propagation delay of the latter. Each Target can be configured by the Initiator to perform a time-stamp whenever a Distributed Interrupt with a specified value has been sent or received. The time-stamping is done with the time that is maintained by the Target.

The Initiator performs similar time-stamping at its end and then uses the time-stamps in both ends to calculate the latency or (propagation delay) in either direction. The calculated latency can be written by the initiator to a specific Target register which can be used for correcting the time maintained in the Target.

The proposed protocol thus provides a means for measuring latency between an Initiator and a Target, and provides means for communicating the result to the Target. The methods for how to send and receive Distributed Interrupts, to perform measurements, and to realize the correction in the Target are left to the implementers.

IV. Jitter and Drift Mitigation

Statistical methods and regulation techniques can be used to mitigate the jitter seen on time control codes (Time-Codes) in a SpaceWire network, as discussed in [1]. Jitter and drift mitigation discussed hereafter could be combined.

Drift mitigation by means of clock rate correction [2] can be performed based on periodically received time control code (Time-Code) by a Target. The mean interval between received time control codes (Time-Codes) could be measured with the local time maintained by the Target as reference, and any long term variation could be fed back to the generation of this local time.

The proposed protocol does not provide any means for jitter and drift mitigation, since this does not affect the Time Distribution Protocol itself, being a problem to be solved locally in the Target. The draft ECSS standard could however be extended with implementation guidelines as part of an informative annex.

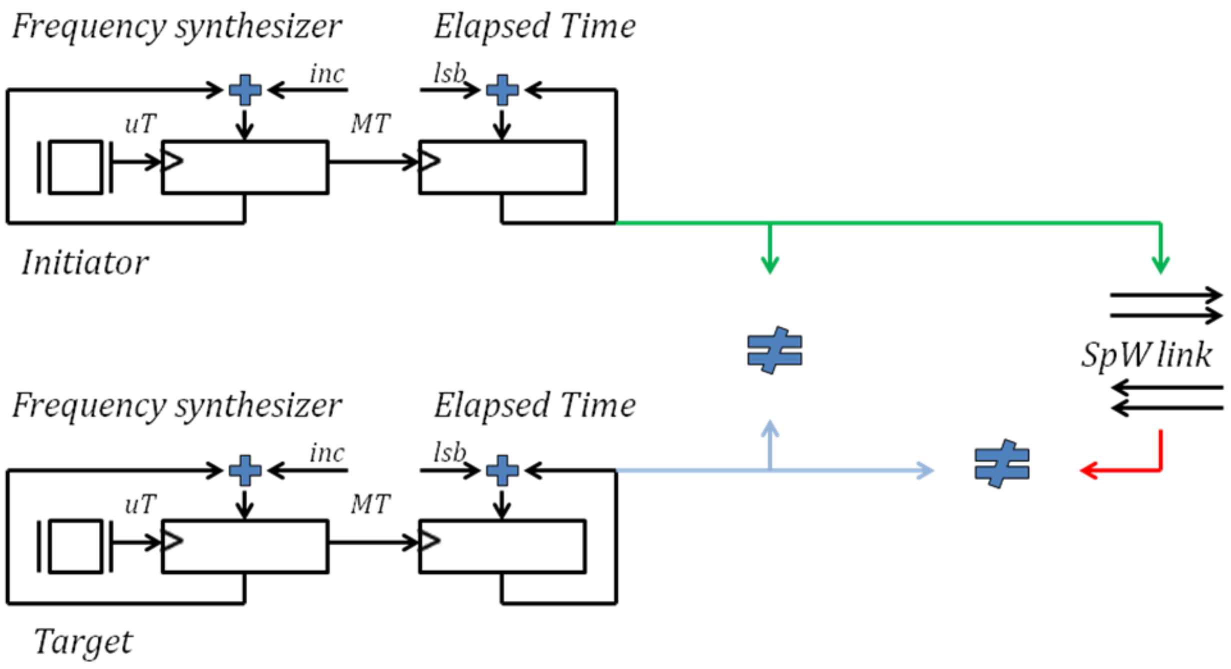


Fig. 1. Problem formulation of jitter and drift problem in a SpaceWire network. The Elapsed Time in the Initiator and the Target(s) are to be synchronized by means of SpaceWire Time-Codes, over SpaceWire links that introduce latency and jitter, and oscillators that introduce drift.

