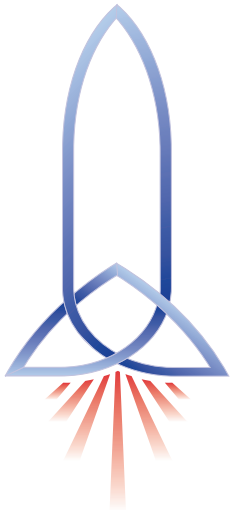
International SpaceWire Conference

Nara 2008

Tuesday 4 to Thursday 6 November

Full Proceedings: Papers and Abstracts



Space
Technology
Centre
University of Dundee

International SpaceWire Conference
Nara 2008
Conference Proceedings

ISBN: 978-0-9557196-1-5

Space
Technology
Centre
University of Dundee

Contents

Papers Indexed by Session

Tuesday 4 November

Tues 15:30-17:00 – Standardisation 1 (90 mins)

Wednesday 5 November

Wed 09:20-09:35 – Standardisation 2 (15 mins)

Wed 09:35-10:30 – Components 1 (55mins)

Wed 10:50-11:45 – Components 2 (55mins)

Wed 11:45-12:10 – Networks & Protocols 1 (25mins)

Wed 13:40-15:20 – Networks & Protocols 2 (100mins)

Wed 15:50-17:05 – Networks & Protocols 3 (75 mins)

Thursday 6 November

Thurs 09:25-10:40 – Onboard Equipment & Software (75 mins)

Thurs 11:00-12:20 – Test & Verification 1 (80 mins)

Thurs 13:20-14:20 – Test & Verification 2 (60 mins)

Thurs 14:20-15:00 – Missions & Applications 1 (40 mins)

Thurs 15:20-16:50 – Missions & Applications 2 (90 mins)

(L) Long papers are 25 minutes
(S) Short papers are 15 minutes

Poster Presentations

* Full paper not included in Conference Proceedings

Papers Indexed by Author

First Author's Surname A-J

First Author's Surname K-R

First Author's Surname S-Z

* Full paper not included in Conference Proceedings

Standardisation 1

Tuesday 4 November
15:30 – 17:00

SPACEWIRE-RT

Session: SpaceWire Standardisation

Long Paper

Steve Parkes and Albert Ferrer

School of Computing, University of Dundee, Dundee, DD1 4HN, Scotland, UK.

Email: sparkes@computing.dundee.ac.uk.

ABSTRACT

SpaceWire-RT (SpaceWire Reliable Timely) aims to provide a consistent quality of service mechanism for SpaceWire and to support proposed CCSDS Spacecraft Onboard Interface Services (SOIS) sub-network services. Furthermore it is intended to support control applications running over SpaceWire where timely delivery of information is essential. To achieve this SpaceWire-RT implements quality of service mechanisms over SpaceWire. SpaceWire-RT forms the quality of service layer of a complete SpaceWire protocol stack which incorporates the Remote Memory Access Protocol (RMAP) and other related protocols currently under development.

This paper introduces SpaceWire-RT, describes how SpaceWire-RT fits into the envisaged SpaceWire protocol stack, outlines the proposed quality of service mechanisms, provides an initial service interface definition for SpaceWire-RT and gives a summary of its architecture.

1 INTRODUCTION

SpaceWire [1] is designed to connect together high data-rate sensors, processing units, memory sub-systems and the down link telemetry sub-system. It provides high-speed (2 Mbits/s to 200 Mbits/s), bi-directional, full-duplex, data links which connect together SpaceWire enabled equipment. Networks can be built to suit particular applications using point-to-point data links and routing switches. The remote memory access protocol (RMAP) [2] was subsequently designed to provide a simple, standard means for one SpaceWire node to write to and read from memory inside another SpaceWire node.

SpaceWire and RMAP operate with a best effort quality of service. While error detection, reporting and recovery techniques are defined in both standards, there is no defined means of recovering any data that was lost or that arrived at its destination in error. Also there is no concept of timeliness in either of these standards. For many SpaceWire applications this is not a problem, but for other applications quality of service is a key issue.

SpaceWire RT [3,4] provides a consistent quality of service framework for SpaceWire and supports both the proposed CCSDS SOIS sub-network services [5, 6, 7, 8] and

control applications running over SpaceWire.. To achieve this SpaceWire-RT implements quality of service mechanisms over SpaceWire.

2 COMMUNICATIONS MODEL

It is important to understand the communication model used by SpaceWire and SpaceWire-RT.

2.1 SPACEWIRE

SpaceWire uses point to point links to directly connect one node to another node. SpaceWire networks can be constructed using wormhole routing switches. Nodes can then be indirectly connected via one or more routing switches.

SpaceWire provides a stream service. The inputs and outputs to a SpaceWire point to point link are FIFOs. Data presented to an input FIFO is transferred across the SpaceWire link to the other end where it appears in and can be read from an output FIFO. A link level flow control mechanism is defined in SpaceWire which prevents overflow of the output FIFO. When the output FIFO becomes full no more data can be transferred across the SpaceWire link until some data is read out of the output FIFO making space available.

There is no management of memory at the destination i.e. there is no mechanism in SpaceWire to reserve memory or other resources in the destination node. If the destination user decides to stop reading data from an output FIFO the corresponding SpaceWire link will block.

2.2 SPACEWIRE-RT

SpaceWire-RT aims to provide a quality of service layer for SpaceWire that gives SpaceWire reliability and timeliness properties. The intention is for this to be done in such a way that any application that used SpaceWire would be able to run over SpaceWire-RT and gain benefit from the QoS provided. This application would be able to run with other applications which required improved QoS (either reliability or timeliness or both) without impairing the QoS of those other applications. To be able to run existing SpaceWire applications over SpaceWire-RT the interface to SpaceWire-RT has to be conceptually the same as that of SpaceWire. The communications model for SpaceWire-RT is thus one of virtual point-to-point connections across a SpaceWire network (referred to as channels) each of which connects a source channel buffer (input FIFO) in one node to a destination channel buffer (output FIFO) in another node.

SpaceWire-RT adds QoS to the SpaceWire paradigm, by providing a stream service over virtual point to point links. Any SpaceWire packet can be sent over a SpaceWire-RT virtual point to point link (channel) receiving the QoS of that channel.

SpaceWire-RT does not provide management of user memory or other resources. It only manages the source and destination buffers (FIFOs) within SpaceWire-RT. Management of these buffers by SpaceWire-RT ensures that no SpaceWire packets are left strung out across the SpaceWire network blocking other traffic while waiting for space in a destination buffer.

3 SPACEWIRE PROTOCOL STACK

SpaceWire-RT is part of the layered protocol stack for SpaceWire which is illustrated in Figure 1.

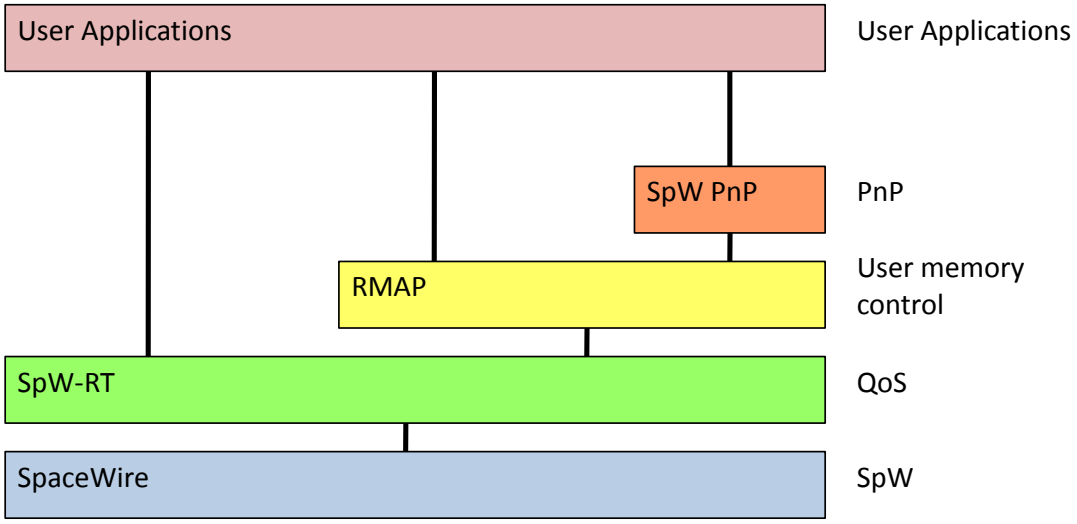


Figure 1 SpaceWire Layered Protocol Stack

SpaceWire is at the bottom of the protocol stack sending SpaceWire packets across the SpaceWire network from source to destination. Immediately on top of SpaceWire is SpaceWire-RT providing quality of service. All traffic has to pass through SpaceWire-RT otherwise the reliability and timeliness QoS cannot be ensured. User applications can talk directly to SpaceWire-RT. RMAP provides a mechanism for reading from and writing to memory in a remote node. SpaceWire-PnP (plug and play) uses RMAP for configuration and management of nodes on the SpaceWire network. User applications can use the services provided by RMAP or SpaceWire-PnP as well as talking directly to SpaceWire-RT. All use the same, consistent quality of service framework.

The CCSDS Spacecraft Onboard Interface Services (SOIS) working group has defined a set of common communication services for use onboard a spacecraft. The SOIS subnetwork layer and three of the services provided are illustrated in Figure 2. The SOIS Packet Service provides for delivery of packets across a subnetwork, the Memory Access Service for the access of memory devices on the subnetwork, and the Device Discovery Service supports plug-and-play capability with notification services.

RMAP and SpaceWire-PnP provide the Memory Access Service and Device Discovery Services. The SOIS Packet Service is provided by a SpaceWire Packet Transfer Protocol (SpaceWire-PTP) that sends packets across the SpaceWire network, providing buffer management and flow control.

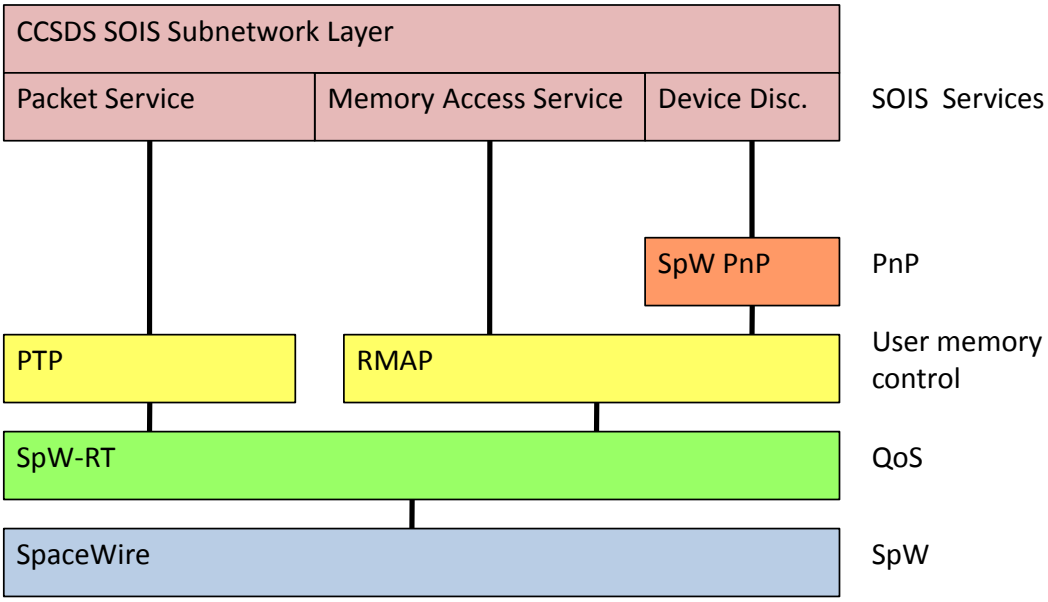


Figure 2 SOIS to SpaceWire Layered Protocol Stack

To send a packet across a SpaceWire network using the CCSDS SOIS Packet Service the packet is presented to the SOIS Packet Service and passed to SpaceWire-PTP which manages buffer space for the complete SOIS packet to avoid sending the packet when there is no room for it in the destination. SpaceWire-PTP passes the packet as a stream of characters to SpaceWire-RT which chops the stream into segments, sends them across the network, and reassembles the SOIS packet at the destination. The SOIS packet emerges from SpaceWire-RT as a stream of characters which are read and put into appropriate packet buffer space by SpaceWire-PTP. The complete buffered packet is then made available to the user application via the SOIS Packet Service interface.

SpaceWire protocols are designed for efficient implementation, high performance and both hardware and software implementation. The SOIS services are designed for generic implementation over various buses or networks, providing a standard software interface.

4 QUALITY OF SERVICE

SpaceWire-RT provides five quality of service (QoS) classes:

- The **Basic QoS** provides a service which does not ensure delivery (i.e. does not provide any redundancy and does not retry in the event of a failure to deliver) and is not timely (i.e. does not deliver information within specified time constraints).

- The **Best Effort QoS** also provides a service which does not ensure delivery and is not timely. This service is different to the Basic Qos only in that Best Effort does not deliver duplicate or out of sequence packets whereas this is possible with the Basic QoS.

- The **Assured QoS** provides a service which is reliable (i.e. retries in the event of a failure to deliver) but is not timely.

- The **Reserved QoS** provides a service which does not ensure delivery but is timely (i.e. when a packet is delivered it is delivered on time).

- The **Guaranteed QoS** provides a service which is both reliable and timely (i.e. it will retry in the event of a failure to deliver and deliver information on time).

5 ASYNCHRONOUS AND SCHEDULED SYSTEMS

To provide timeliness of delivery imposes constraints on the SpaceWire system design. Specifically, a means of ensuring that there is no conflict over the use of network resources is required i.e. every channel has to be given an appropriate share of the available network bandwidth over which it can send information without fear of that communication being held up by other traffic on the network.

Not all systems require timeliness, hence SpaceWire-RT supports two types of system: Asynchronous and Scheduled.

5.1 ASYNCHRONOUS

In an asynchronous system the sending of information over the SpaceWire network is asynchronous i.e. SpaceWire packets are sent when there is a packet in the source to be sent and room in the destination to receive the packet. Priority is used to provide a limited mechanism for controlling timeliness of delivery: a high priority packet will be sent before a low priority one, provided that there is room in the high priority destination buffer.

An asynchronous network supports three QoS classes: Basic, Best Effort and Assured.

5.2 SCHEDULED

In a scheduled system the network bandwidth is split up using time-codes. Time-codes are distributed across the network periodically. Each time-code indicates the start of a time-slot. Each source channel is assigned one or more time-slots when it is allowed to transmit information. Timeliness of delivery is controlled by a schedule table used to specify which source channel can send information in which time-slot. This provides deterministic delivery.

A scheduled network supports five QoS classes: Basic, Best Effort, Assured, Resource-Reserved and Guaranteed.

6 SPACEWIRE-RT SERVICE INTERFACE

To isolate the applications using SpaceWire-RT from the particular type of SpaceWire system being used for communication the same service interface is used for both asynchronous and scheduled systems.

There are six primitives currently defined for the SpaceWire-RT service:

- Send_Data.request (channel, source address, destination address, priority, cargo) which requests to send a Service Data Unit (SDU) from the source node where the request is being made to a destination node on a SpaceWire network;
- Receive_Data.indication (channel, source address, destination address, priority, cargo) which indicates that a SpaceWire-RT packet has been received and which passes the SDU it carried to the SpaceWire user;
- Notify_Delivered.indication (channel, source address, destination address, SDU_ID) which indicates to the user that issued a Send_Data.request over a channel that provided assured or guaranteed services that the SDU was safely delivered to the destination.
- Notify_Error.indication (channel, source address, destination address, SDU_ID, error metadata) which indicates to the user that issued a Send_Data.request that there was a problem delivering the SDU over a channel that provided assured or guaranteed services.
- Configure.request (channel, configuration information) which configures the channel parameters.
- Redundancy_Invocation.indication (channel, reliability metadata) which indicates that one or more retries or redundancy switching were invoked for a channel.

7 ARCHITECTURE

To provide the required QoS capabilities the SpaceWire-RT protocol has to implement a number of different functions each of which is described in the following subsections:

7.1 SEGMENTATION

User information is passed to SpaceWire-RT for sending across a SpaceWire network. The size of this user information is arbitrary and unknown to SpaceWire-RT. SpaceWire-RT sends information across the SpaceWire network in data protocol data units (DPs) each with a size up to a specific maximum DP size (256 bytes). To fit the user information into one or more DPs it has to be split into segments that fit into the available space in the DPs.

The segmentation function is responsible for splitting up the user information into user data segments no larger than a maximum segment size.

7.2 END TO END FLOW CONTROL

End to end flow control manages the source and destination buffers within SpaceWire-RT. A DP for a particular channel cannot be sent unless there is room for it in the destination buffer for that channel. SpaceWire-RT keeps track of the space available in destination buffers for all channels using a flow control mechanism whereby the destination informs the source of the space available in the destination buffer either periodically (scheduled system) or when space becomes available (asynchronous system).

7.3 RETRY

To ensure that data is delivered when there are temporary faults on the SpaceWire network a retry mechanism is required. When a DP is sent using the assured or guaranteed QoS the source keeps a copy of the transmitted DP. When the DP arrives at the destination an acknowledgement PDU (ACK) is sent back to the source. When the source receives the ACK it can free the buffer containing the DP. If no ACK is received within a time-out period due to either the DP or the ACK being lost or corrupted, the DP can be resent to recover from a transitory error.

Retry is combined with the redundancy function to give automatic recovery from transitory and permanent faults.

7.4 REDUNDANCY

The Redundancy Model adopted by SpaceWire-RT is that of alternative paths from a source node to a destination node across a SpaceWire network. SpaceWire-RT supports autonomous switching between alternative paths. These alternative paths may be used in one of two ways:

- Sending data over both paths at the same time, which is referred to as simultaneous retry.
- Sending over the prime path and then if there is a failure using the redundant path.

7.5 ERROR DETECTION

Error detection is needed to support the retry functions and also for error notification for the Basic, Best Effort and Resource Reserved classes of traffic.

7.6 ADDRESS TRANSLATION

SpaceWire can provide up to 223 logical addresses, permitting up to 223 separate nodes. This number is adequate for most foreseen space missions, so for node identification SpaceWire-RT uses the SpaceWire logical address. Path addressing may be used to route a packet to its destination but the node identification is done using the logical address.

The address translation function translates from the SpaceWire logical address to the SpaceWire address that will be used to send the packet across the network. The SpaceWire address can be a path, logical, or regional logical address or an address

constructed using any combination of these addressing modes. The type of address used is dependent upon the redundancy approach being used. Address resolution is used to determine the SpaceWire address bytes that are included in the header of the SpaceWire packet to route it along the required path across the SpaceWire network to its intended destination.

7.7 PDU ENCAPSULATION

The user data segment (UDS), destination address, source address, channel number and other necessary information have to be packaged into a Data PDU (DP) for sending across the SpaceWire network as a SpaceWire packet. The DP encapsulation is illustrated in Figure 3.