V. Discussion

The proposed protocol is based on distribution of time control codes (Time-Codes) throughout a SpaceWire network, as a means for synchronizing the local time in targets with the local time of the initiator in a system, i.e. providing the timing service. As discussed above, continuous and periodic reception of time control codes (Time-Codes) can also be used for mitigating jitter and drift in the Target.

It has been argued that Distributed Interrupts should not only be used for latency measurements in a SpaceWire network, but also be used as the sole means for synchronization instead of time control codes (Time-Codes). This argument could be valid for low-end systems, but when high accuracy time synchronization over SpaceWire networks is required, one would anyway need to rely on the continuous and periodic distribution of such interrupts to allow jitter and drift mitigation. Time control codes (Time-Codes) are considered more suitable for periodic distribution and have therefore been selected for the timing service in the proposed protocol.

VI. Status and Conclusion

The SpaceWire Time Distribution Protocol is based on existing ECSS standards and CCSDS recommendations. It utilizes basic functionalities such as packet transport and Time-Code distribution, but also takes advantage of newer functions such as Distributed Interrupts. The protocol is fully compatible with RMAP, but it is envisaged that the protocol will have its own protocol identifier in the future, or be part of the future SpaceWire Plug-and-Play protocol. The current draft specification of the protocol does not prevent either solution.

Currently a VHDL IP core is being developed that implements the proposed standard. Additionally, the latency, jitter and drift mitigation methods are being prototyped, with the objective to include them in the aforementioned IP core.

The goal is to include the new IP core in future RASTA prototyping systems and standard ASICs. The IP core will be made accessible through ESA or through Aeroflex Gaisler as part of their GRLIB VHDL IP core library.

An excerpt from the first draft of the proposed standard text is provided as an annex to this article. The excerpt only includes the background and the principles of the protocol. For the detailed requirements the reader is invited to submit a request to the authors.

References

[1] High Accuracy Time Synchronization over SpaceWire Networks - Problem formulation: Jitter and drift of Time-Codes in SpaceWire networks, Sandi Habinc, SPWCUC-REP-0002, 25th June 2012, Version 1.2, Aeroflex Gaisler

[2] Integration of Internal and External Clock Synchronization by the Combination of Clock-State and Clock-Rate Correction in Fault-Tolerant Distributed Systems, Hermann Kopetz, Astrit Ademaj, Alexander Hanzlik, Proceedings of the 25th IEEE International Real-Time Systems Symposium (RTSS 2004), 1052-8725/04, 2004

[3] Space engineering - SpaceWire ‐ Links, nodes, switches and networks, ECSS ‐ E ‐ ST ‐ 50 ‐ 12C Rev.1 Draft 1

[4] European Cooperation for Space Standardization, “Space Engineering; SpaceWire Links, nodes, routers and networks,” ECSS-E-ST-50-12C, July 2008.

Annex to article: excerpt from proposed standard

High Accuracy Time Synchronization over SpaceWire Networks

Time Distribution Protocol

SPWCUC-REP-0003,

29 September 2012, Version 1.1

Introduction

This document represents a draft of an ECSS standard which is in development. The document has been written as far as possible according to ECSS drafting rules. The document should not be considered an ECSS standard.

This version of the document encompasses basic time distribution and initialisation/ synchronization. It does not completely cover configuration, status and low level synchronization.

Foreword

This Standard is one of the series of ECSS Standards intended to be applied together for the management, engineering and product assurance in space projects and applications. ECSS is a cooperative effort of the European Space Agency, national space agencies and European industry associations for the purpose of developing and maintaining common standards. Requirements in this Standard are defined in terms of what shall be accomplished, rather than in terms of how to organize and perform the necessary work. This allows existing organizational structures and methods to be applied where they are effective, and for the structures and methods to evolve as necessary without rewriting the standards.

This Standard has been prepared by the ECSS-E-ST-50-## Working Group, reviewed by the ECSS Executive Secretariat and approved by the ECSS Technical Authority.

Disclaimer

Scope

There is a number of communication protocols that can be used in conjunction with the SpaceWire Standard (ECSS-E-ST-50-12), to provide a comprehensive set of services for onboard user applications. To distinguish between the various protocols a protocol identifier is used, as specified in ECSS-E-ST-50-51.

This Standard specifies the Time Distribution Protocol, which is one of these protocols that work over SpaceWire.

The aim of the Time Distribution Protocol is to synchronize time across a SpaceWire network. It does this by an initiator writing a CCSDS Time Code using an RMAP command placed in a SpaceWire packet, transferring it across the SpaceWire network and then extracting the CCSDS Time Code at the target, and by means of SpaceWire time control codes (Time-Codes) used to convey the time instant at which the CCSDS Time Code becomes valid.

This standard may be tailored for the specific characteristic and constrains of a space project in conformance with ECSS-S-ST-00.

Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this ECSS Standard. For undated references, the latest edition of the publication referred to applies.

- ECSS-S-ST-00-01 - ECSS system – Glossary of terms
- ECSS-E-ST-50-12C - Space Engineering - SpaceWire - Links, nodes, routers and networks
- ECSS-E-ST-50-51C - Space Engineering - SpaceWire protocol identification
- ECSS-E-ST-50-52C - Space Engineering - SpaceWire - Remote memory access protocol
- CCSDS 301.0-B-4 - Time Code Formats, Blue Book

Principles

Purpose

The Time Distribution Protocol has been designed to allow time synchronization across a SpaceWire network, by means of SpaceWire packets and SpaceWire Time-Codes.

Protocol features

The Time Distribution Protocol provides the capability to transfer CCSDS Time Codes between onboard users of a SpaceWire network. The CCSDS Time Codes may be of variable length or fixed size at the discretion of the user and may be submitted for transmission at variable time intervals, providing a communication service.

The Time Distribution Protocol provides the capability to synchronize nodes in a SpaceWire network by using SpaceWire time control codes (Time-Codes), providing a timing service.

An Initiator is a SpaceWire node distributing CCSDS Time Codes and SpaceWire time-control codes (Time-Code). An Initiator is also an RMAP initiator, capable of transmitting RMAP commands and receiving RMAP replies. There is only one active Initiator in a SpaceWire network during a mission phase.

A Target is a SpaceWire node receiving CCSDS Time Codes and SpaceWire time-control codes (Time-Codes). A Target is also an RMAP target, capable of receiving RMAP commands and transmitting RMAP replies. There can be one or more Targets in a SpaceWire network.

The protocol also provides means for time-stamping of incoming and outgoing SpaceWire time-control codes (Time-Code) in the Target, make this information accessible to an Initiator by means of RMAP accesses.

Note: SpaceWire time-control codes (Time-Code) in this context should be interpreted as the Distributed Interrupts currently being defined for ECSS-E-ST-50-12C Rev.1.

The protocol also provides means for transferring latency correction information from an Initiator to a Target by means of RMAP accesses.

Operation

The Initiator and the Target maintain their own time locally, for which the implementation is independent of this standard. The Time Distribution Protocol provides the means for transferring the time of the Initiator to the Targets, and for providing a synchronization point in time.

The time is transferred by means of an RMAP write command carrying a CCSDS Time Code. The synchronization event is signalled by means of transferring a SpaceWire time control code (Time-Code). The transfer of the SpaceWire Time-Code is synchronized with the time maintained by the Initiator.

To distinguish which SpaceWire Time-Code is to be used for synchronization, the value of the SpaceWire Time-Code is transferred from the Initiator to the Target by means of an RMAP write command prior to the actual transmission of the SpaceWire Time-Code itself.

When there is more than a one Target, the CCSDS Time Code need be transferred to each individual Target separately (unless SpaceWire packet broadcast or multicast can be used). Only one transmission of the SpaceWire Time-Code is however need, although one can imagine systems where different SpaceWire Time-Code values are used for different Targets.

Services

The Time Distribution Protocol provides users with communication services based on RMAP service primitives and parameters, transferring amongst others CCSDS Time Codes.

The Time Distribution Protocol provides users with timing service based on SpaceWire time-control codes (Time-Codes).