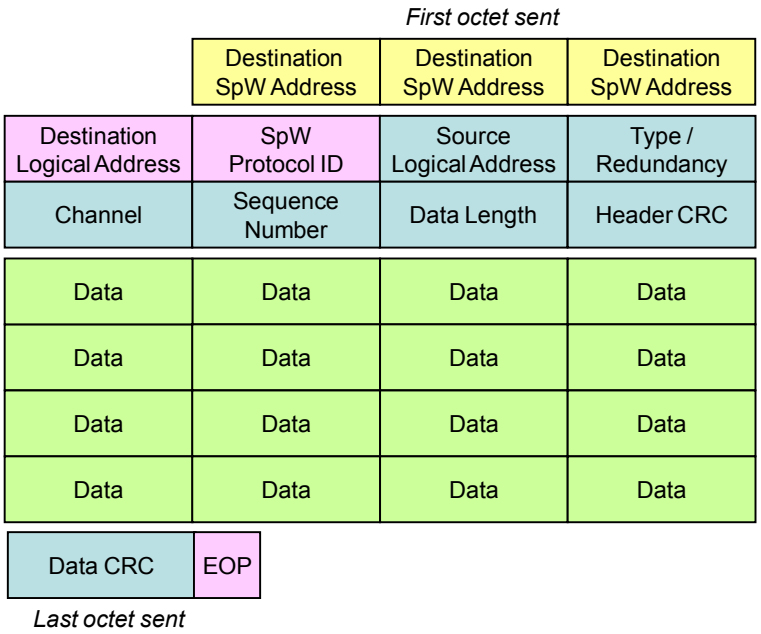


Figure 3 DP Encapsulation

Control information, including ACKs, that are necessary for the operation of SpaceWire-RT also need to be encapsulated into control PDUs.

7.8 PRIORITY

If there are several DPs waiting to be transmitted, the priority function selects which DP to send next based on its priority, the availability of space in the relevant destination buffers, and, for scheduled systems, the schedule and current time-slot. If there is space in the appropriate destination buffers then the DP with highest priority will be sent first.

7.9 RESOURCE RESERVATION

Resource reservation is required for the resource reserved and guaranteed qualities of service to provide timeliness of data delivery. Resource reservation is only provided in the scheduled system which was specifically designed to support it.

In a scheduled network the network bandwidth is separated using time-division multiplexing into a series of repeating time-slots. A schedule table is used in each source to specify which source channel buffer(s) are allowed to send information during each time-slot. The schedule tables in every source are configured to avoid conflicts on the network. Only one source is allowed to send information at a time, or multiple sources can send information at the same time provided that they do not use any common network resource i.e. send information over the same SpaceWire link.

8 FUTURE WORK

SpaceWire-RT is defined in substantial detail in the SpaceWire-RT Initial Protocol Definition [4]. An earlier version of this document was made available to the SpaceWire Working Group to indicate progress with the draft standard and the latest version is now available. The document is not yet complete nor at a stage where a complete prototype SpaceWire-RT systems can be designed.

Current effort is focused on filling the gaps in the draft standard, making it consistent with the CCSDS SOIS protocols, and making it ready for review by the SpaceWire Working Group.

9 ACKNOWLEDGEMENTS

The authors acknowledge the support of ESA for the present work which is funded by ESA under ESA Contract No. 220774-07-NL/LvH.

10 REFERENCES

[1] ECSS, “SpaceWire: Links, nodes, routers and networks”, ECSS-E50-12A, January 2003

[2] ECSS “SpaceWire Protocols”, ECSS-E-ST-50-11C, Draft 1.3, July 2008

[3] S.M. Parkes, “SpaceWire-RT Requirements”, SpaceNet Report No. SpW-RT WP3-100.1, ESA Contract No. 220774-07-NL/LvH, February 2008

[4] S.M. Parkes and A. Ferrer-Florit, “SpaceWire-RT Initial Protocol Definition”, Draft 2.1, SpaceNet Report No. SpW-RT WP3-200.1, ESA Contract No. 220774-07-NL/LvH, October 2008.

[5] CCSDS 850.0-G-R1.1 Draft Green Book Spacecraft Onboard Interface Service Draft Informational Report Jan 2008

[6] CCSDS 851.0-R-1.1 Draft Red Book Spacecraft Onboard Interface Service Subnetwork Packet Service Draft Recommended Practice Jan 2008

[7] CCSDS 852.0-R-1.1 Draft Red Book Spacecraft Onboard Interface Service Subnetwork Memory Access Service Draft Recommended Practice Jan 2008

[8] CCSDS 854.0-R-1.1 Draft Red Book Spacecraft Onboard Interface Service Device Discovery Service Draft Recommended Practice Jan 2008

MIXED SPACEWIRE - SPACEFIBRE NETWORKS

Session: SpaceWire Standardisation

Long Paper

Martin Suess

ESA, European Space Technology Centre,
PO Box 299, 2200 AG Noordwijk ZH, The Netherlands

Steve Parkes

Space Technology Centre, University of Dundee, Dundee, DD1 4HN, Scotland, UK

E-mail: martin.suess(at)esa.int, sparkes(at)computing.dundee.ac.uk

ABSTRACT

In this paper the different levels of SpaceWire and SpaceFibre are compared. The mechanisms for data segmentation and virtual channels used in SpaceFibre are explained and it is shown how traffic from standard SpaceWire links can be connected into and be transmitted via a SpaceFibre network. The requirements on mixed SpaceWire - SpaceFibre networks are discussed and the intrinsic features of SpaceFibre which can be used to provide priorities are pointed out.

1 INTRODUCTION

SpaceFibre is aiming to complement SpaceWire and to overcome some of its limitations. While the data rate is improved by a factor more than 10, the cable length can span up to 100m and the cable mass is significantly reduced. In addition SpaceFibre links can provide galvanic isolation. An important requirement from the beginning was to allow for mixed SpaceWire – SpaceFibre networks and to maintain compliance with the protocols and the routing mechanisms as defined in the SpaceWire standard. This compatibility is important to secure the value and to take full benefit of the investments already made into SpaceWire developments.

2 SPACEWIRE – SPACEFIBRE COMPARISON

SpaceWire has been defined by the standard [1] in 2003. Since then plenty of SpaceWire implementations have been made around the world. Interoperability of the designs has been tested and proven in several missions. SpaceFibre is still at the stage of definition and prototype implementations. ESA has conducted a number of activities to define the properties and to make a prototype implementation of the fibre optical elements, the codec and the end to end link. The objective was to trade-off the technical parameters and to verify the feasibility of the implementation. This paper reviews the current status of breadboards and concepts. Note that there might be modifications and further improvements until the successful standardisation of SpaceFibre.

In order to illustrate similarities and differences between SpaceWire and SpaceFibre both are compared at the different levels starting at the physical level.

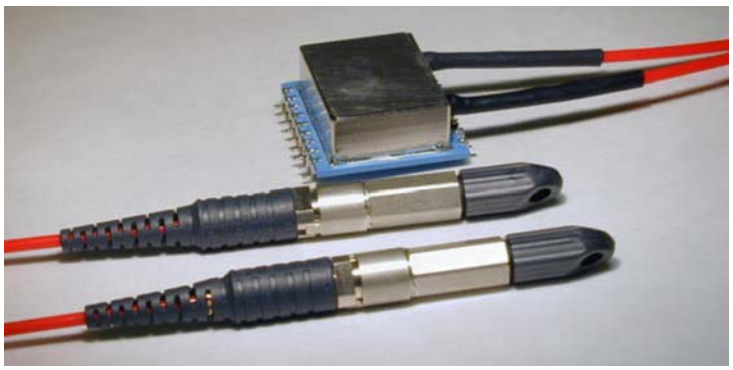


Figure 1: Diamond AVIM connectors and electro optical transceiver breadboard

2.1 PHYSICAL LEVEL

The SpaceWire physical level is defined as a cable with 4 twisted pairs and with the nine-pin micro-miniature D-type connectors. The specified cable assembly is specified to bridge distances up to 10 m.

The SpaceFibre physical level will have two possible implementations. The first implementation is based on optical fibre and second one is based on an electrical copper connection.

A number of optical fibre constructions and implementations where investigated with respect to the requirements on alignment tolerances, bandwidth and radiation sensitivity [2]. The fibre selected after testing a number of options is the Draka MaxCap 300 with is a radhard a laser-optimised graded-index multimode fibre. Radiation tests were performed at a dose rate of 450 Gy/h and with a total irradiation dose of 1000 Gy. The radiation-induced attenuation during this test showed a loss of 7dB for the 100 m of fibre. When transferring this data to the typical dose rates in space the radiation-induced attenuation is expected to be well below 1dB.

A cable around one fibre was designed using an expanded polytetrafluoethylene (ePTFE) buffering system to minimize microbend-induced attenuation changes. The Diamond AVIM connector, which has been already in several space missions, was selected for the SpaceFibre link. For the bidirectional SpaceFibre link connection two of these cables are needed to bridge a distance of up to 100 m.

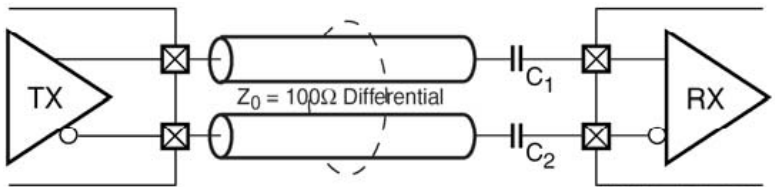


Figure 2: AC coupling of a CML transmitter and receiver

The electrical copper connection has not yet been optimised for space use. During the testing of the breadboard four co-axial cables, two per direction, with SMA connectors have been used. With this a connection was demonstrated over a distance

of several decimetres [3]. Other possible copper media like twisted pairs need to be investigated and characterised before making the final choice. An improvement compared to SpaceWire is the introduction of AC coupling (use of DC blocking capacitors in the signal path) to allow common mode voltages differences between transmitting and receiving side. This can enable also hot plug-in capability.

2.2 SIGNAL LEVEL

The Signal Level of SpaceWire is defined by the LVDS specification according to ANSI/TIA/EIA-644. Two LVDS signals per link direction, the Data and the Strobe signal, are sent in parallel. The Data and Strobe coding used allows to encode the clock signal and to recover it on the receive side by simply XORing the Data and Strobe lines together.

For SpaceFibre the Signal Level is defined by the 850-nm laser light with a peak transmitted optical power of 3dBm. In the modular architecture the light is generated and received in an electro-optical transceiver. The electrical interface of the current prototype of this transceiver employs Current Mode Logic (CML) with a typical peak to peak voltage swing of 1.6V and a common mode voltage of 1.25V at 100Ω differential impedance. This CML defines also the signal level of the electrical version of the SpaceFibre link.

The link uses a single fibre per direction because the clock is transmitted together with the data by means of 8B10B encoding. On the receive side the clock is reconstructed by locking a PLL on the transitions of the encoded signal.

2.3 CHARACTER LEVEL

SpaceWire knows two types of characters, the data and the control characters. In a data character one eight bit value is encoded together with parity bit and data-control flag using 10 bits. There are four 4 bit control characters defined, the FCT, EOP, EEP and ESC. The ESC character followed by the FCT is defined as NULL control code and the ESC character followed by a data character is defined as Time-Code.

SpaceFibre employs the 8B10B encoding scheme [4] where an eight bit value is encoded using 10 bit. In order to ensure a transition rich signal only the balanced 10 bit codes, which have as may one as zeros, and the codes which have a difference of two between the numbers of ones as zeros are used. For any eight bit value which is not represented by a balanced code there are two 10 bit codes assigned, one with positive and one with negative disparity. The aim is to keep the total number of ones and the number of zeros which are sent over the link equal. In order to achieve this goal the character representation with the best disparity is selected before transmitting. This coding scheme allows to define 12 special characters in addition which can be used for control functions. Among them are 3 comma characters (K.28.1, K.28.5, K.28.7) which contain a unique sequence of ones and zeros only present in these characters. Due to this signature they can be used to align the code boundaries in the continuous stream of ones and zeros.

The data characters and the control information are 32 bit aligned. The control information is sent in form of ordered sets. Ordered sets are 4 characters sequences

starting with a comma character (K28.5), the next character determines the type of ordered set while the last to characters can carry two bytes additional information.

There are several types of ordered set defined in SpaceFibre. Those are link-level ordered sets, power management ordered sets, reset ordered sets, flow control ordered sets, framing ordered sets and user ordered sets. Similar to the SpaceWire control characters they are inserted in the data stream at link level to manage certain link functions. The user ordered sets can be inserted in the data stream in addition and are used to implement time-codes or user interrupt functions which can be propagated throughout the network.

2.4 EXCHANGE LEVEL

The exchange level of SpaceWire defines the link initialization, detection of disconnect errors, detection of parity errors, link error recovery and flow control.

In case of a link disconnection or a parity error the SpaceWire link is reset and the connection is restarted after an exchange of silence. This drastic measure interrupts the data flow in both link directions and it reinitializes the character synchronisation and flow control status to guarantee correct operation again.

The flow control in SpaceWire manages the data flow in a way to ensure that the transmitting side is only sending data when the receiving side has sufficient buffer space available. The availability of receive buffer space is indicated to the transmitting side from the receiving side by sending one flow control token (FCT) per every of eight bytes buffer space available.

While the lower levels of SpaceFibre discussed so far where quite different to SpaceWire some important features of the SpaceWire exchange level are maintained in SpaceFibre. The link flow control is one of these features which are maintained and further extended. The higher target speed of SpaceFibre requires that the flow control is operating on larger pieces of data. The 8 characters of 8 bit each per flow control token in SpaceWire are extended to a frame of data of a maximum length of 255 data words of 32 bit each.

Every frame starts with an ordered set indicating the start of frame, a virtual channel number and the number of words in a frame. The frame ends with an ordered set indicating the end of the frame and which contains the 16 bit CRC of the data. This CRC is used to verify that the data have been received without error.

For the transport over the SpaceFibre link the SpaceWire packets are split into frames. The first data word in the frame is reserved to indicate if this frame contains the beginning of a SpaceWire packet, the middle of a SpaceWire packet or the end of a SpaceWire packet. It further contains the information on the location of the first valid SpaceWire packet byte in the second word of the frame and the location of the last valid SpaceWire packet byte in the last word of the frame. This is required because SpaceWire packets are not 32 bit aligned and can have a length which is not always a multiple of 4. At the start of the packet padding can occur when the leading path address byte is deleted during routeing by a SpaceFibre router. Due to header deletion the content and the length of the first frame can change and the frame CRC needs to be recomputed.

In contrast to the data characters sent over SpaceWire the SpaceFibre frames can be distinguished by the virtual channel byte. The flow control ordered sets contain a virtual channel byte as well. By implementing separate frame buffers the data flow in each virtual channel can be separated while using a single physical medium [5], [6]. Congestion in one virtual channel does not influence the traffic in other virtual channels and SpaceWire packets in one virtual channel can pass a packet in another virtual channel. Each virtual channel is able to use the complete bandwidth of the SpaceFibre link. Priorities which are controlling the access of the virtual channels to the physical medium ensure that the higher priority channel has always direct access to the link even if the bandwidth is already fully used by other lower priority channels.

In case of a CRC error in one of the virtual channels it is important that not the complete SpaceFibre link is interrupted and restarted as it is the case for SpaceWire. The SpaceWire receive packet is appended by an EEP, the rest of the transmit packet is spilled and the flow control status of the virtual channel is reset.

In case there is no data to be sent over the SpaceFibre link the link is kept busy by sending IDLE ordered sets similar to the NULL characters in SpaceWire. If there is no data frame to be sent in any of the virtual channel an IDLE frame is automatically generated. Idle frames are terminated as soon as a data frame becomes available.

2.5 PACKET LEVEL

The packet level of SpaceFibre is mostly the same as for SpaceWire. The packet starts with SpaceWire address field which contains zero ore more bytes used to route the packet to the target. This is followed by the SpaceWire logical address of the target and the protocol identifier or extended protocol identifier. After this the cargo follows. The end of packet marker is represented by the end of frame ordered set of the last frame of a SpaceWire packet.

2.6 NETWORK LEVEL

At network level SpaceFibre is intended to be fully compatible with the network level of SpaceWire.

3 MIXED SPACEWIRE – SPACEFIBRE NETWORKS

There are a number of different network topologies which could employ a combination of SpaceWire and SpaceFibre links.

3.1 SINGLE SPACEFIBRE LINK

The simplest topology is of course a single SpaceFibre link connecting e.g. a very high data rate instrument with a mass memory. In comparison a network based on SpaceWire links would require several parallel links to provide the necessary bandwidth. Besides the higher data rate the virtual channels in SpaceFibre allow transmitting the instrument telemetry in parallel to the high rate instrument data over the same medium as well as the instrument control information in the other direction. The different data streams of highly different data rate don’t interfere as they are kept separated by the virtual channels in the SpaceFibre link. A SpaceFibre interface

device for end nodes could have several parallel interfaces which allow injecting and retrieving very high speed data from different virtual channels with an aggregated data rate only limited by the SpaceFibre bandwidth. In order to allow cross strapping for redundancy reasons such a SpaceFibre interface device should provide two SpaceFibre interfaces.

3.2 SPACEFIBRE AS BACK BONE CONNECTION

In some spacecraft the onboard network has to connect clusters of instruments or electronic boxes which are physically placed at different locations. This could for example be the connection between an instrument suite on a mast or detectors in a focal plane with the mass memory, the OBC and the data down link unit. With increased distance the harness mass reduction becomes important as well as the galvanic isolation and robustness against common mode voltage drift. One SpaceFibre link connecting two SpaceWire – SpaceFibre routers could provide a solution for this type of need. In such a case all instruments and electronic units would be equipped with conventional SpaceWire interfaces which are connected to the SpaceWire – SpaceFibre routers to provide the local connectivity between the units. The virtual channels of SpaceFibre can be operated as logical parallel SpaceWire links. It is clear that the bandwidth of one end to end connection is limited by the slower SpaceWire links. A cross bar switch inside the router is able to connect all SpaceWire link interfaces with each other as well as with each of the virtual channel buffers of the SpaceFibre link. The virtual channels are accessed through the router by a path or by a logical address in the SpaceWire packet. To provide basic redundancy such a SpaceWire to SpaceFibre router needs at least two SpaceFibre links in this kind of back bone application.

3.3 MIXED NETWORKS

The need for very high data rate link is often limited to one or few instruments connected to the mass memory. The other instruments and units have lower bandwidth requirements but they need to connect to the same mass memory or to control the high data rate instruments. A single mixed SpaceWire – SpaceFibre network is capable fulfil these requirements. The SpaceFibre connection between high data rate instrument and the mass memory passes through SpaceWire – SpaceFibre routers. The other instruments and units connect to these routers using normal SpaceWire links. The very high speed and the lower speed traffic is separated through the different virtual channels while it runs over the same medium.

3.4 PURE SPACEFIBRE NETWORKS

In some applications there is only the need for very high speed interconnections. In this case a pure SpaceFibre based network can be used. Also there the virtual channels can be used to separate different traffic classes like payload data and telemetry or control data.

4 NUMBER OF VIRTUAL CHANNELS IN SPACEFIBRE

The maximum number of virtual channels in SpaceFibre is 256. In practice the number of implemented virtual channels will be much smaller. Virtual channels

require not only the provision of separate virtual channel buffers but also separate connections to the host system.

In SpaceWire routers the number of ports for path addressing is limited to 31 plus the configuration port. This limitation is inherent to the SpaceWire path addressing scheme and is also applicable to SpaceFibre. There are two ways to overcome this limitation. The SpaceWire standard permits that for very large routers two consecutive path address bytes are used. In the SpaceFibre case the first path address byte could indicate the SpaceFibre link while the second would indicates the virtual channel number. The second possibility is that per SpaceFibre link only a few virtual channels are accessible via path addressing. The other virtual channels would be only accessible via logical addressing.