The followings services are defined in this Standard:

- Configuration
- Status
- Command (CCSDS Time Code)
- Datation
- Timing (Initialisation/Synchronization)
- Time-Stamp (of SpaceWire time-control codes (Time-Codes))
- Latency

Papers Indexed by Author

Papers Indexed by Session

Wednesday 12th June

4LINKS

4Links test and simulation equipment for SpaceWire saves users time, delay, risk, and money. It does exactly what test equipment needs to do. It has proved to be interoperable with every design that it has connected to, while detecting faults including many not found by other methods. It provides information to resolve faults, including long-standing ones, and often without the need to reproduce the fault. And the same hardware can be re-used, for devices, subsystems and complete satellites, at all stages of a mission development.

Even before the technology acquired the name SpaceWire, 4Links supplied SpaceWire in CPLDs for test equipment that passed all tests at the first attempt.

Customer recognition of 4Links quality has led to requests for SpaceWire IP and chips, which we are now supplying, and to numerous accolades such as “4Links equipment is good value, very reliable and very accurate”.

AEROFLEX

Aeroflex Microelectronic Solution divisions supply integrated circuits such as standard products for HiRel applications including FPGAs, LEON 3FT Microprocessors, Logic, MIL-STD-1553 Databus/Transceivers, Clocks, Voltage Regulators and Supervisors, MUXes, Diodes, MOSFETS, LVDS and Memory families and our SpaceWire products - Transceivers, Protocol IP, Routers.

Our RadHard-by-Design Digital and Mixed-Signal ASICs handle design complexities up to 3,000,000 usable gates. We also offer Radiation Testing and Circuit Card Assembly Services.

Aeroflex Gaisler, based in Goteborg, Sweden, is a provider of SoC solutions and IP-cores for exceptionally competitive markets such as Aerospace, Military and Commercial applications. The Aeroflex Gaisler's IPcores consist of user-customizable 32-bit SPARC V8 processor and floating-point-unit cores, SpaceWire cores, peripheral IP-cores and associated software and development tools. The new GR712 LEON Microprocessor is in production. Aeroflex Gaisler solutions help companies develop application-specific SoCs that are highly competitive for customer specific applications. Gaisler Research's personnel have extended design experience, and have been involved in establishing standards for ASIC and FPGA development.

ATMEL

In Europe, ATMEL has 2 main Business Units:

- **MCU: MicroController Business Unit**:
  This BU develops Standard products and Custom products based on AVR8, AVR32 and ARM core. ATMEL is becoming the first supplier of 8bit controllers thanks to its success with many applications and especially the MaxTouch family
- **Automotive, Memory and Aerospace Business Unit:**
  This BU develops products for dedicated Markets and applications

Aerospace developments within ATMEL are all located in Europe, mainly in France (Nantes and Rousset) but also with technical centers supporting ASIC and FPGA business locally (France, Italy, Germany, UK).

There is no involvement of any USA Atmel employees and Aerospace products are guaranteed not restricted by ITAR and EAR rules.

ATMEL Nantes site has been developing Integrated Circuit for space application since 1985. The development team installed now in Nantes and Rousset has a very large experience of radiation hardened circuits design and fabrication constraints.

ATMEL circuits are available in **rad-hard versions** that meet the harsh environment (cumulated dose, latch-up and transient phenomena) of space applications. Design and manufacturing facilities reach international quality standards recognition and are QML-V certified and ESCC QML certified.

High-reliability radiation-hardened products provided by Atmel mean :

- Full military operating temperature range (-55 to + 125°C)
- 100K - 300Krd range, Latch-Up, SEE, SEFI hardened

Atmel proposes advanced technical and competitive solutions for space market for the following products range:

- Processors (32-bit SPARC)
- Memories (Up to 16Mb)
- Communication ICs
- SRAM-based Reprogrammable FPGAs
- NVM 4Mb
- ASICs (up to 30M gates)

Atmel is committed for the long term to support the aerospace industry. Further developments will address 200 MIPS+ SPARC-based microprocessors, >5M equivalent ASIC gates SRAM based re-programmable FPGA, high density EEPROMs and a new generation of rad-hard ASIC libraries with a complexity higher than the 30 million gates.