5 VIRTUAL CHANNEL PRIORITIES AND GROUP ADAPTIVE ROUTING

Each virtual channel can occupy the complete bandwidth of the SpaceFibre link. If two or more connections with a total bandwidth higher than the provided bandwidth of SpaceFibre are accessing the same link some kind of arbitration is needed. One possibility is to assign priorities so that the higher priority virtual channel is granted always direct access. This is a good scheme if there is the need to combine a low rate, low latency traffic and high rate traffic on one link. If there are several concurrent high rate connections a round robin arbitration is preferred. A combination of the two concepts would assign priority levels to the virtual channels and perform round robin arbitration between virtual channels of the same priority.

If these connections are routed thought the same virtual channel the arbitration is provided by the router like in the SpaceWire case. The router can be programmed to provide group adaptive routing among virtual channels of the same priority. This allows for that traffic to the same logical address can take automatic benefit of the availability of several virtual channels.

It should be mentioned that ordered sets used for virtual channel control, time codes or interrupts etc. have always priority over data frames and are directly transmitted in the middle of the current data frame.

The separation of data connections using virtual channels and the possibility to assignment of priorities are important building blocks for controlled qualities of service. The SpaceWire RT protocol will be used to provide the end to end flow control and where quality of service levels beyond priorities are needed.

Figure 3: First prototype implementation of one SpaceFibre link and two SpaceWire – SpaceFibre Routers

6 CONCLUSION

ESA has developed a first prototype implementation in several contracts together with Patria New Technologies Oy, VTT, INO, Fibre Pulse, Gore industry and University of Dundee. The first focus was the development and environmental test of the electro-optical transceivers and the cable based on optical fibre. This effort is continued in a follow-on activity. The second focus was the development of the SpaceFibre CODEC, the SpaceWire – SpaceFibre router and the overall SpaceFibre network concept. The first prototype has been implemented based on Xilinx FPGA using the Rocket I/O as serialiser / deserialiser and as electrical physical layer. A first outline specification [7] has been published and discussed with NASA and US industry in the frame of a SpaceFibre working group. The experience gained will be consolidated and used for the development of a SpaceFibre demonstrator. It is intended to use an Actel FPGA and the Wizard Link [8] as the hardware platform for this coming demonstrator. This will allow to show the feasibility of SpaceFibre links for space flight as both key components are available as qualified parts. Further an IP core of the CODEC will be developed and the necessary documentation to start standardisation will be produced.

7 REFERENCES

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

[2] Toivonen J. et.al., SpaceFibre – High Speed Fibre Optic Links for Future Space Flight Missions, Sixth International Conference on Space Optics, Proceedings of ESA/CNES ICSO 2006, 27-30 June 2006, Noordwijk, The Netherlands

[3] Parkes S., McClements Ch., Suess M., SpaceFibre, Proceedings International SpaceWire Conference 2007, September 2007, Dundee, Scottland

[4] Widmer A. X., Franaszek P. A., A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code, IBM Journal of Research and Development, Communications Technology, Volume 27, Number 5, Page 440, 1983

[5] Dally W.J., Virtual-Channel Flow Control - IEEE Trans. Parallel and Distributed Systems, Vol. 3, No. 2, 1992

[6] Ni L.M., McKinley P.K., A Survey of Wormhole Routing Techniques in Direct Networks - IEEE Computer 1993 Vol.26

[7] Parkes S., McClements Ch., Dunstan M., SpaceFibre Outline Specification, Draft A, 31 October 2007

[8] TLK2711, 1.6 TO 2.7 GBPS TRANSCEIVER, Datasheet Texas Instruments, SLLS501 – SEPTEMBER 2001

SPACEFIBRE VIRTUAL CHANNELS AND FLOW-CONTROL

Session: Standardisation

Long Paper

Clifford Kimmery

Honeywell International Space Electronic Systems

13350 US Highway 19 N, Clearwater, FL, 33764

E-mail: clifford.kimmery@honeywell.com

ABSTRACT

The SpaceFibre CODEC Functional Specification [1] defines a mechanism for identifying virtual channels within a SpaceFibre link and performing flow-control independently for each virtual channel. The SpaceFibre literature [2] [3] envisions the use of virtual channels for multiplexing multiple traffic flows and for providing quality of service features, but does not address the details necessary for standardized implementations.

The virtual channel concept as described in the SpaceFibre literature implements one buffer per virtual channel at each end of the SpaceFibre link. The buffer must be large enough to contain at least one SpaceFibre frame. The link receiver issues a flow-control ordered-set for any virtual channel when the buffer for that virtual channel has room for at least one frame. The link transmitter sends a waiting data frame on a virtual channel if the number of flow-control ordered-sets received for that virtual channel is greater than zero.

The ramifications of implementing virtual channels and flow-control as defined by the SpaceFibre CODEC Functional Specification and described in the SpaceFibre literature are explored. The complexity and link efficiency of various SpaceFibre virtual channel and flow-control implementation choices are evaluated and potential alternatives suggested. The topics addressed include: coupling of SpaceFibre link flow-control to the individual virtual channels, synchronization of the number of active virtual channels between the SpaceFibre link transmitter and receiver, synchronization of the maximum virtual channel frame buffer capacity between the SpaceFibre link transmitter and receiver and the effects of various Quality-of-Service factors such as bounded latency and allocated bandwidth.

VIRTUAL CHANNEL BACKGROUND

The classical use of virtual channels is to perform bandwidth allocation on a network link as a primary mechanism for providing network QoS features. Some standard packet protocols provide virtual channel support under a different name in the form of traffic flow priority levels (RapidIO [4]) or virtual lanes (Infiniband [5]). Others (PCI Express [6] is an example) use the virtual channel term as a synonym for traffic flow priority levels. Connection-oriented protocols (Fibre Channel [7], ATM [8], SDH [9] and SONET [10] are examples) define virtual channels for statically allocating

bandwidth within a link or connection. In all cases, the virtual channels gain access to the physical link based on some arbitration mechanism.

The arbitration mechanisms used by the standard protocols range from fixed time-sequencing (typically to provide guaranteed bandwidth in connection-oriented networks) and fixed packet priority (most packet protocols) to more complex schemes based on the dynamic behavior of traffic flows.

Virtual channels allow a variety of potential SpaceFibre Quality of Service (QoS) features based on allocation of link bandwidth. By establishing standard bandwidth allocation mechanisms for implementation by endpoints and routers, SpaceFibre can support the QoS needs of many different networking applications. The primary purpose of introducing virtual channels in SpaceFibre is to allow specific traffic flows to progress in the presence of congestion on other traffic flows.

SPACEFIBRE VIRTUAL CHANNELS

SpaceFibre virtual channels are part of the SpaceFibre equivalent of the SpaceWire Exchange level as shown in Table 1 – SpaceFibre/SpaceWire Network Model Relationships.

	SpaceFibre Level	SpaceWire Level
	Application	Application
	Network	Network
	Packet	Packet
	Flow-control	Exchange
	Virtual Channel	Not Used
SpaceFibre CODEC	Framing	Not Used
	Link Control	Exchange
	Encoding	Character
	Serialisation	Character
	Signal	Signal
	Physical	Physical

Table 1 – SpaceFibre/SpaceWire Network Model Relationships

A primary goal of SpaceFibre is to provide transparent flow of SpaceWire packets over SpaceWire and SpaceFibre networks integrated in arbitrary configurations. Because of the significantly greater bandwidth provided by SpaceFibre links, the SpaceFibre CODEC Functional Specification defines support for a maximum of 256 virtual channels to be simultaneously active over one physical SpaceFibre link. Each virtual channel can be viewed as a virtual SpaceWire link sharing a single SpaceFibre link with up to 255 other virtual SpaceWire links.

Each SpaceFibre frame header contains the identity of the associated virtual channel. The data contained within the sequence of frames associated with a specific virtual channel identifier are treated by the SpaceFibre routers and endpoints as equivalent to a sequence of SpaceWire packets.

When considered in the context of a typical SpaceWire network and given the historical preference by the SpaceWire community for simple, low-complexity network implementations, the SpaceFibre virtual channel support appears to have

significant capacity for future growth. A maximum-capacity SpaceFibre endpoint can bridge one SpaceWire link to one of the 256 SpaceFibre virtual channels for a maximum of 256 SpaceWire links over one SpaceFibre link. The complexity impact of virtual channel support on SpaceFibre router implementations can be significant as shown in Table 2 – SpaceFibre Router Complexity Scaling (assumes one maximum-size frame buffer per virtual channel per direction).

Ports	Virtual Channels/Port	Equivalent SpaceWire Ports	Minimum Buffer Memory (bytes)	Comment
4	4	16	32,896	Small
4	64	256	526,336	
32	4	128	263,168	
32	256	8,192	16,842,752	Maximum

Table 2 – SpaceFibre Router Complexity Scaling

SPACEFIBRE FLOW CONTROL

The SpaceWire protocol uses a credit-based flow-control mechanism between the link receiver and link transmitter. The link receiver issues credit tokens (flow-control characters) indicating the availability of eight characters of buffer space at the receiver. The link transmitter accumulates the credit tokens to maintain a running total of the available buffer space and transmits data characters at will if the total is greater than zero.

The SpaceFibre CODEC Functional Specification extends the SpaceWire credit-based flow-control mechanism for use with SpaceFibre virtual channels. Each credit token (flow-control ordered-set) represents the availability of buffer space at the receiver for one maximum-length SpaceFibre frame on the associated virtual channel. The SpaceFibre link transmitter accumulates the credit tokens to maintain a running total of the number of available frame buffers on the associated virtual channel and transmits frames at will if the total is greater than zero.

VIRTUAL CHANNEL COUNT INTERACTION WITH FRAME BUFFER COUNT

Since SpaceFibre flow-control uses frame granularity, the SpaceFibre link receiver cannot issue a flow-control ordered-set until a frame buffer is empty. The latency between receiver recognition of the empty frame buffer, transmission of the flow-control ordered-set, transmitter reception of the flow-control ordered-set and transmission of the next frame can be significant. Provision of sufficient frame buffers for each virtual channel to sustain the maximum throughput is ideal.

Since the Spacefibre link is a time-shared resource, the number of frame buffers per virtual channel needed to maintain a high per-channel throughput is inversely proportional to the number of virtual channels actively sharing the link. The throughput effects of flow-control latency for a virtual channel can be partially hidden by the availability of the link to that channel (if the virtual channel cannot transmit because the link is in use by another channel, the flow-control latency may be invisible).

Virtual channels with a greater share of the link bandwidth need a greater number of frame buffers since the flow-control latency is proportionally more visible. As an

example, the sustained throughput capability of a minimal SpaceFibre link receiver (one virtual channel and one frame buffer) is inversely dependent on the link propagation delay.

FACTORS DRIVING SPACEFIBRE BUFFER MEMORY CAPACITY

There are three factors that drive the buffer memory capacity of a SpaceFibre link receiver: the size of each frame buffer, the number of virtual channels that can be simultaneously active on the link and the number of frame buffers used for each virtual channel.

1. In the absence of a frame-size coordination mechanism, the frame buffers must be sized for the maximum-capacity frame (255 32-bit data words).

2. The number of simultaneous virtual channels supported is an implementation decision, but greater numbers of virtual channels offer more application flexibility.

3. The number of frame buffers per virtual channel is an implementation decision that can significantly impact sustained virtual channel throughput.

DECOUPLING SPACEFIBRE LINK FLOW-CONTROL FROM VIRTUAL CHANNELS

While the flow-control mechanism defined in the SpaceFibre CODEC Functional Specification is easy to understand, it forces the SpaceFibre link receiver to allocate buffer space to each virtual channel whether that virtual channel is in use or not. As shown in Table 2 – SpaceFibre Router Complexity Scaling, SpaceFibre link receiver buffer capacity must increase linearly with the number of virtual channels on a port.

The correlation between buffer memory capacity and the number of virtual channels could be minimized by associating SpaceFibre flow-control with the framing level of the link rather than with each virtual channel. RapidIO [4] defines a transmitter-controlled flow-control mechanism that imposes responsibility for link receiver packet buffer management on the link transmitter. The link receiver implements a pool of packet buffers available to all packet priorities (the RapidIO equivalent of virtual channels) and regularly reports the number of available packet buffers to the link transmitter. The link transmitter implements the buffer allocation mechanism to guarantee that low-priority packets cannot block higher-priority packets by consuming all available receiver packet buffers.

Applying flow-control at the virtual channel level is conceptually simple and limits the decision-making complexity of flow-control management. The percentage of dedicated chip resources can be significant since the virtual channel flow-control mechanism requires correspondingly more frame buffer memory at the link receiver (because of the need for frame buffers to be dedicated to each virtual channel) and correspondingly more flow-control credit counters at the transmitter.

In contrast, applying flow-control at the framing level would allow a pool of link receiver frame buffers to be shared by all active virtual channels and a single link transmitter flow-control credit counter. The size of the frame buffer pool becomes an implementation decision that doesn’t limit the number of virtual channels that can be

simultaneously active (overall link throughput would still be affected by the total number of frame buffers and the flow-control latency). The link transmitter must implement an allocation mechanism that manages the available link receiver frame buffers according to the desired QoS.

SPACEFIBRE QUALITY-OF-SERVICE OPTIONS

While the SpaceFibre literature [1][2][3] assigns Quality-of-Service support to the Virtual Channel and Flow Control levels, it leaves the goals and details undefined. Without guidance, speculation on the extent of SpaceFibre QoS support is appropriate.

Since some traditional QoS behaviors associated with reliable delivery are not associated with virtual channels and flow-control, we can assume that such behaviors are to be performed by either the SpaceFibre CODEC or at the Application level. Similarly, the Application level is appropriate for QoS behaviors designed to aid in alleviating network congestion (such as limiting packet ingress rates at the data source and providing end-to-end flow-control mechanisms).

The data-frame-based nature of SpaceFibre virtual channels suggest the feasibility of some form of guaranteed bandwidth/latency QoS similar to SDH [9] or SONET [10]. These connection-oriented network protocols use time-scheduled data frames partitioned to allocate link bandwidth. Because of their roots in telephony, the SDH/SONET time schedules are based on a 125 millisecond repeat interval (corresponding to the 8 kHz sampling rate of typical voice communications). SDH/SONET solves the more complex aspects of time-schedule-based bandwidth allocation by restricting the scheduled time increments to power-of-two divisors of the 125 millisecond interval. Each channel is assigned to a time-slot sized to provide a percentage of link bandwidth no less than the maximum needed by the channel. As a result, the connection-oriented virtual channels provide constant bandwidth regardless of the utilization profile of the application.

Typical QoS behaviors for packet networks are based on priority packet delivery. Packet networks attempt to increase the utilization efficiency of the network infrastructure by dynamically allocating bandwidth on demand rather than reserving bandwidth based on peak needs. The primary side-effect of dynamic bandwidth allocation is network congestion resulting from contention for bandwidth. By introducing packet priority into the bandwidth allocation mechanism, packet networks can allow the higher-priority packets to proceed at the expense of lower-priority packets. The network congestion still exists, but is separated into QoS classes that experience different levels of congestion.

Since SpaceWire packet headers do not include a QoS field, SpaceFibre cannot provide QoS-based packet delivery in the same manner as other packet network protocols. The QoS level must be attached to the SpaceWire packet upon entry into the SpaceFibre network and discarded upon exit from the network. There are two straightforward ways to attach the QoS level to a SpaceWire packet: the first is by associating the QoS level with the virtual channel used and the second is to attach the packet QoS level as a field in the SpaceFibre data frame header. Associating the QoS level with the virtual channel (through network configuration) is a good fit for connection-oriented virtual channel implementations since the connection setup must

be performed using network configuration in any case. Associating the QoS level with the virtual channel is unlikely to be acceptable for dynamic bandwidth allocation in any but the simplest SpaceFibre networks since the number of QoS levels supported is constrained by the number of virtual channels supported by the most limited SpaceFibre link. Attaching the QoS level as a field in the SpaceFibre data frame header allows any virtual channel supporting dynamic bandwidth allocation to convey packets of any QoS level with a minor impact on link bandwidth efficiency.

SpaceFibre could provide a combination of the guaranteed bandwidth QoS of connection-oriented networks and the priority-based QoS behaviors of packet networks in a manner similar to IEEE 1394 [11] by including time-schedule-based (also known as isochronous) bandwidth allocation as the underlying behavior for some virtual channels and allowing the remaining virtual channels to compete for access to the residue link bandwidth on a priority basis. Within a scheduling interval, each time-scheduled virtual channel has a fixed schedule position relative to the other time-scheduled virtual channels and the time-scheduled virtual channels have priority over all other virtual channels. The bandwidth allocation mechanism grants link access to each time-scheduled virtual channel in sequence and the current virtual channel transmits either a data frame or an idle frame (or nothing if some frame jitter is acceptable) based on data availability. After the sequence of time-scheduled virtual channels is completed, the other virtual channels compete for link access based on the priority of the waiting data.

A few of many possible dynamic bandwidth allocation arbitration methods are identified in Table 3 – Potential Arbitration Methods for Dynamic Bandwidth Allocation.

Arbitration Method	Description
Virtual Channel Fixed Priority	The priority of each virtual channel is fixed in hardware based on the channel number.
Virtual Channel Assigned Priority	The priority of each virtual channel is configurable by software.
Virtual Channel Rotating Priority	The priority of each virtual channel is increased by one priority level each arbitration cycle until the channel is granted access to the link. After gaining access to the link, the virtual channel is assigned the lowest priority. Each virtual channel has a different initial priority. The initial priority of the virtual channels is determined using either the Virtual Channel Fixed Priority method or the Virtual Channel Assigned Priority method.
Packet Priority	The priority of each virtual channel is established by the packet priority associated with the frame waiting to be transmitted on that virtual channel.

Table 3 – Potential Arbitration Methods for Dynamic Bandwidth Allocation

SYNCHRONIZING SPACEFIBRE LINK PARTNERS

Because different SpaceFibre router and endpoint implementations are likely to be based on conflicting goals, sufficient configurability is necessary to allow

optimization of network operation. For example, a minimal SpaceFibre link receiver could be implemented using a small maximum frame size and the link transmitter must be configured to match. Another example is a link receiver implementation with the flexibility to reallocate the frame buffers to optimize either the number of virtual channels supported or the number of frame buffers per virtual channel. Similarly, a link receiver might be capable of increasing the number of frame buffers by reducing the maximum frame size.

Allowing SpaceFibre link receivers to implement a frame buffer size less than full-size (255 32-bit words) has ramifications beyond minimizing resource utilization or optimizing network operation. A SpaceFibre router receiving full-size frames from one endpoint and transmitting them to another endpoint incapable of accepting full-size frames would need to implement a form of frame segmenting to allow the data content of the larger frames to be transparently partitioned into smaller frames.

While software control of configurable link partner parameter values is consistent with the SpaceWire philosophy, a hardware-based protocol capable of communicating the link receiver maximum frame size to the link transmitter should be considered since a common understanding of maximum frame size is fundamental to link behavior. An alternative would be to require full-size frame buffers for virtual channel 0 (likely to be used during initial network configuration) after hardware reset with the ability for software adjustment.

Although knowledge of the number of virtual channels supported between link partners is not necessary for proper link operation, there are potential benefits when frame buffers are reassigned from unused virtual channels to increase the number of frame buffers available to in-use virtual channels. In the absence of that knowledge, the link receiver must issue flow-control ordered sets for each virtual channel it is capable of supporting. The link transmitter must ignore any flow-control ordered-sets associated with a virtual channel it doesn't support. Although the link transmitter will never transmit frames on an unsupported virtual channel, the link receiver can only determine that a virtual channel is supported/in-use when a frame is received.