AXON’ CABLE

The Axon’ group designs and manufactures wire, cable, connectors and cable assemblies for advanced technology applications in the principal fields of space, aeronautics, medical electronics, automotive and scientific research. Headquartered in France (100 Km east of Paris) the Group employs some 1700 staff in 14 subsidiaries across Europe, America and Asia, with an annual turnover of €115 million euro.

Axon’ Cable has been involved in many space projects, including the International Space Station, various LEO and GEO satellites and rocket launchers including Ariane 5, and can boast flight heritage dating back to 1997. The group offers various types of products for space applications:
- ESCC approved wires, cables and connectors,
- lightweight aluminium round cables and braids,
- aluminium bus bars for satellite power distribution,
- MIL-STD-1553 databus looms for digital transmission systems,
- high data rate links for Voice-Data-Image transmission including SpaceWire, IEEE1394, Ethernet, Fibre Channel,
- solutions suitable for the forthcoming multi-gigabit protocol, SpaceFibre,
- and custom-designed products for specific applications.

ELVEES

www.multicore.ru

R&D Center “ELVEES”, OJSC is a leading Russian ASIC design house, number one developer of multicore digital signal processors and “systems on a chip (SOC)” with SpaceWire links: microprocessors, routers, adapters, controllers, ADC/DAC — the largest chipset in Russia for space and telecommunications, navigation and embedded systems.

R&D Center “ELVEES”, OJSC (www.multicore.ru) was founded in 1990 on the base of ELAS Space Corporation that in 1960–80 (USSR) had been involved in space equipment design and development, such as VLSI IC, onboard control and data processing systems, space computers being implemented in “Salyut” computers for the “MIR” orbital Space station. Nowadays ELVEES has its own innovative MULTICORE IC design platform which includes a great 250–40 nm silicon proven analog and digital IP-cores library (SpaceWire IP-cores also), based on the commercial CMOS RadHard/temperature stability libraries suitable for space. ELVEES provides chips, IP-cores, RT-library’s, new generation IP-cameras, tools and software for image compression, adaptive signal processing, optical and radar monitoring, artificial vision, telecommunication and navigation applications.

GLENAIR

GLENAIR – MINIATURIZED CONNECTORS AND CABLES
Glenair manufactures ultra-miniature interconnect solutions for high-performance applications such as missile systems, satellites, and fighter-jets. Our innovative contacts, connectors and cable assemblies are used in air and space platforms that require reliable performance as well as miniaturized packaging. Glenair is the world's largest manufacturer and supplier of both mil-qualified and commercial Micro-D and Nano miniature connectors in wired and unwired space-grade formats. We also offer turnkey flex circuitry assemblies as well as space-grade wire harnesses terminated to our high-availability connector products.

GLENAIR – SPACEWIRE CONNECTORS AND CABLE ASSEMBLIES
Reduced Cost of Ownership, Easy Integration, and High- Performance for Flight and Lab Grade Cable Assemblies. The success of any space mission begins with reliable data transmission and Glenair SpaceWire cables, built to meet the strict standards set forth by ECSS-E-ST-50-12C make this a reality. Our SpaceWire cables offer bidirectional, high speed data transmission rates up to 400 Mbits/s while significantly reducing cross talk, skew, and signal attenuation. By incorporating a serial, point-to-point cable, with low voltage differential signalling (LVDS) reduced costs are realized through an easily integrated data transmission cable. These features allow SpaceWire cables to be incorporated across various satellite programs without the expense of costly design customization. SpaceWire: The Space Industry Data Transmission Standard

Glenair Inc
1211 Airway
Glendale
California
91201-2497
USA

Glenair UK Ltd
40 Lower Oakham Way
Oakham Business Park
Mansfield
Nottinghamshire
NG18 5BY
UK

www.glenair.com

Contact details (Micro D SpaceWire connectors and cables).
Deniz Armani, SNR Scientist – high speed interconnect solutions
Phone: +1 818 247 6000
darmani@glenair.com

Ross Thomson, Business Development Manager – interconnect systems
Phone: + 44 1623 638114
Cell: +44 7711 029 715
rthomson@glenair.com

W.L. GORE

Gore Products Meet the Challenges of Aerospace

Gore's commitment to innovation is based on a thorough understanding of materials and how they interact with their environment — with the result of reliable products for the aerospace industry. Smaller, lighter-weight **cables and cable assemblies** can reduce mass and simplify routing while delivering electrical and mechanical integrity in the most challenging applications.