SUMMARY

There are a number of concerns that the SpaceFibre community needs to address as the details of the Virtual Channel and Flow Control protocol levels are defined. Some derive from the traditional SpaceWire preference for low complexity and simple behavior. Others offer opportunities to impact the flexibility of SpaceFibre and the ability to optimize the performance of SpaceFibre networks.

Because SpaceFibre links operate at much greater data rates than SpaceWire links, the number of virtual channels necessary to fill the bandwidth of a SpaceFibre link with typical SpaceWire traffic is substantial (for example, a 3 Gbps SpaceFibre link can carry roughly 30 SpaceWire links each operating at 100 Mbps). The combination of high virtual channel count and the full-size data frame defined by the SpaceFibre creates a need for large frame buffer memory capacity in SpaceFibre link receivers. SpaceFibre routers are particularly affected because of the need to support multiple ports.

The SpaceFibre definition of flow-control on each virtual channel exacerbates the frame buffer memory capacity issue by forcing link receiver frame buffers to be reserved for use by inactive virtual channels. A flow-control method that is independent of the virtual channel implementation would allow the link receiver to use its complement of frame buffers for any virtual channel. By considering a transmitter-controlled flow-control mechanism similar to that defined by RapidIO, the community can make the link receiver frame buffer capacity an implementation decision.

SpaceFibre has the facilities to support a variety of QoS behaviors ranging from guaranteed bandwidth based on time-scheduled frame transmission to various forms of dynamic arbitration. The SpaceFibre community has the opportunity to define standard QoS behavior or to follow the SpaceWire precedent and leave QoS capabilities as implementation decisions.

The full-size SpaceFibre data frame can be an issue for resource-constrained implementations. Allowing the maximum size of data frame buffers to be less than full-size would be beneficial in such circumstances, but introduces the need for link transmitters to segment large frames to fit within the size established for the link.

[1] Steve Parkes, et. al., "SpaceFibre CODEC Functional Specification Draft A" , SpaceFibre Outline Specification.pdf, 31-Oct-2007, University of Dundee

[2] Martin Suess, et. al., "Future Focus: SpaceFibre", 2006 MAPLD International Conference, 20-Sep-2006, http://klabs.org/mapld06/seminars/spacewire/presento/21.ppt

[3] SpaceFibre, Steve Parkes, et. al., 23-Feb- 2008, UoD-SpaceFibre.pdf

[4] RapidIO Trade Association, "RapidIO Rev. 2.0", 03/2008, http://www.rapidio.org/specs/current/Rev2.0_stack2.zip

[5] InfiniBand Trade Association, "InfiniBand Architecture Release 1.0.a", June 19, 2001, http://www.infinibandta.org/specs/

[6] PCI-SIG, "PCI Express Base Specification Revision 2.0", December 20, 2006, http://www.pcisig.com/specifications/pciexpress/specifications

[7] T11/Project 1822-D/Rev 8.1, "Fibre Channel Switch Fabric - 5 (FC-SW-5)", 07/21/2008, http://www.t11.org/ftp/t11/pub/fc/sw-5/08-412v0.pdf

[8] ITU-T I.150, "B-ISDN Asynchronous Transfer Mode Functional Characteristics", 02/99, http://www.itu.int/rec/T-REC-I.150-199902-I/en

[9] ITU-T G.707, "Network Node Interface for the Synchronous Digital Hierarchy (SDH)", 01/07, http://www.itu.int/rec/T-REC-G.707-200701-I/en

[10] ANSI T1.105-2001, "Synchronous Optical Network (SONET) - Basic Description including Multiplex Structure, Rates, and Formats", May 2001

[11] IEEE Std 1394-1995, "IEEE Standard for a High Performance Serial Bus", 1995, ISBN 1-5593-7583-3

SPACEWIRE STANDARD: LOW SPEED SIGNALLING RATES

Session: SpaceWire Standardisation

Short Paper

Chris McClements, Steve Parkes

University of Dundee/STAR-Dundee, School of Computing, Dundee, DD1 4HN, Scotland, UK

E-mail: cmcclements@computing.dundee.ac.uk, sparkes@computing.dundee.ac.uk

ABSTRACT

The SpaceWire standard defines that the start-up speed of a SpaceWire link shall be 10 Mbit/s +/- 10%. Therefore part of the SpaceWire interface logic and the LVDS drivers and receivers must be capable of operating at a clock speed of 10 MHz and a 10 MHz clock source must be available on the PCB (or 5 MHz for double data rate).

When lower clock speeds are required for lower power applications the 10 Mbit/s start-up speed requirement poses a significant problem for successful link connection. At 10 Mbit/s it takes 800 ns to transmit one NULL character therefore approximately 14 can be transmitted during the minimum Started timeout period of 11.64 µs. This gives enough time to receive and decode the SpaceWire NULLs at the other end of the link. At 2 Mbit/s it takes 4 µs to transmit one NULL character and at most two complete NULL characters can be transmitted during the minimum Started timeout. This does not give enough time for NULLs to be exchanged between SpaceWire interfaces and link start-up cannot occur.

1. INTRODUCTION

The Bepi-Columbo MMO Mission Data Processor SpaceWire Interface Design requirements specify the SpaceWire interface to run at a data rate between 2 and 16 Mbits/s exclusively to save power on the instrument. The current design runs on a 2 Mbit/s link speed with an internal clock speed of 2 MHz. As the interface cannot start up at 10 Mbit/s it is not compliant with ECSS-E-50-12 [1].

This short paper describes a worked example of the University of Dundee/ESA SpaceWire IP core operating at 2 Mbit/s and the exchange level changes required to ensure that a link connection occurs.

2. CONTEXT

The SpaceWire initialisation state machine is shown in Figure 1.

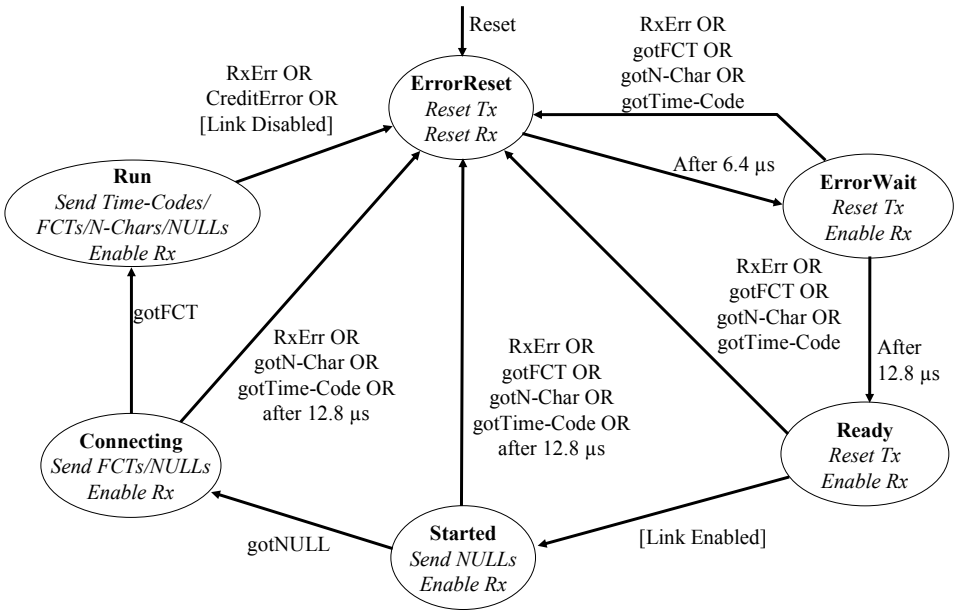


Figure 1 SpaceWire Initialisation State Machine

The state machine timeouts are defined below:

Transition	Nominal Timeout	Lower Limit	Upper Limit
ErrorReset → ErrorWait	6.4 µs	5.82 µs	7.22 µs
ErrorWait → Ready	12.8 µs	11.64 µs	14.33 µs
Started → ErrorReset	12.8 µs	11.64 µs	14.33 µs
Connecting → ErrorReset	12.8 µs	11.64 µs	14.33 µs

Table 2-1 SpaceWire Initialisation State Machine Timeouts

For link initialisation to occur both ends of the SpaceWire link must exchange NULL characters and flow control tokens. NULL characters are exchanged so both state machines can move from Started to Connecting. FCT characters are exchanged so both ends can move from state Connecting to state Run. In state Run all characters can be exchanged and normal operation of the link is performed until an error is detected or the link is disabled.

3. OPERATION AT 2 MBITS

The SpaceWire standard specifies Started and Connecting timeout periods dependent on an initialisation bit period of 10Mbit/s. At 10Mbit/s it takes 800 ns to transmit one NULL character therefore approximately 14 can be transmitted during the minimum Started timeout period of 11.53 µs. This gives enough time to receive and decode the SpaceWire NULLs at the other end of the link.

At 2 Mbit/s it takes 4 µs to transmit one NULL character and at most 2 complete NULL characters can be transmitted during the minimum Started timeout period of

11.53 µs. This does not give enough time for the receiver to decode the First NULL and the state machine to move to state Connecting. The time from receiving the first bit of the first NULL character to entering state Started is shown in Figure 2.

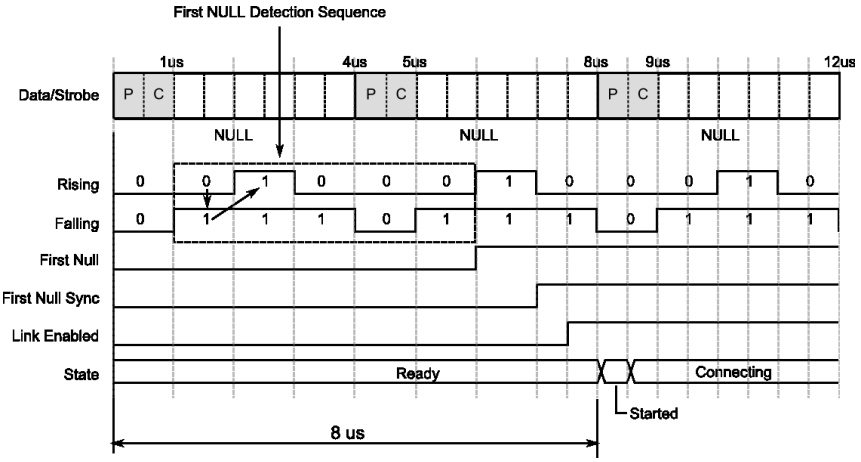


Figure 2 Receive first NULL to Started/Connecting time

The Input bit stream is decoded on the rising and falling edges of the recovered receive clock. The complete first NULL plus the parity and control bit of the rising/falling edge data are shifted into the receiver and checked before the first null signal is set. Another two system clock cycles are required to synchronise the first null signal and an extra cycle is used up by setting link enabled. Once link enabled is set the state machine moves to state started and then immediately to Connecting.

The time from moving to state started to the sending the first NULL character is defined in Figure 3.

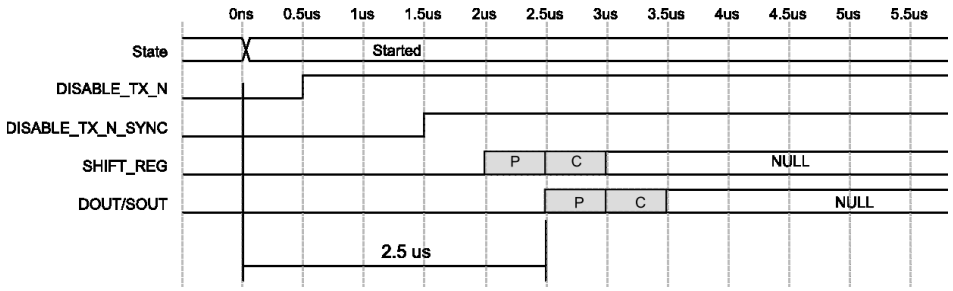


Figure 3 Started to sending first NULL time

One cycle after started is entered DISABLE_TX_N is set low. In the transmitter DISABLE_TX_N is passed through a two stage synchroniser. DISABLE_TX_N is asynchronously reset therefore it must be synchronised. When DISABLE_TX_N_SYNC is set high the shift register is enabled and one cycle later the data strobe pattern is present on DOUT/SOUT. The time to send the first bit of the NULL character is therefore 2.5 µs.

Therefore the basic time to receive and decode the first NULL and instruct the transmitter to reply with a NULL character almost violates the minimum 11.53 us Started timeout and the link cannot start.

4. RESULTS

By increasing the state machine Started and Connecting timeouts for a 2 Mbit/s bit period the First NULL/FCT encoding and decoding can be performed successfully. The proposed state machine time-out changes are listed in Table 2.

Transition	Normal Minimum Timeout	Adjusted Minimum Timeout
ErrorReset → ErrorWait	5.82 µs	5.82 µs
ErrorWait → Ready	11.64 µs	11.64 µs
Started → ErrorReset	11.64 µs	21 µs
Connecting → ErrorReset	11.64 µs	19 µs

Table 2 Normal and adjusted state machine timeouts

The SpaceWire standard defines the minimum bit period before a disconnect can be detected and the maximum bit period where a disconnect is always detected as 727 ns and 1 µs respectively. With a bit rate of 2 Mbit/s a special disconnect detection timer using both edges of the clock is used to provide a disconnect detection period between the upper and lower limits. A better approach would be to increase the timeout period or adjust the tolerance so low speed clocks can be used.

5. CONCLUSION

The SpaceWire standard 10 Mbit/s start-up speed requirement, SpaceWire state machine timeouts and the SpaceWire disconnect period prohibit a SpaceWire link from running at 2 Mbit/s. By extending the Started and Connecting timeout periods without changing the error recovery time of the SpaceWire link (ErrorReset and ErrorWait timeouts) the link can operate successfully at lower clock speeds.

6. REFERENCES

1. ECSS, “SpaceWire: Links, nodes, routers and networks”, ECSS-E50-12A, January 2003.

2. Chris McClements, “SpaceWire CODEC IP - Application Note: 2Mbit/s Operation”, ESA Micro-electronics, 28th May 2008.

Standardisation 2

Wednesday 5 November
09:20 – 09:35

SPACEWIRE STANDARD EVOLUTION

Session: SpaceWire Standardisation

Short Paper

Martin Suess

ESA, European Space Technology Centre,
PO Box 299, 2200 AG Noordwijk ZH, The Netherlands

E-mail: martin.suess(at)esa.int

1 ABSTRACT

This papers lists and describes the modifications to the SpaceWire standard which have been proposed to be included in the next update of the standard. The discussion within the SpaceWire Working Group on the details of the updates will be started next year. Suggestions for updates beyond what is described in this paper are welcome and can still be considered.

2 INTRODUCTION

SpaceWire has been standardised within ECSS in ECSS-E-50-12A and the standard was first published in 2003 [1]. Since then many groups worked on the development of SpaceWire links, nodes, routers and networks and on the application of this technology in space systems. In the past years the standardisation effort aimed at the definition of higher level protocols such as RMAP which is awaiting its standardisation within ECSS as ECSS-E-ST-50-11C - SpaceWire Protocols. In parallel the SpaceWire Working Group is discussing new concepts and additional protocols like SpW PnP and SpW-RT.

Partially through the experience gained with the implementation of real systems and partially through the development of new concepts several issues have been identified to be considered for the update of the ECSS-E-50-12A standard. Among those are the specification of SpaceWire cables and connectors, the introduction of interrupts using the same side channelling mechanism as Time-Codes and the introduction of a configuration port 0 also for nodes. All these issues will be introduced in the next update of the SpaceWire standard which will aim to maintain backwards compatibility. In the following the identified items are discussed per level.

3 PHYSICAL LEVEL

During the past years a number of suggestions have been made to modify the specification of the physical level.

3.1 CABLES

The standard provides a very detailed and rigid specification on the construction of the cable. It specifies e.g. wire type and size of the conductors but also of the shield, filler, binder and jacket material. This kind of specification can be directly given to a

cable manufacturer who can based on this produce a cable compliant to the standard, which is able to transmit the signal over a length of 10 m and support a data rate of 200 Mbps. The disadvantage is that this cable may be too heavy and rigid for some short connections and too lossy for distances beyond 10 m. Some different cable constructions have been proposed in the past. The idea for the update of the standard is to specify not the construction but some physical and electrical parameters. These could comprise parameters like Differential Impedance, Signal Skew, Return Loss, Insertion Loss, Near-end Crosstalk (NEXT) and Far-end Crosstalk (FEXT) [2], [3], [4], [5], [6].

3.2 CONNECTORS

A nine-pin micro-miniature D-type is specified as the SpaceWire connector. It is compact and available for space use. The differential impedance of the D-type connectors does not match the 100 Ω of the cables and the termination. Still in practice the distortion introduced by it is acceptable in most cases. Other connectors like a 4-way twinax connector [2][3][4] or circular 13 pin 38999 Series II connector [6] have been proposed and investigated. It should be discussed if and which additional connector types should be included in the standard.

3.3 CABLE ASSEMBLY

The micro-miniature D-type connector has nine signal contacts. Eight are used for the 4 twisted pair cables and one is used to terminate the inner shields at end of the cable from which the signals are being driven. The inner shields are isolated from one another. This feature can be useful to prevent loops in the grounding design and the symmetrical arrangement avoids the problem of having to know which end of the cable is which during installation.

A problem occurs when the cable is broken into several parts due to bulk head connectors which are often used in larger structures. This leads to the situation that the inner shields on both sides of the bulkhead are not connected to the ground of either side. A connection of the inner shield on both sides with the possibility to implement a controlled capacitive decoupling on one side behind the plug could be investigated as a solution.

3.4 BACKPLANE

SpaceWire links are often used within a unit or electronic box. The current SpaceWire standard contains some requirements on PCB and backplane tracking but no requirements on backplane connectors or backplane construction.

4 CHARACTER LEVEL

The time control codes are defined as an ESC character followed by a single data character. Six bits of time information are held in the least significant six bits of the Time-Code (T0-T5) and the two most significant bits (T6, T7) contain control flags which are distributed isochronously with the Time-Code. The two control flags are reserved for future use and both are set to zero. Since the issue of the SpaceWire standard a number of proposals have been made on how a good use could be made of the three reserved states of the control flags.

4.1 DISTRIBUTED INTERRUPTS

The first proposal is to use one of the reserved states to distribute interrupts through the SpaceWire network using the same side channel mechanism as for Time-Codes. The available bits allow defining 32 Interrupts Codes and 32 Interrupt-Acknowledge Codes. Routers and nodes only propagate the interrupts when they receive it for the first time. At this point a timer is started to reset this lock. It is also reset when the corresponding Interrupt-Acknowledge Code is received [7], [8].

4.2 MULTI-TIME-CODE MECHANISM

Only one node in a SpaceWire network should provide the active TICK_IN signal which triggers the broadcast of the Time-Codes. This is to avoid collisions of Time-Codes within the network. For fail safety and redundancy reasons it can be useful to have simultaneous Time-Codes from different time masters in a system. This could be implemented by using the two remaining reserved states of the control flags [9].

5 EXCHANGE LEVEL

The exchange level is responsible for making a connection and for managing the flow of data across the link.

5.1 SPACEWIRE STATE TRANSITIONS

During the implementation of the SpaceWire codec some inconsistencies in the transitions described in the state diagram have been identified [10].

a) The transition from Started to ErrorReset is impossible when gotNULL condition is set.

b) The transition from Connecting to Run shall be applied only after sending FCT to channel.

These inconsistencies will have to be corrected by making some slight modifications of the standard text and state diagrams.

5.2 SIMPLEX LINK OPERATION

For many high speed payload data applications only a simplex connection from the instrument to the memory is required. In these cases the back channel provided by SpaceWire is often seen as unnecessary complexity and cable mass. It has been proposed to modify the SpaceWire codec and the state machine to support simplex operation [11], [12]. Also the possibility of a half-duplex SpaceWire implementation has been suggested [13].

It remains to be investigated what consequences these changes will have for the backwards compatibility of SpaceWire and if they should be included in the update of the standard.

6 NETWORK LEVEL

The network level is the highest level specified in the standard.

During the development of higher layer protocols a number of issues and clarifications where identified.

6.1 CONFIGURATION PORT IN NODES

Every SpaceWire routing switch has one internal configuration port with address zero. It can be used to configure the routing switch and to obtain status information. This is an important feature for network discovery and PnP. It showed to be a problem that this port zero is only present in routing switches and not in nodes. The update of the definition will align the SpaceWire Node addressing with the SpaceWire Routing Switch addressing. An internal configuration port with address 0 will be introduced for nodes but normal SpaceWire packets starting with a logical address (32 – 254) will be passed to the next layer as before.

6.2 ROUTER FUNCTION IN NODES

What has been described before corresponds to a very simple router with one external port, one internal configuration port and one node internal port. This concept can be extended to several external ports by introducing path addressing and a routing table. This would not only fulfil the needs of network discovery but would also enable an elegant cross strapping method for redundancy switching and easy packet routing through the end nodes.

6.3 ROUTER TIMEOUT

If a router stops receiving data due to an internal failure the packet is stuck and can block some paths in the network. It is difficult to detect and recover this situation from outside the routers. An effective method to recover from this failure condition is to introduce a timeout inside the routing switches which removes the stuck packet from the link after a certain period of time.

7 CONCLUSION

A non exhaustive list of proposed modifications to the SpaceWire standard has been provided in this paper. Additional proposals are welcome and can still be submitted to the author. The different options will be discussed and consolidated within the SpaceWire working group. In many cases breadboard implementations exist already. Based on the results of the discussion the modifications may be included in the next update of the SpaceWire standard.

8 REFERENCES

[1] SpaceWire - Links, Nodes, Routers and Networks, ECSS Standard ECSS-E-ST-50-12C, 24 January 2003

[2] Allen S., “SpaceWire Physical Layer Issues”, 2006 MAPLD International Conference, Washington, DC, September 2006

[3] Mueller J. W. L., “Design Challenges of an Advanced SpaceWire Assembly for High Speed Inter-Unit Data Link,” 2006 MAPLD International Conference, Washington, D.C. September 2006.

[4] Allen S., “SpaceWire cable and connector variations”, ISC 2007, Dundee, Scotland, September 2007

[5] Suess M., “SpaceWire cable characterisation”, ISC 2007, Dundee, Scotland, September 2007

[6] Schierlmann D., Jaffe P., “SpaceWire cabling in an operationally responsive space environment”, ISC 2007, Dundee, Scotland, September 2007

[7] Sheynin Y., “Distributed Interrupts in SpaceWire Interconnections”, 8th SpW WG meeting, Noordwijk, January 2007

[8] Sheynin Y., “Distributed Interrupts in SpaceWire Networks”, draft A, 28 December 2006

[9] Rakow G., NASA multi-time-code mechanism, 2006 MAPLD SpW Seminar, Washington, DC, September 200

[10] Sheynin Y., “SpaceWire Standard Problems”, 6th SpW WG meeting, Noordwijk, May 2006

[11] Sheynin Y., “SpaceWire Features and SOIS Services”, 9th SpW WG meeting, Noordwijk, April 2007

[12] Yablokov E., “Simplex Mode in SpW Technology”, ISC 2007, Dundee, Scotland, September 2007

[13] Cook B. M., Walker C. P. H., "SpaceWire-and-IEEE-1355-Revisited", ISC 2007, Dundee, Scotland, September 2007

Components 1

Wednesday 5 November
09:35 – 10:30

SPACEWIRE RMAP IP CORE

Session: SpaceWire Components

Long Paper

Steve Parkes, Chris McClements, Martin Dunstan

University of Dundee, School of Computing, Dundee, DD1 4HN, Scotland, UK

Wahida Gasti

European Space Agency, Postbus 299, NL-2200 AG Noordwijk, The Netherlands

E-mail: sparkes@computing.dundee.ac.uk, cmcclements@computing.dundee.ac.uk, mdunstan@computing.dundee.ac.uk, wahida.gasti@esa.int

ABSTRACT

The SpaceWire Remote Memory Access Protocol (RMAP) provides a standard mechanism for reading from and writing to memory in a remote SpaceWire node. This simple but powerful capability is already being designed into components like the SpW-10X router and missions like Bepi Colombo and MMS.

The development of a generic IP core implementing the RMAP protocol will enable users to readily implement the RMAP protocols in FPGAs or ASICs, customising it to their specific mission needs. The RMAP IP core is being developed by University of Dundee for ESA. This paper describes the main features of the RMAP IP core, its architecture, interfaces and initial performance.

1 INTRODUCTION

1.1 SPACEWIRE

SpaceWire is a communications network for use onboard spacecraft. It is designed to connect high data-rate sensors, large solid-state memories, processing units and the downlink telemetry subsystem providing an integrated onboard, data-handling network. SpaceWire links are serial, high-speed (2 Mbits/sec to 200 Mbits/sec or higher), bi-directional, full-duplex, point-to-point data links which connect together SpaceWire equipment. Application information is sent along a SpaceWire link in discrete packets. Control and time information can also be sent along SpaceWire links. SpaceWire is defined in the European Cooperation for Space Standardization ECSS-E50-12A standard [1]. It is being used on many space missions.

1.2 RMAP

The remote memory access protocol (RMAP) [2] provides a means for one SpaceWire node to write to and read from memory inside another SpaceWire node. The aim of RMAP is to standardize the way in which SpaceWire units are configured and to provide a low-level mechanism for the transfer of data between two SpaceWire nodes. For example RMAP may be used to configure a camera or a mass memory device. The camera device may then write image data to allocated areas of memory in the mass memory, or the mass memory may read image data from the camera.

RMAP provides three commands: read, write and read-modify-write:

- The read command reads one or more bytes of data from a specified area of memory in a destination node. The data read is returned in an RMAP reply packet.

- The write command writes one or more bytes of data to a specified area of memory in a destination node. An acknowledgement may be returned to the initiator of the write command if requested in the command.

- The read-modify-write command reads a register (or memory) returning its value and then writes a new value, specified in the command, to the register. A mask can be included, in the command, so that only certain bits of the register are written. This may be used to provide a variety of semaphore and handshaking operations.

The RMAP standard is currently going through the European Cooperation for Space Standardization (ECSS) review and approval process and should be issued formally in the first quarter 2009. In the meantime the draft specification has been used for several devices and missions.

1.3 IP CORES

SpaceWire has been adopted for use on many space missions partly because of the ready availability of intellectual property (IP) cores, components, software drivers, and development support and test equipment. A SpaceWire CODEC designed by the University of Dundee and implemented in VHDL is available as an IP core from ESA for European space projects [3]. This CODEC has been designed into several SpaceWire chips including the SpW-10X SpaceWire router chip designed by University of Dundee and Austrian Aerospace and implemented in an Atmel radiation tolerant ASIC [4]. A wide range of development support and test equipment is available from STAR-Dundee Ltd [5] and other organizations.

With the development of the RMAP standard there is a need for an IP core that implements the RMAP protocol. This core needs to be configurable to suit many different applications.

2 RMAP IP CORE CONTEXT

The RMAP IP core fits between User logic and the SpaceWire interface as illustrated in Figure 1. There are two types of RMAP IP core:

- The one that sends out RMAP commands and receives any replies, which is referred to as the Initiator RMAP Interface
- The one that receives RMAP commands executes them and sends out any required replies, which is referred to as the Target RMAP Interface.

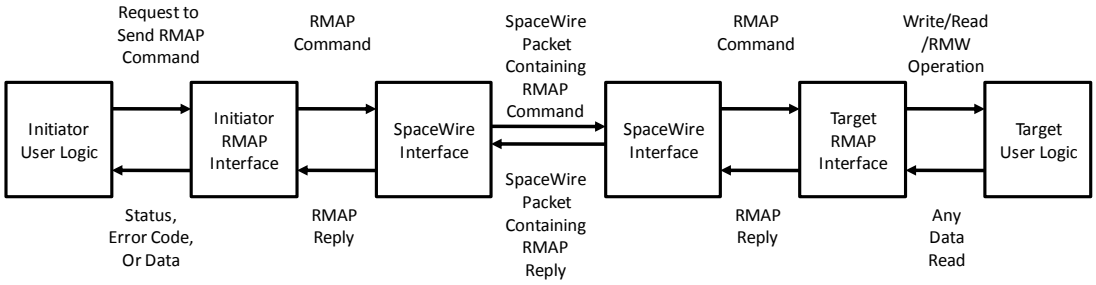


Figure 1 RMAP Initiator and Target

The initiator user logic prepares the command along with any associated data for a write or read/modify/write (RMW) command. In the case of a read command the initiator user logic also prepares information detailing where the data from the read reply is to go. Once all this information is ready the initiator user logic instructs the initiator RMAP interface to send the command. The initiator RMAP interface reads the relevant command and data information from the user logic, forms the RMAP header, generates header and data CRCs and then sends the command. It will also reject the command if the information provided does not correspond to a valid RMAP command. The RMAP command is sent in a SpaceWire packet over the SpaceWire interface.

The SpaceWire packet is received by the SpaceWire interface at the target. If the packet is an RMAP command packet it will be passed to the target RMAP interface which checks that it is valid and then executes the command, writing to or reading from target user logic. In the case of a write command the RMAP data field is written into memory or registers within the target user logic. If an acknowledgment to the write command has been requested then this will be formed by the target RMAP interface and sent out of the SpaceWire interface. In the case of a read command data is read from target user logic memory or registers by the RMAP interface and returned in a reply packet via the SpaceWire interface.

Reply packets travel across the SpaceWire fabric back to the initiator of the original command. They are received by the SpaceWire interface and (if they are RMAP replies) are passed to the initiator RMAP interface. Replies to write commands may signal to the initiator user logic that the command has been executed. Replies to read commands will write the data read from the target user logic to the required designation in the initiator user logic.

A node can contain both an initiator RMAP interface and/or a target RMAP interface i.e. be able to both send and receive commands. An initiator/target is illustrated in Figure 2.

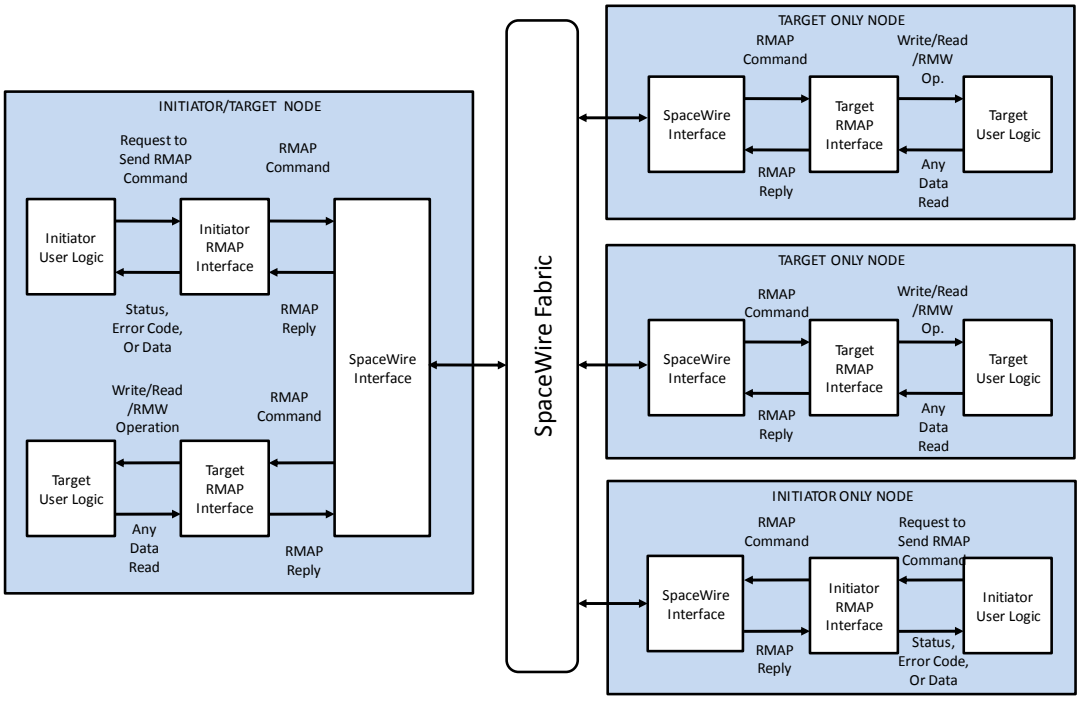


Figure 2 System using RMAP Initiator/Target

3 RMAP IP CORE ARCHITECTURE

The architecture of the RMAP IP core is illustrated in Figure 3.

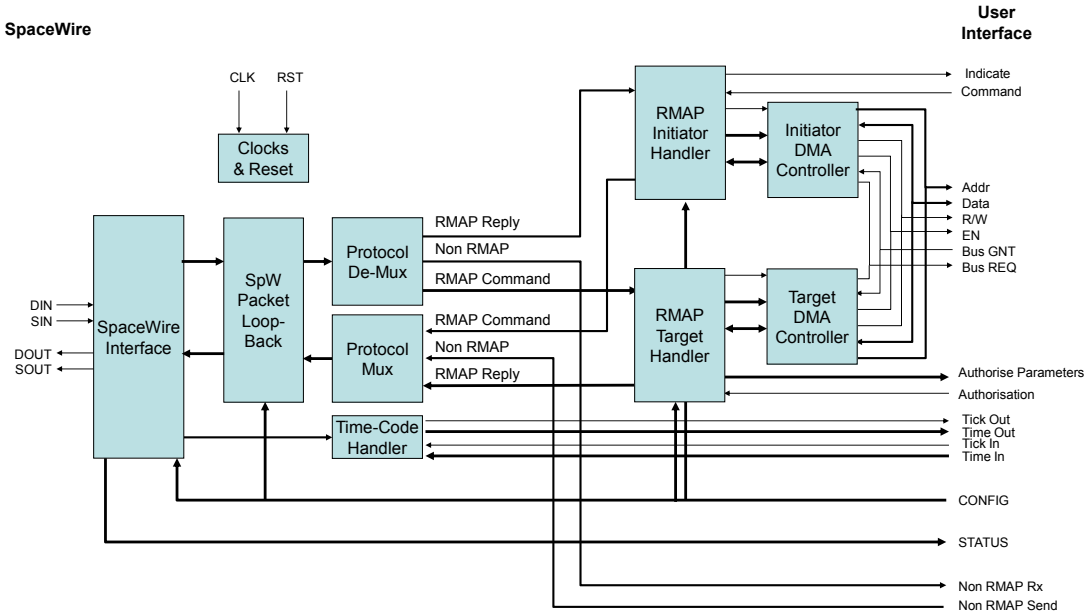


Figure 3 RMAP IP Core Architecture

The SpaceWire Interface is responsible for receiving SpaceWire packets and time-codes and passing them to the SpaceWire Loop-Back and Time-Code Handler respectively. It also transmits SpaceWire packets when requested to do so by the

SpaceWire Loop-Back. The SpaceWire Interface is configured by CONFIG inputs and SpaceWire link status information is made available on the STATUS outputs.

The SpaceWire Loop-Back block provides a means of looping back SpaceWire data characters, EOPs and EEPs. When Loop-Back is disabled, received SpaceWire packets are passed to the Protocol De-Multiplexer and SpaceWire packets from the Protocol Multiplexer are passed to the SpaceWire interface for transmission. When Loop-Back is enabled, received SpaceWire packets are passed immediately to the SpaceWire transmitter, and SpaceWire packets from the Protocol Multiplexer are passed immediately to the Protocol De-Multiplexer. Time-codes are not affected by the SpaceWire Loop-Back block.

The Protocol Multiplexer is responsible for deciding which SpaceWire packet is to be passed to the SpaceWire Interface via the SpaceWire Loop-Back for transmission. There are three sources of SpaceWire packet for transmission: RMAP Initiator Handler which sends RMAP commands, RMAP Target Handler which sends RMAP replies and the non-RMAP interface which may provide non-RMAP packets for transmission.

The Protocol De-Multiplexer is responsible for deciding where received SpaceWire packets are to go: the RMAP Initiator Handler receives RMAP replies, the RMAP Target Handler receives RMAP commands and the non-RMAP interface is passed any other packets.

The RMAP Initiator Handler is responsible for generating and sending RMAP commands based on information provided in user memory. When appropriate it will wait for any reply to the command and inform the initiator user application that the RMAP operation has completed or failed. Any data received with a reply is placed in memory specified by the user application when the RMAP command was generated. Multiple outstanding transactions are permitted.

The Initiator DMA controller provides an interface to user memory for reading RMAP commands from memory that are to be sent and for writing any data in RMAP replies to memory. It is responsible for gaining access to the user data bus and performing necessary memory read and/or write operations.

The RMAP Target Handler is responsible for checking and responding to valid RMAP commands. It will set up the Target DMA controller to perform reads and writes to user memory and registers and will form the reply to the RMAP command for sending by the SpaceWire interface. The RMAP Handler is configured by the CONFIG inputs and status information from the RMAP Handler is available on the STATUS outputs.

The Target DMA controller provides an interface to user memory and registers for writing data sent in an RMAP command to memory or reading from memory specified in an RMAP command. It is responsible for gaining access to the user data bus and performing memory read and/or write operations as determined by the RMAP command. The Target DMA controller competes with the Initiator DMA controller for access to the user data bus.

The Time-Code Handler is responsible for checking time-codes and maintaining the value of the time-code counter. It will assert the TICK_OUT signal when a valid time code is received and put the value of each valid time-code on the TIME-OUT output. The time-code handler can also generate time-codes using the TICK_IN and TIME-IN inputs.

The Configuration and Status registers hold configuration and status information for the RMAP IP core. On power up certain configuration registers are loaded with default values specified by the CONFIG interface. Thereafter the configuration values may be changed by writing to the configuration registers either by a SpaceWire-RMAP command or by the user logic writing to the appropriate registers. Status information from the RMAP IP core is held in status registers which can be read by SpaceWire-RMAP command or by the user logic. Certain status information is also available on dedicated signals, STATUS, from the RMAP IP core.

The Clock and Reset block is responsible for providing the user reset signal, RESET, to the relevant parts of the SpW/RMAP IP core ensuring a clean condition after the reset signal has been asserted. It is also responsible for generating any necessary clock signals from the single clock input signal, CLK.

4 RMAP IP CORE PERFORMANCE

The RMAP Target IP core has been extensively tested using an in-house test bench that performs extensive testing, covering many possible configurations and error conditions.

The RMAP Target IP core has been implemented in both Xilinx and Actel FPGAs. On the Xilinx Spartan 3 device the complete design related SpaceWire interface operates comfortably with link speeds of 200 Mbits/s. An initial implementation on an Actel AX1000 FPGA operates at 100 Mbits/s transmit speed and over 150 Mbits/s receive. Little effort has been spent on optimising either design for performance.

5 CURRENT AND FUTURE WORK

At present work is focussed on completing the RMAP Initiator design and testing this in Xilinx and Actel FPGAs. Two alpha customers are currently working with the RMAP Target IP core providing feedback on the design.