GORE® Space Cables and Assemblies: SpaceWire Cables

Reduce Costs for High-Quality Flight and Ground Data Transfer

Data transmission is essential to the success of every space mission. Meeting the stringent electrical and mechanical requirements of ECSS-E-ST-50-12C, GORE® SpaceWire Cables provide bidirectional, high-speed data transmission up to typically 400 Mbit/s with minimal crosstalk, signal attenuation, and low skew.

The key to the outstanding performance of GORE® SpaceWire Cables is the proprietary material used in the cable insulation — expanded polytetrafluoroethylene (ePTFE). Using ePTFE, Gore supports LVDS, which allows data to pass through the cable without significant signal loss. By combining this LVDS technology with standard hardware protocols, GORE® SpaceWire Cable provides a simple alternative to he need for customized program designs.

JAXA - Japan Aerospace Exploration Agency

Web: http://jaxa.jp
JAXA, the Japanese space agency, has been collaborating with SpaceWire Working Group since its beginning, and adopting SpaceWire in multiple spacecraft missions including BepiColombo/MMO, ASTRO-H, SPRINT-A, and HAYABUSA-2. JAXA organizes Japan SpaceWire Users Group so that Japanese industries can share experiences and outcomes of SpaceWire R&D.

NEC CORPORATION

NEC Corporation is a leader in the integration of IT and network technologies that benefit businesses and people around the world.

By providing a combination of products and solutions that cross utilize the company's experience and global resources, NEC's advanced technologies meet the complex and ever-changing needs of its customers. NEC brings more than 100 years of expertise in technological innovation to empower people, businesses and society.

For more information, visit NEC space system solutions at http://www.nec.com/en/global/solutions/space/

SHIMAFUJI ELECTRIC

Since 1990, Shimafuji Electric has been developing microcomputer boards including transmission, graphics and other complex peripheral functions and also producing small amount of products for some OEMs. We have more chances to develop evaluation boards for various RISCs and intelligent peripheral functions devices and T-Engine boards/T-Engine appliance products these days.

Shimafuji have joined the SpaceWire Working Group since early days. We developed the 4 port Space Wire to Gigabit Ether Unit and we are developing the 24-link SpaceWire Packet Recorder based on the 12-slots microTCA SpaceWire Backplane system.

STAR-DUNDEE

STAR-Dundee specialises in supporting users and developers of SpaceWire and SpaceFibre; data networking standards for on-board satellites and spacecraft.

SpaceWire is established as one of the main data-handling networks used on many ESA, NASA and JAXA spacecraft and by research organisations and space industry across the world. SpaceWire's speed, simplicity, flexibility and interoperability have contributed to its continuing adoption and popularity.

STAR-Dundee has a comprehensive product line of SpaceWire test and development equipment that can test across all levels of SpaceWire standard. The product portfolio encompasses equipment to enable the design, development, integration and testing of SpaceWire networks and devices, along with industry-leading flight IP cores, chip designs, design services, consultancy and training.

SpaceFibre is an emerging ESA standard networking technology that provides a very high-speed serial data-link for high data-rate payloads. SpaceFibre aims to complement the capabilities of the widely used SpaceWire standard: achieving initial data rates of 2 Gbits/s improving to 5 Gbits/s long-term, capable of operating over fibre-optic and copper cable, reducing cable mass by a factor of four, adding integrated QoS including bandwidth reservation, priority and scheduling, enhancing robustness with FDIR features at all protocol levels, providing galvanic isolation, and multi-laning improves the data-rate further to well over 20 Gbits/s.

SpaceFibre is being developed by the University of Dundee for ESA and STAR-Dundee can now provide SpaceFibre IP Cores and chip designs, SpaceFibre interfaces, SpaceWire to SpaceFibre Bridge, and SpaceFibre link analysis tools; everything needed for the early adoption of this new technology.

The STAR-Dundee team has leading expertise in all areas of SpaceWire and SpaceFibre technology and is committed to helping our customers adopt these technologies, providing continued support through the full development life-cycle.