Some effort will be spent improving the performance of the Actel AX1000 implementation.

The RMAP IP core will be available from ESA for use on ESA projects only and from STAR-Dundee Ltd for use on any other project.

6 ACKNOWLEDGEMENTS

The authors acknowledge the support of ESA for the present work which is funded by ESA under ESA Contract No. 220774-07-NL/LvH.

7 REFERENCES

[1] ECSS, “SpaceWire: Links, nodes, routers and networks”, ECSS-E50-12A, January 2003

[2] ECSS “SpaceWire Protocols”, ECSS-E-ST-50-11C, Draft 1.3, July 2008

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

[4] C. McClements, S. Parkes, G.Kempf, “SpW-10X SpaceWire Router User Manual”, Issue 3.4, July 2008 available from http://www.atmel.com/dyn/products/product_card.asp?part_id=4339

[5] www.star-dundee.com

SPACEWIRE CODEC IP CORE UPDATE

Session: SpaceWire Components

Short Paper

Chris McClements, Steve Parkes

University of Dundee/STAR-Dundee, School of Computing, Dundee, DD1 4HN, Scotland, UK

Kostas Marinis

European Space Agency, Postbus 299, NL-2200 AG Noordwijk, The Netherlands

E-mail: cmcclements@computing.dundee.ac.uk, sparkes@computing.dundee.ac.uk, Kostas.Marinis@esa.int

ABSTRACT

The University of Dundee developed the ESA SpaceWire IP core as the first step in the development of the SpaceWire router ASIC device, now available as Atmel standard part AT7910E. The core is used widely in many ESA contracts and is available from ESA for use on ESA projects, or under license from STAR-Dundee. To date the IP core, named "SpaceWire-b" by the ESA micro-electronics section, has been licensed for use in over 40 ESA projects [1].

The SpaceWire IP core was first released in 2003 and presented at the first SpaceWire seminar at ESTEC. Development of the IP core continued at the University of Dundee and three major revisions of the VHDL code have been released, adding extra features and fixing known issues.

1. INTRODUCTION

The SpaceWire [2] system is based on nodes connected together indirectly through routers or directly node to node via SpaceWire links. The role of a SpaceWire CODEC [3] in a system is the physical device which encodes and decodes the serial bit-stream over the SpaceWire links. The CODEC is implemented in technology independent RTL VHDL code. This paper presents a reference design example of the University of Dundee SpaceWire CODEC targeted at an Actel RTAX. The reference design is available with the new 2.03 release of the SpaceWire CODEC which is currently under review for release.

Actel Axcelerator RTAX devices are radiation tolerant version of the Axcelerator series of devices which are protected against single event latch-up and employ triple mode redundancy to resolve single event upsets in the internal FPGA fabric. Commercial grade AX devices are available, providing a prototyping path for lab testing.

2. SYSTEM DESIGN

An overview of the reference design is shown in Figure 1.

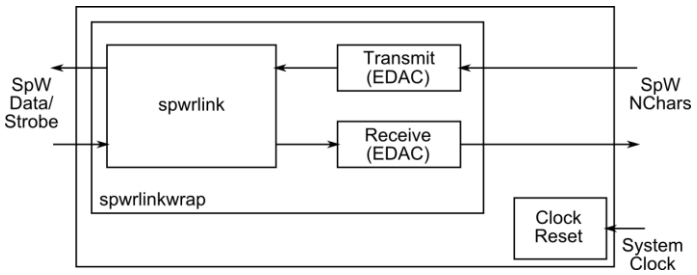


Figure 1 Reference Design Overview

The system clocks all the flip-flops except the receive clock domain. An HCLK global resource clock buffer is used for the system clock and is assigned an HCLKBUF primitive in the top level VHDL spwrlink_top.vhd file.

The system clock frequency is 100 MHz generating a 100 Mbps bit stream. The default 10 Mbps data rate is generated by enabling the transmitter flip-flops every three clock cycles. This is accomplished internally in the CODEC using the SYS_EN configuration of the transmit clock. The transmit rate can be further divided at run time using the TXRATE input signal.

A 100 MHz system clock with a 10 ns period is sufficient to give a valid 10MHz, 100 ns period, reference clock enable pulse for the timing blocks. In this case the CFG_SLOW_CE_SEL configuration signal is not set and the internal clock enable generator is used.

The receive clock domain is used exclusively inside the spwrlink block. The receive clock is attached to an internal CLKINT buffer. The receive clock frequency is 50 MHz for a 30 MHz input bit stream.

The system reset is a high fanout net which is internally routed on an RCLK global resource.

The transmit FIFO is implemented using an Actel AX memory block component (RAM64K36). In the Actel RTAX parts the same RAM block can be used with EDAC protection using an Actel smart design core. The FIFO control logic and buffer pointers are implemented in the FPGA fabric and are automatically protected from single event effects. The transmit FIFO RAM core is generated in Libero in the implementation section of this document. Scrubbing is not performed on the memory block.

The receive buffer is implemented using an Actel AX memory block component (RAM64K36). In the Actel RTAX parts the same RAM block can be used with EDAC protection using an Actel smart design core. The receive buffer RAM core is generated in Libero in the implementation section of this document. Scrubbing is not performed on the memory block.

3. RESULTS

The University of Dundee SpaceWire CODEC is a customisable model of the SpaceWire point to point serial interface specification, the SpaceWire standard. In this reference design example the CODEC is configured to run of a single clock in single data rate mode. A wrapper file is placed around the CODEC to include Actels EDAC

RAM blocks and global resource buffers. The main features and results of the reference design are given below.

- Error detection and recovery Actel Smart Design cores for input and output FIFOs.
- Single data rate transmitter implementation with one system clock and one receive clock.
- Step by step implementation guide using Actel Libero IDE, Synplify/Synplify Pro and Actel Designer.
- Layout guidelines and static timing analysis of the sensitive data recovery flip-flops.

Performance

The implementation is targeted to run at 100 MHz. The actual performance figures are listed in Table 2-1 Clock Performance and are discussed in section 8.3. Using the reference design a system clock frequency of 115.52 MHz can be achieved giving an output bit rate of 115.52 Mbps.

Clock	Requested	Achievable	Description
SYSCLK	100 MHz	115.52 MHz	System clock frequency.
RX_CLK	50 MHz	84.02 MHz	Receive clock frequency

Table 1 Clock Performance

Setting the Radiation setting in the Device Selection Wizard – Operating Conditions to 100Krad degrades the performance as follows

Clock	Requested	Achievable	Description
SYSCLK	100 MHz	112.18 MHz	System clock frequency.
RX_CLK	50 MHz	84.97 MHz	Receive clock frequency

Table 2 Clock Performance (100Krad degradation)

Resource Usage

The reference design is targeted to be as small as possible on the device. The area resource usage of the device is shown in Table 2-3 and the global buffer network resource usage is listed in Table 2-4.

Resource	Used	Available	Percentage Used	Description
R-Cells	508	6048	8.4%	Register elements
C-Cells	1095	12096	9.1%	Combinatorial elements
R+C-Cells	1603	18144	8.8%	Combined total
RAM	2	36	5.5%	Internal RAM blocks
IO	74	198	37.3%	Input output pads.

Table 3 Resource Usage

Net	Global	Fanout	Description	Inferred
i_sysclk_buf	HCLK	434	System clock	No
i_rclk_rst_n	RCLK	344	System reset	No
spwrlinkwrap_1/spwrlink_1/RX_RST_N_buf	RCLK	75	Receiver Reset	Yes
spwrlinkwrap_1/spwrlink_1/RX_CLK_buf	RCLK	114	Receive Clock	Yes

Table 4 Global network resource usage

Estimated Power Consumption Using the Smart Power estimation tool the following power estimations.

Parameter	Power (mW)	Percentage
Total Power	181.09	-
Static Power	97.01	53.6 %
Dynamic Power	84.08	46.4 %

Table 5 Power Estimation

4. CONCLUSION

The SpaceWire CODEC RTAX reference design is capable of achieving a 100 Mbit/s data rate using error recovery and detection FIFOs. The design has been proven using the Axcelerator commercial packages running at 100 Mbit/s in the lab. The latest version of the SpaceWire CODEC and the Actel reference design will be available after internal review by ESA and further testing of the design will be performed to achieve a 200 Mbit/s double data rate design. The core is used widely in many ESA contracts and is available from ESA for use on ESA projects, or under license from STAR-Dundee for non ESA contracts.

5. REFERENCES

[1] http://www.esa.int/TEC/Microelectronics/SEMKBWSMTWE_0.html, ESA IP Cores Usage Statistics, Updated on 8th Jan 2008

[2] ECSS, “SpaceWire: Links, nodes, routers and networks”, ECSS-E50-12A, January 2003

[3] "The SpaceWire Codec", C. McClements, S. Parkes and A. Leon, ISWS International SpaceWire Seminar 2003 (2003), pp.139-146. Noordwijk, The Netherlands, 4-5 November.

SPACEWIRE PHYSICAL LAYER FAULT ISOLATION

Session: Test, SpaceWire Components

Short Paper

Barry Cook

4Links Limited, Bletchley Park, MK3 6EB, England

Wahida Gasti, Sven Landstroem

ESA/ESTEC, Noordwijk, The Netherlands

E-mail: Barry@4Links.co.uk, Wahida.Gasti@esa.int, Sven.Landstroem@esa.int

ABSTRACT

The SpaceWire physical layer is required to use Low Voltage Differential Signalling (LVDS) as defined in the document ANSI/TIA/EIA-644 (A).

It has been shown that a likely failure mode in such drivers can result in high fault currents that can propagate between SpaceWire links and cause catastrophic failure of systems - including redundant systems when cross-strapping is used. Great care must be taken in power systems to limit such excessive effects.

We begin by describing the usual (LVDS) physical layer implementation, its consequence and power system requirement implications. We then consider receiver protection and its limitations. Alternative driver designs that can significantly reduce the undesirable effects resulting from component failure are described.

1 OVERVIEW

SpaceWire [1] is a relatively high speed (up to 400Mb/s) link protocol intended to be used for point-to-point data links or, in conjunction with suitable routing switches, for fault tolerant networks. The physical layer is required to use Low Voltage Differential Signalling (LVDS) as defined in the document ANSI/TIA/EIA-644 (A) [2].

Use of multiple links, whether for nominal and redundant point-to-point links, or for networks, requires interconnections between links – at least for data transfers. It has been shown [3] that a failure in one link can propagate between links unless care is taken to provide mechanisms to provide isolation between links. We consider normal operation and behaviour under failure conditions in order to suggest suitable isolation mechanisms.

2 NORMAL OPERATION

LVDS specifies a low differential output voltage, nominally 350mV, sitting at a defined common mode level, nominally 1.25V. The nominal output terminal voltages are thus between 1.075V and 1.425V. This allows an end-to-end voltage difference between ground wires (the common mode voltage range) of ±1v whilst keeping the receiver input terminal voltages between 0.075V and 2.425V.

It is often believed (and implied in [1]) that LVDS drivers must be current sources. In fact, the opposite is true – they can't be current sources. If they were current sources then the output common mode level cannot be controlled – but it is essential that it is closely controlled. Note that the standard specifies output voltages, not currents.

3 Fault Scenario

The fault that concerns us here is that of the supply rail to the LVDS driver rising above its maximum level. This can be caused by a power-supply fault The common method of achieving ground isolation and secondary regulated supply voltages, is via insulated DC/DC converters which have credible failures causing over-voltage emission. Depending on the topology chosen, the over-voltage emission can be predictable (buck topology, limited value) or non-predictable (boost topology, "non-limited" value). Furthermore, the DC/DC converter may have Over-Voltage Protection (OVP) built in or it may not. In the second case, which is usual for low cost off-the-shelf DC/DC converters, the possible failure propagation paths from the supply voltage failure is a serious matter. A non protected "flyback" (i.e. boost) converter may easily cause over-voltage emission of 2-5 times nominal voltages, if not protected by OVP. Even if OVP exists, the LVDS user (the designer) must be careful on exactly what peak voltage level the DC/DC converter will emit in dynamic mode, i.e. how big the max over-shoot will be before the DC/DC converter is successfully shut-down or clamped.

Excessive supply voltage on the LVDS driver is likely to result in its failure and, unfortunately, the likely failure mode here (as it is for the regulator) is to propagate the excessive voltage to its output pins. This voltage is passed to the input pins of the receiver and, after causing this device to fail, might be passed to the receiver supply rail. This catastrophic failure scenario is rarely assumed or analysed in the FMECA analysis on system design level. The analysis is often depending on the equipment designer's knowledge about power supplies and in fact often overlooked.

Power supply regulators are most commonly designed only to source current, not to sink DC current supplied from high voltages on device input pins. Excessive signal input voltage can result in a rise in supply voltage at the receiver – leading in turn to failure of nearby transmitters and propagation of the fault to other receivers etc. The effect may, in this way, be passed to redundant circuits. See Figure 1 (taken from [3]).

4 Controlling the fault at source

The first line to consider is preventing the initial fault damaging the LVDS drivers. Some sort of supply rail over-voltage detector and limiter would appear to be enough.

We must, however, consider the practicality of such devices … they must allow a normal supply rail voltage – for example, 3.3V nominal (acceptable range 3.0 to 3.6V) – and clamp an excessive voltage – 4V maximum in this example. That is a small difference which in a true worst-case analysis may be very hard to prove.

We must also consider whether the protection device can clamp the rail voltage cleanly or whether there will be a transient over-voltage, possibly caused by intentional or stray inductance in the circuit. Even a transient can cause damage: semiconductors are noted for the speed at which they fail with over-voltage stress.

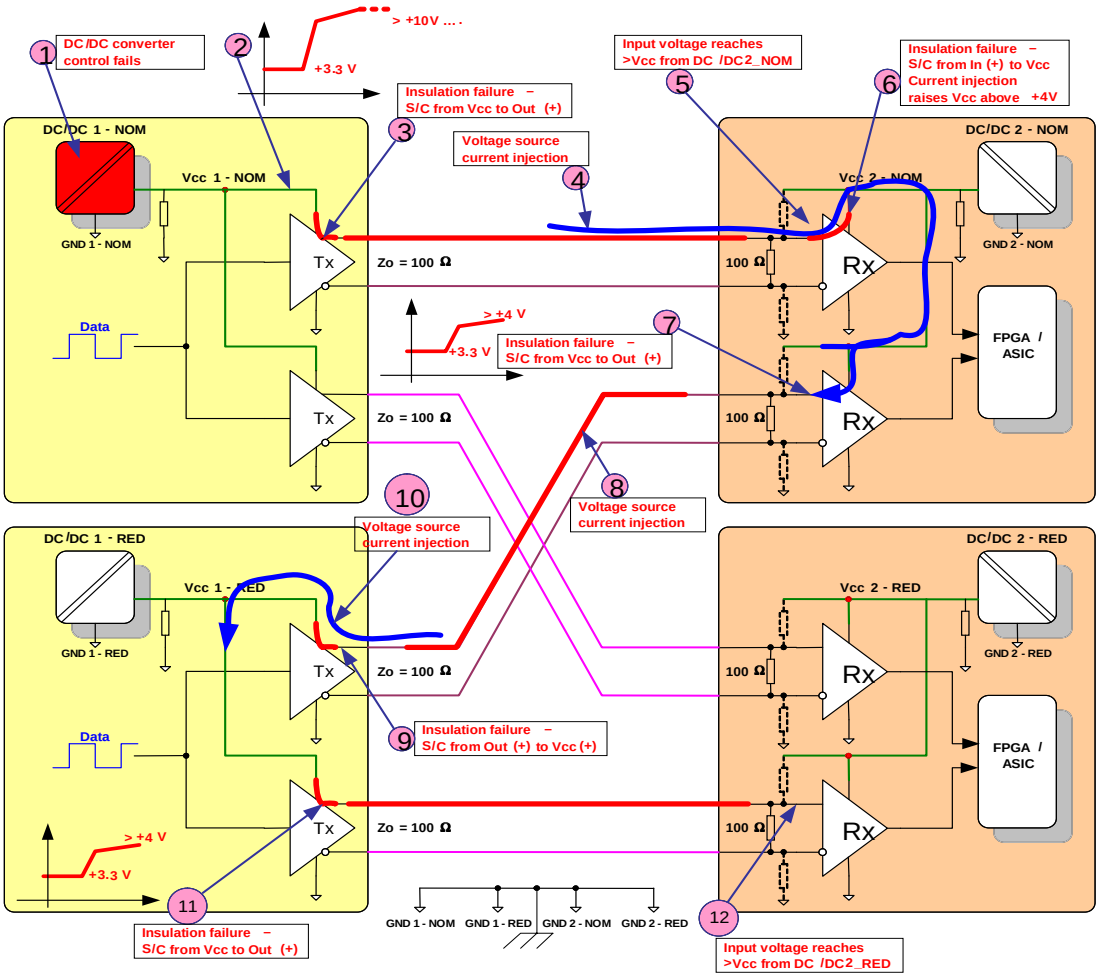


Figure 1 How a fault can cause damage to both nominal and redundant circuits.

5 LIMITING PROPAGATION

It may be possible to arrange input clamping or isolation circuits at the receiver inputs. The circuits used must not excessively load the signal lines (capacitively, for example) and as a result will tend to be small and thus have limited current and energy absorbing capability. It is essential that the fault current available from a compromised LVDS transmitter is limited to a safe value – at nowhere the level available from a supply bus.

We cannot rely on the LVDS transmitter chip to do this – it will have been damaged by this fault situation. We need to provide a reliable current limiting device that will not be easily damaged by excessive voltage – and if it is damaged will be sure to fail open-circuit. A simple resistor has the correct properties (dependant on technology, but thick-film SMD resistors and hole mounted metal-film resistors are accepted by most agencies as S/C free).

A driver circuit such as that shown in figure 2 where series resistors and, optionally, a parallel resistor are used with complementary logic-swing outputs can be used to produce LVDS signal levels. One suitable combination uses a 2.5V supply, 300Ω series resistors (R_s = 300) and no parallel resistor (R_p = ∞). It produces correct common mode and differential LVDS voltage levels. Its output resistance of 300Ω in each signal wire limits fault currents into a receiver – with a driver over-supply of

15V the current is limited to less than 50mA on each wire, a level that could be safely absorbed by receiver protection devices. This circuit is equivalent to a reasonable, but not perfect, current source (the higher the series resistor value, the better the current source). The required supply current is 3.5mA (9mW consumption), exactly the current needed to develop the required load voltage across a 100 Ω termination. Similar circuits may also be used with 3.3V or 5V supplies with the advantage of having higher valued series resistors to further limit fault currents, but the disadvantage of consuming more power.

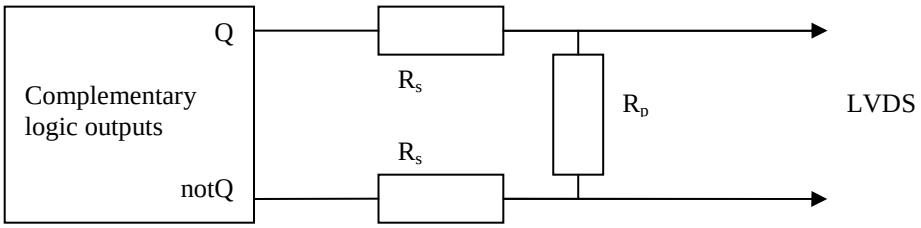


Figure 2 LVDS transmitter from complementary logic output and resistors

6 EMC

In addition to transferring data the signal lines must no be susceptible to interference from electro-magnetic fields.

Experience from IEEE1355 (which was developed into SpaceWire, keeping the same physical layer) reveals that differential drivers are often poor in this regard – their output characteristic is far from linear. It was common-mode noise on the signal lines was converted, by the drivers, into differential noise. Series resistance serves to linearise the outputs and improve EMC performance. A degree of source termination can also improve the situation – leading to consideration of re-scaling the series resistors in Figure 2 to allow the addition of a parallel resistor. The circuit used by Actel™ is very good in providing a near ideal source termination.

7 CONCLUSIONS

Far from being non-LVDS compliant, an LVDS output circuit containing real, non-integrated, resistors meets the LVDS standard, has a safe failure mode and is likely to improve EMC characteristics.

8 REFERENCES

1. The ECSS-E-50-12 Working Group, "**ECSS-E-50-12A 24 January 2003**, SpaceWire - Links, nodes, routers and networks", published by the ECSS Secretariat, ESA-ESTEC, Requirements & Standards Division, Noordwijk, The Netherlands

2. ANSI/TIA/EIA-644-A-2001 "**Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits**"

3. "**SpaceWire Link interface: LVDS, Power & Cross-strapping Aspects**" Sven Landstroem and Wahida Gasti Presentation at the SpaceWire working group meeting #11, Noordwijk, June 10&11 2008

Components 2

Wednesday 5 November
10:50 – 11:45

EVOLUTION AND APPLICATIONS OF SYSTEM ON A CHIP SPACEWIRE COMPONENTS FOR SPACEBORNE MISSIONS*

Session: Components 2

Long Paper

Joseph R. Marshall

BAE Systems, 9300 Wellington Road, Manassas, Virginia, 20110

E-mail: joe.marshall@baesystems.com

ABSTRACT

As reported at the first SpaceWire International Conference in 2007, BAE Systems in conjunction with NASA has created and space qualified a radiation hardened SpaceWire system on a chip ASIC containing a four port SpaceWire router, embedded microcontroller and memory, built-in memory, discrete, test and dual PCI interfaces. This ASIC has been inserted into a number of missions, including both the single board computer and the Mini-RF payload for the Lunar Reconnaissance Orbiter (LRO) NASA Mission planned for launch in early 2009. This versatile ASIC also supports creating routers scalable beyond its four ports. The SpaceWire core has been updated by NASA and was integrated with other ASIC functions into BAE Systems' latest RAD750™ bridge ASIC, the Golden Gate, shrinking the number of ASICs required for a Processor with SpaceWire by two-thirds.

This paper will describe the BAE Systems' SpaceWire ASIC's use within and between the Mini-RF system and the LRO Spacecraft as well as details about the board containing the ASIC and software supporting it mission. This paper will discuss the latest information on creating and programming larger than 4 port SpaceWire routers using this ASIC. The paper will describe the SpaceWire ASIC Evaluation Board that has been created for prototype development using the ASIC in a COTS chassis. Finally the paper will describe the integration of the SpaceWire core and other functional upgrades that were included in the Golden Gate ASIC, our latest bridge ASIC fabricated for standalone or processor bridge use in Spacecraft processing.

1. SPACEWIRE ASIC FEATURES REVIEW

The BAE Systems SpaceWire arose from a joint development between BAE Systems and NASA Goddard Space Flight Center (GSFC) merging the SpaceWire IP from GSFC into the Processor Bridge family from BAE Systems [1]. A block diagram of the SpaceWire ASIC is shown in Figure 1. The ASIC contains four SpaceWire ports, each of which may operate up to 264 MHz through integrated LVDS interfaces, and a

* Approved for Public Release: N68936-05-C-0066 on 10-10-2008.

7 port router between those four ports along with two external data interface ports and a router maintenance port. The external ports connect into an on-chip bus (shown as the blue bar at the left side of Figure 1) that provides a cross-bar switch between the various cores attached to it. Thus SpaceWire sourced or destined data may be connected to one of two 32 bit 33 MHz PCI 2.2 compliant busses, a 2 GB memory space for external memory, 32 KB of internal SRAM, a DMA controller and an embedded microcontroller[3] that may be used to manage or direct the ASIC traffic. Slower speed connections may be made with discrete I/O, a UART interface, a JTAG Interface as well as several clocks and timing circuits.

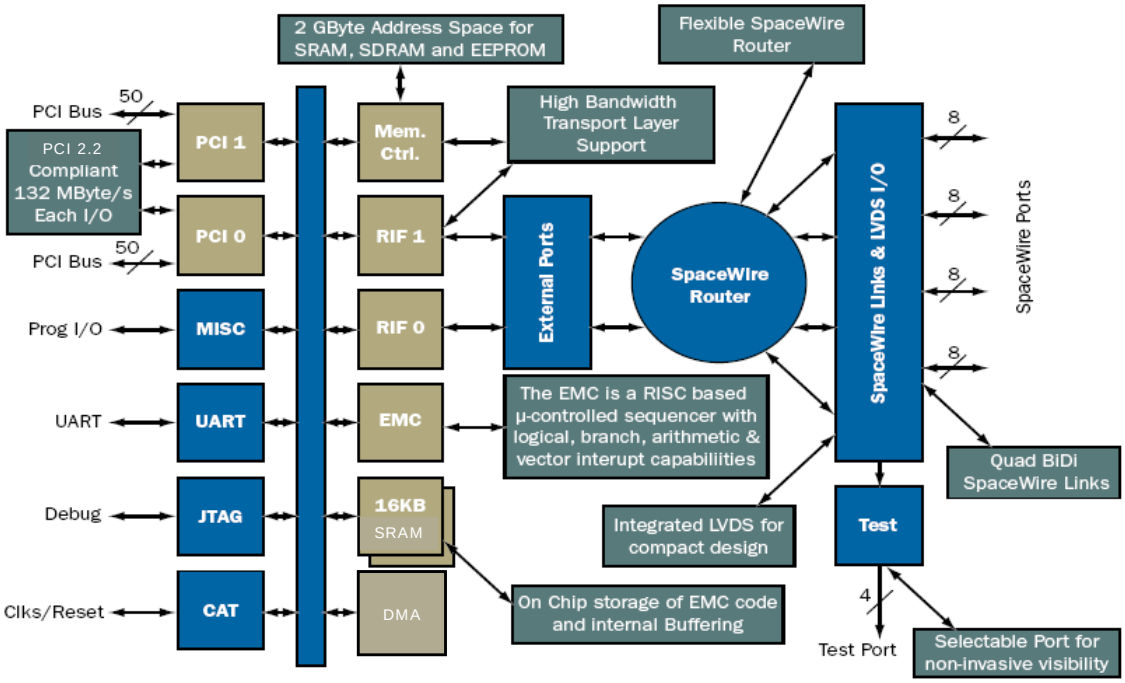


Figure 1. SpaceWire ASIC Block Diagram

The SpaceWire ASIC operates over the full military temperature range and is powered by 2.5V supply for its core and 3.3V supply for its I/O. It is packaged in a 625 pin Column Grid Array package. It is rated for operation over 200 Krad (Si) total ionizing dose, while experiencing single event upsets under 10^{-9} upsets / bit-day. Thus the SpaceWire ASIC is a complete bridging system on a single chip and is being applied to several spacecraft applications including the Geostationary Operational Environmental Satellite (GOES-R) and the NASA Lunar Reconnaissance Orbiter (LRO).

2. SpaceWire Use within LRO

The LRO is a satellite designed to map the terrain in intimate detail and will launch in early 2009. SpaceWire was selected as the main data interface throughout the spacecraft [2] and is visible in the spacecraft functional block diagram shown in Figure 2. Note the SpaceWire links connecting the RAD750 SBC to the HK/IO, Ka-Comm and S-Comm units as well as the link from the HK/IO to the Mini-RF Payload.

The RAD750 SBC was described in detail in [2]. We will now discuss the use of the SpaceWire ASIC within the Mini-RF payload.

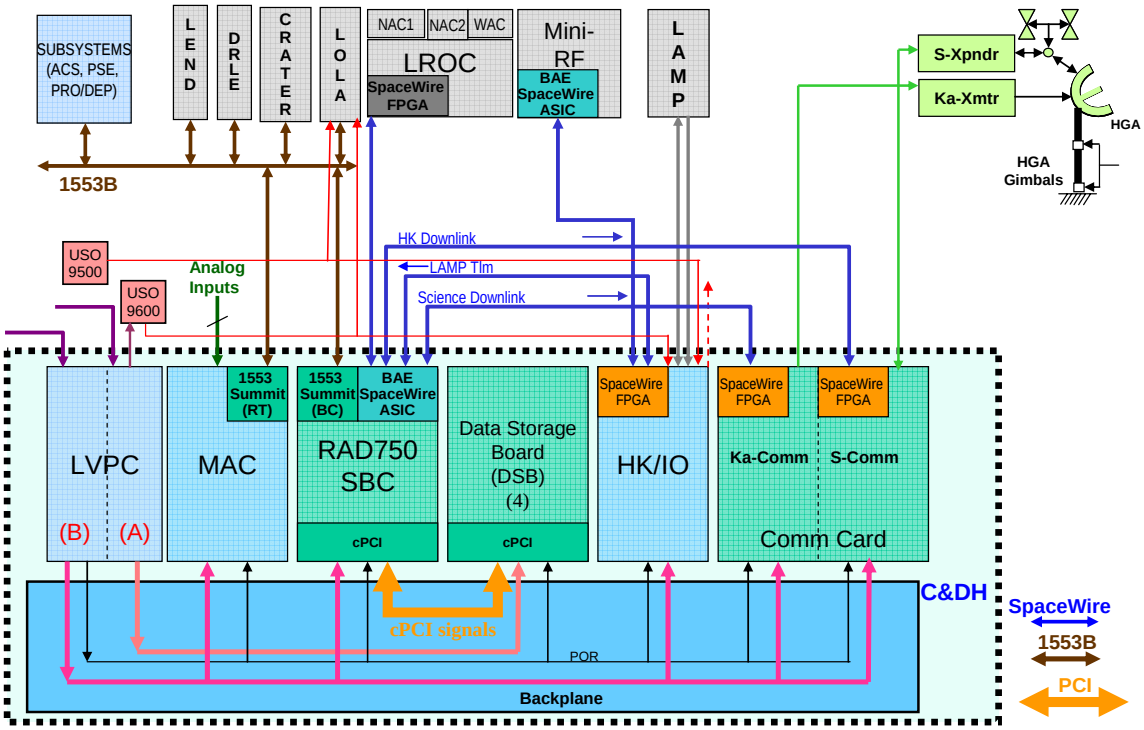
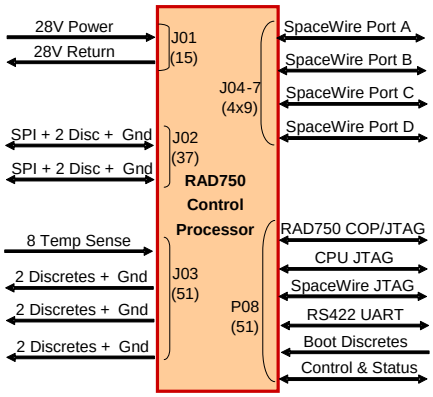


Figure 2. LRO C&DH Architecture

3. MINI-RF PROCESSOR SLICE

The Mini-RF Payload is a radar system used to search for ice in the craters at the poles of the moon. Controlling the Mini-RF Payload is a RAD750 Control Processor Slice. Processor connections to the Control Processor Slice of the Mini-RF Payload are shown in Figure 3. The Control Processor provides the direct link to the Bus Electronics via the HK/IO module and receives commands and reports status over that interface to the RAD750 SBC. The Control Processor manages the Mini-RF payload and uses SpaceWire links to control high performance elements in the payload. High speed data is also routed back to the HK/IO unit for storage and transmission to earth. Slower speed interfaces (Serial Peripheral Interconnect or SPI) are used for controlling the lower performance functions in the payload. A block diagram of the processor I/O is shown below. Connector designations and pin counts are shown.



The Mini-RF RAD750 Control Processor Slice is a self-contained enclosed single board computer that runs directly off of spacecraft 28V power. A block diagram of the Control Processor is shown in Figure 4. Colors are used to show voltage regions. Power is converted and filtered from 28V down to +12, +5V and +3.3V. The 3.3V is further regulated to 2.5V

for use by the various devices needing that core voltage. The design of the power conversion for the Control Processor Slice was especially challenging. Initially we selected some promising COTS converters and point of load devices. Radiation testing of these devices uncovered single event latch-up problems which disqualified their usage. A set of less efficient but radiation hardened devices were then selected.

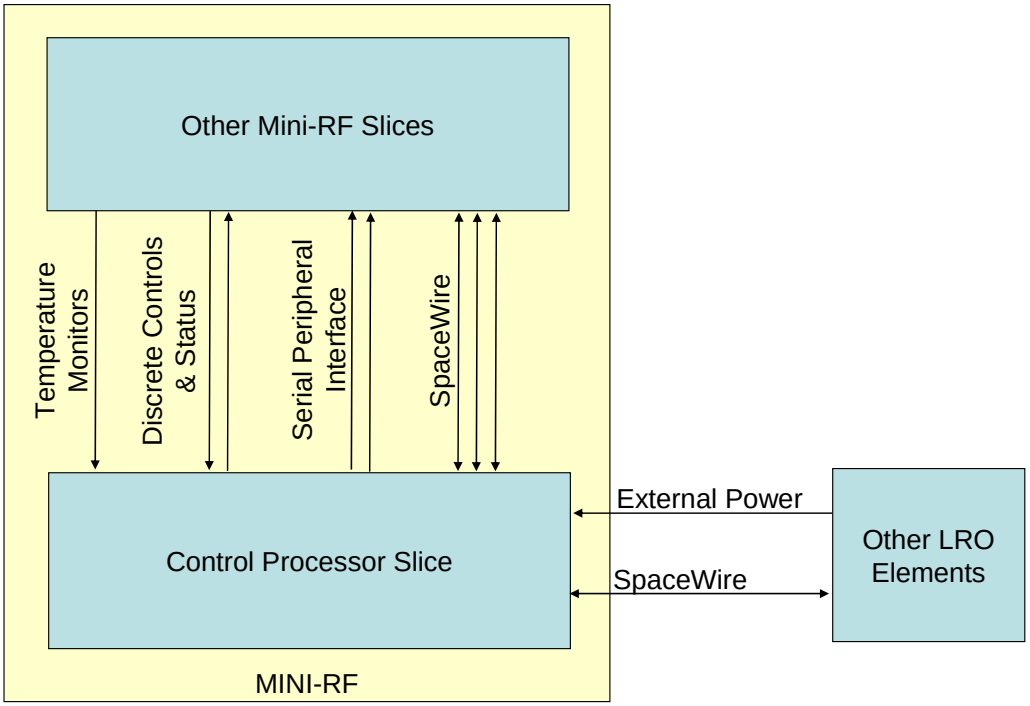


Figure 3. Mini-RF Processor Interconnect Block Diagram

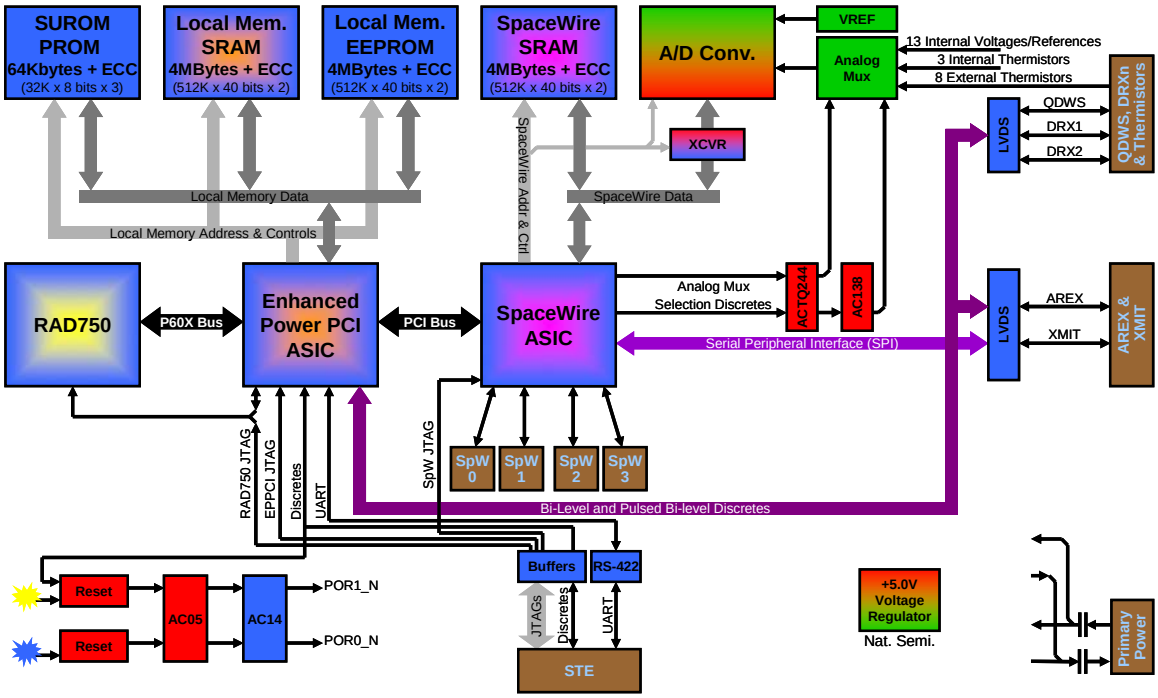


Figure 4. Mini-RF RAD750 Control Processor Block Diagram

At the left center of Figure 4 is the RAD750 Processor running at 132 MHz[5]. This is connected to a processor bridge chip that interfaces to the Processor's onboard memory and to the SpaceWire ASIC. 64KB of PROM, 4 MB of EEPROM and 4 MB

of SRAM provide a full array of memory for the controller software to manage the Mini-RF payload. The Enhanced Power PCI ASIC also provides connections to the boards software test interfaces, a UART and JTAG as well as various configuration discretes. The SpaceWire ASIC provides its four routed SpaceWire ports for external control and data movement. It also utilized the EMC in the ASIC to run a software controlled SPI interface to the slower elements of the payload. The SpaceWire ASIC's memory interface is used to provide 4 MB of SRAM for message buffering and for spare storage that the RAD750 may utilize. This interface was also used to connect an Analog to Digital Converter to several voltage measuring circuits and Thermistors. Thus, the Mini-RF RAD750 Controller took full advantage of the SpaceWire ASIC's many interfaces and capabilities to avoid the need for the typical glue logic FPGA so common on most processing and interface applications. By using all off the shelf components, this card was able to be developed, brought up and integrated quicker than if it had included an FPGA design.

The RAD750 Controller Printed Wiring Board was mounted in a standalone slice with all connectors brought to D and D sub miniature cables. Thus there were no bus interfaces; however, SpaceWire was a natural for its point to point connections with other spacecraft busses. 3D representations of the design are shown in Figure 5. The SpaceWire connectors are on the top of the right drawing. Note the tabs that were used to join the slice to the deck or slice below it and the slice above it. Getting these to be accessible for mounting while also being able to connect the many cables was a particularly challenging problem.

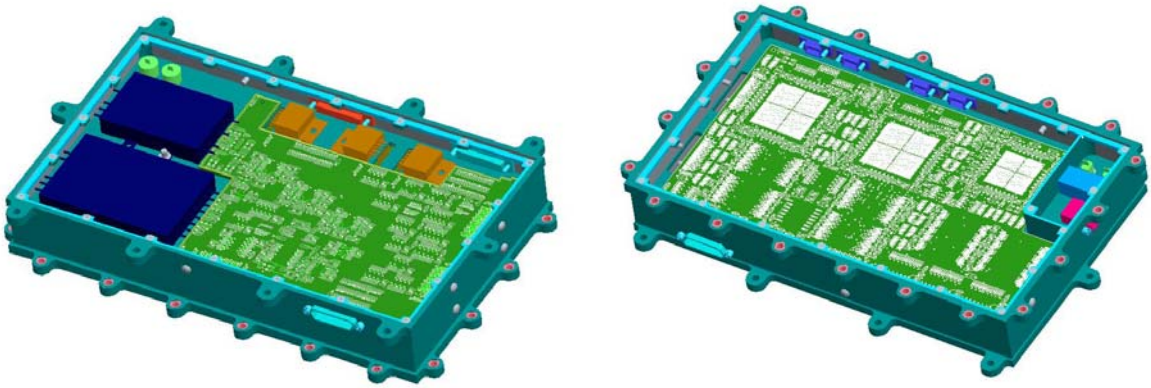
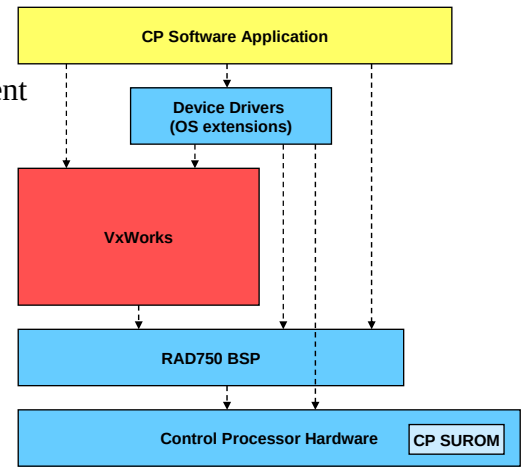


Figure 5. Controller Slice Isometric Physical Views with Covers Removed – Top and Bottom

4. MINI-RF PROCESSOR SOFTWARE

The RAD750 Controller relies on PROM-stored resident software for booting and EEPROM-stored resident software for the Operational Code and its operating system support. The internal structure of the software support layers is shown in the diagram to the right. SUROM is resident in the PROM on Controller Slice. Wind River's VXWorks is used as the real time operating system supported by board support and device driver extensions for this particular design. The Mini-RF Software Application



calls various routines from these in order to access the underlying hardware including extensions to use all the functions attached to the SpaceWire ASIC. Software routines were created and made part of the support software in order to sample telemetry and to send SPI messages to the other slices in the system as well as access any discrete signals from other modules. The combined code module was stored in the EEPROM on Control Processor Slice.

The Mini-RF Control Processor was designed to use off the shelf components and not stress the interfaces or processor. Because of board clock selections, the SpaceWire links may be configured up to 200 MHz each on this design but only were required to provide data at 40 MHz for the LRO mission. The Control Processor has been built, qualified, tested and integrated into the Mini-RF Payload, part of the 2009 LRO mission.

5. SPACEWIRE EVALUATION BOARD

The 6U-220 LRO SBC [2] is one of the most complex RAD750 single board computers built to date. A subset of this design was mapped onto a smaller 6U-160 board to make the SpaceWire ASIC available for prototyping usage in COTS CompactPCI backplanes. A block diagram of the evaluation board is shown in Figure 6. Note that there is no separate processor on this card; however the embedded microcontroller (EMC) in the SpaceWire ASIC may be utilized as a card controller and 256KB of ECC-protected EEPROM is provided to store EMC code and variables. 4 MB of SRAM is provided to store messages or as a buffer for the EMC.

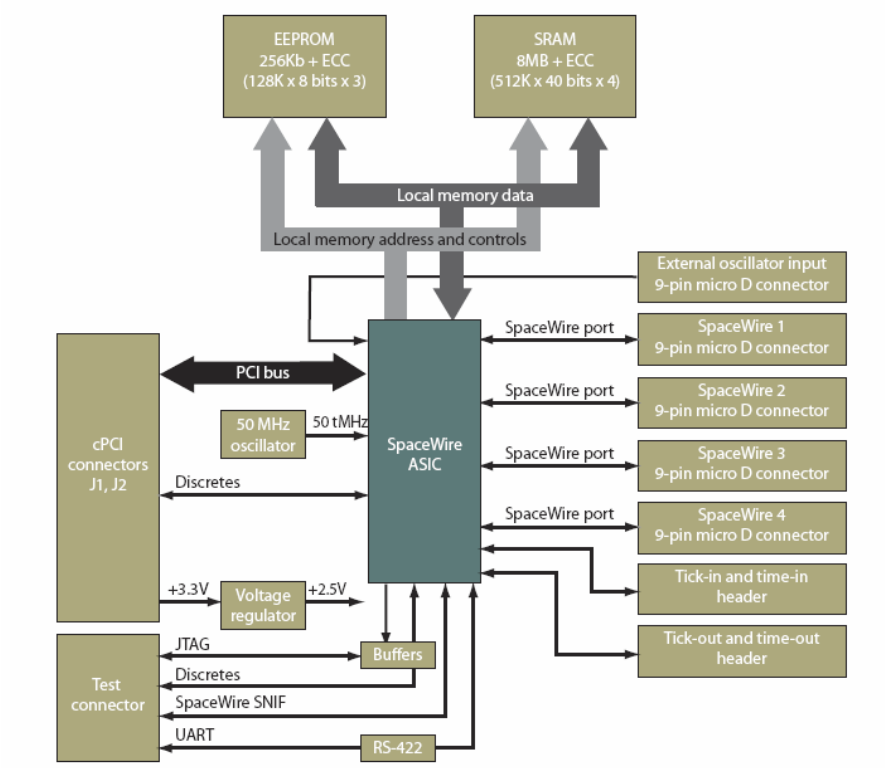


Figure 6. SpaceWire Evaluation Board Block Diagram

The SpaceWire Evaluation board has four SpaceWire ports and the integrated router and a 33 MHz 32 bit PCI 2.2 Bus interface for connecting to other boards as well as UART and JTAG test interfaces. The board runs solely off of a 3.3V supply.

The board has been built and tested with COTS SpaceWire test equipment. Two photos of the tested evaluation board is shown in . From a component point of view, this could fit on a 3U card; however the engineering required to incorporate all the front panel connectors in the small space was prohibitive for an evaluation card.

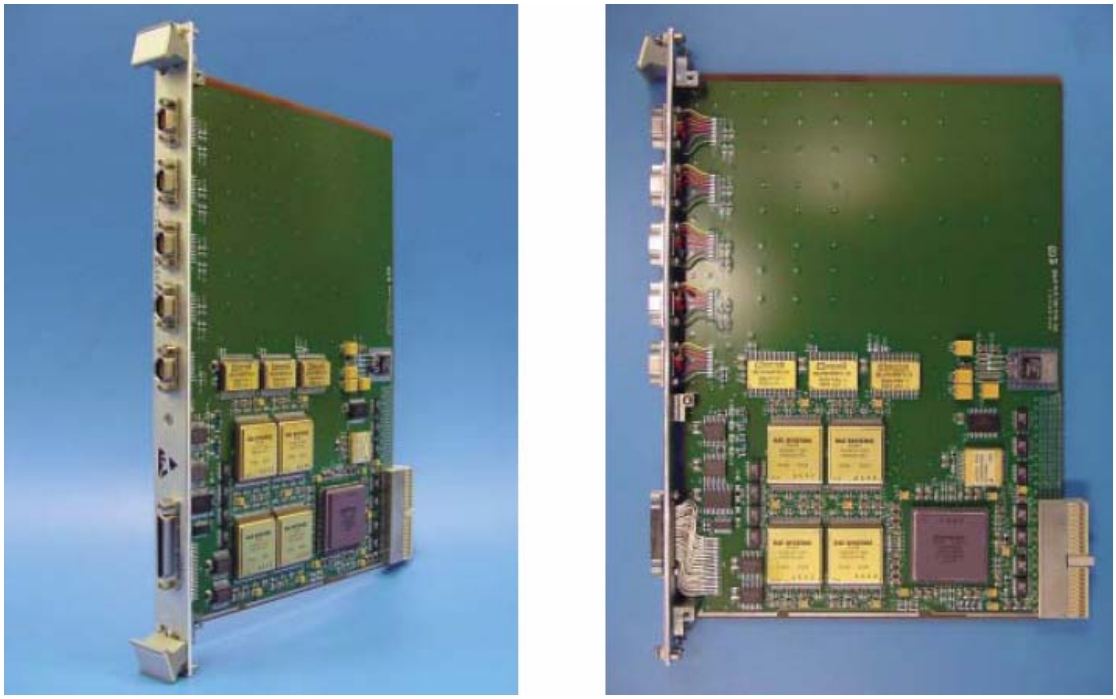


Figure 7. SpaceWire ASIC Evaluation Board Photos

6. MULTI-PORT ROUTERS

The SpaceWire ASIC Evaluation Boards may be grouped together on a CompactPCI backplane to create six or more ports of a multi-port router [1] as shown in . These could either be controlled by one of the embedded EMCs[3] or some other COTS processing card, such as the BAE RAD750 Commercial Evaluation card, could be used. The primary connection between the boards is the PCI Bus. Packets arriving on one SpaceWire ASIC may be mapped off the ASIC to the PCI Bus and then sent to the proper addressed other SpaceWire ASIC where the packet may be recreated as sent out on another SpaceWire Port. Analysis of these interfaces show that the 33 MHz 32 bit PCI bus has sufficient bandwidth (1056 Mbps) to handle small numbers of simultaneous high speed packets (100 to 250 Mbps) or larger numbers of lower speed packets (under 100 Mbps). If after analyzing the traffic more simultaneous switching bandwidth is needed, one or two of the SpaceWire ports on each ASIC may be connected to others so that some or the highest priority high speed packets would not be required to leave the SpaceWire fabric. For example, a group of four SpaceWire evaluation boards could provide anywhere from an 8 to 16 port router depending the bandwidth of the using nodes.

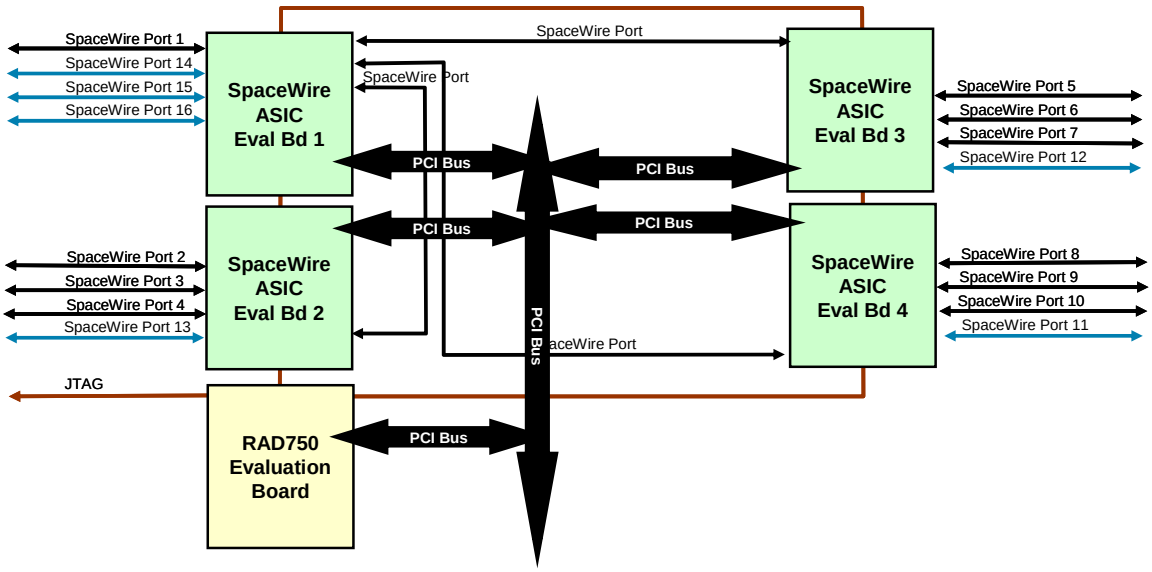


Figure 8. Multi-Port Router Using SpaceWire ASIC Evaluation Boards

7. GOLDEN GATE ASIC

BAE is fabricating its latest bridge ASIC, the Golden Gate ASIC. This device marries three of the most important interface devices onto a single radiation-hardened silicon ASIC, namely the Enhanced Power PCI Processor Bridge, the SpaceWire Bridge and 1553 Interface Bridge. It expands the size and speed of the PCI interfaces and provides a generic FIFO interface. Additionally, the EMC, DMA, discretes, internal memory and clocks are upgraded. A block diagram of the Golden Gate ASIC is shown in Figure 9.

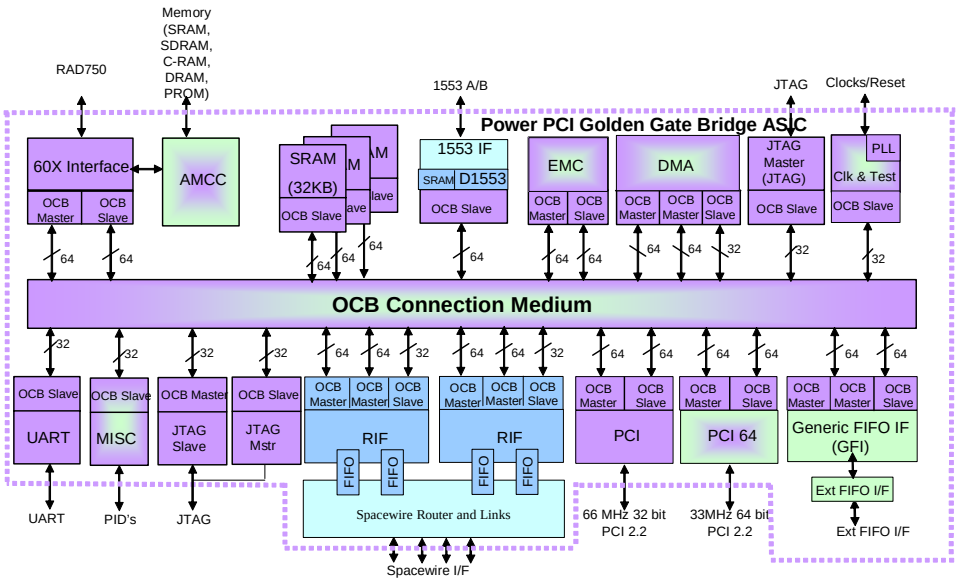


Figure 9. Golden Gate ASIC Block Diagram. Purple blocks represent reused cores.

This device utilizes an updated version of the NASA GSFC SpaceWire IP and a COTS 1553 interface IP core integrated into the On-Chip Bus. This device will result in significant savings in board area, power and higher performance processing when used on future processing applications.

8. SUMMARY

The BAE Systems SpaceWire ASIC has now been used in several space applications and will be a principal interface device in the upcoming NASA LRO Mission for command and data handling and payload control applications. Many of its system-on-a-chip capabilities were utilized on the Mini-RF Control Processor resulting in a small efficient slice that may find use in future systems. BAE has captured the essence of the LRO and Mini-RF Single board computers and distilled them into a SpaceWire Evaluation board for lab prototyping of 4 port or larger router configurations. The new Golden Gate ASIC will provide up to a 3x savings in real estate, improved power and performance and an updated SpaceWire interface. This along with the RAD6000 Microcontroller [4] mentioned at last year's conference provide application users two different single chip solutions marrying SpaceWire with processing applications spanning small instruments to multi-processor payloads.

9. REFERENCES

[1] Marshall, Joseph R. & Berger, Richard W., "A One Chip Hardened Solution for High Speed SpaceWire System Implementations", *2007 International SpaceWire Conference*, Dundee, Scotland, September 2007.

[2] Berger, Richard W. et. al., "RAD750 SpaceWire-Enabled Flight Computer for Lunar Reconnaissance Orbiter", *2007 International SpaceWire Conference*, Dundee, Scotland, September 2007.

[3] Marshall, Joseph R. & Robertson, Jeffrey, "An Embedded Microcontroller for Spacecraft Applications", *IEEE Aerospace Conference 2006 Proceedings*, Big Sky, Montana, March 2006.

[4] Berger, Richard W. et. al., "A System-On-Chip Radiation Hardened Microcontroller ASIC With Embedded SpaceWire Router", *2007 International SpaceWire Conference*, Dundee, Scotland, September 2007.

[5] Richard W. Berger et. al., "The RAD750 - A Radiation Hardened PowerPC Processor for High Performance Spaceborne Applications", *IEEE Aerospace Conference 2001*, April 2001.

EVALUATION AND ANALYSIS OF CONNECTOR PERFORMANCE FOR THE SPACEWIRE BACK PLANE

Session: SpaceWire Components

Short Paper

Koji Shibuya, Keitaro Yamagishi , Hideyuki Oh-hashi, and Seiichi Saito

Mitsubishi Electric Corp., 5-1-1 Ofuna, Kamakura, Kanagawa, 247-8501, JAPAN

Masaharu Nomachi

Osaka University, 1-1 Machikane-yama, Toyonaka, Osaka, 560-0043, JAPAN

E-mail: Shibuya.Koji@ak.MitsubishiElectric.co.jp, Yamagishi.Keitaro@bk.MitsubishiElectric.co.jp,Ohashi.Hideyuki@dr.MitsubishiElectric.co.jp, Saito.Seiichi@eb.MitsubishiElectric.co.jp,nomachi@lns.sci.osaka-u.ac.jp

ABSTRACT

SpaceWire interconnection using a backplane which makes a system compact has been investigated. In the backplane interconnection system, high-speed signals are connected between daughter boards via connectors and a backplane. The connectors used to interface the daughter boards and the backplane are key components because the impedance mismatching and the transmission loss in the connector might cause serious waveform distortions of high-speeds signal. In this paper, connectors for the backplane, which have been selected by the space qualified design and compactness, are modelled for various cases of signal and ground pin assignment by the measurement of S-parameters. And, then, the evaluation and the analysis are carried out by the SPICE simulation of the signal transmission between daughter boards via the connectors on the backplane. It was found that the crosstalk is generated at the connector and the characteristic impedance of the connector changes according to the pin assignment. Further, it was confirmed that the investigated connector can be used in backplane systems up to 3Gbps when the pin assignment is designed suitable.

1 CONNECTOR EVALUATION BOARD

For on-board equipment, higher-speed signal interconnections are required due to increasing data rates between the boards in the equipment. High-speed SpaceWire interconnection using a backplane which makes a system compact and high-performance has been investigated. The connectors on the backplane are key components because the impedance mismatching and the transmission loss of the connector might cause serious waveform distortions of high-speed signals.

Because no high speed backplane connector with space qualified design was found, a backplane connector without guarantee of high frequency performance was selected from the point of view of space use and compactness for SpaceWire backplane

interconnection. In order to evaluate the connector, a pair of printed circuit boards (PCBs) has been developed, one PCB corresponds to a backplane and another PCB corresponds to a daughter board. The backplane connector has four row of conducting pins, and consists of a right angle plug and a receptacle. The plug is mounted on the daughter board and the receptacle is mounted on the backboard, respectively. The signal pins of backplane connector are connected to the SMA connectors on the PCB edges through micro-strip lines on PCBs. Fig.1 shows a photograph of the PCBs for evaluation of the backplane connector.

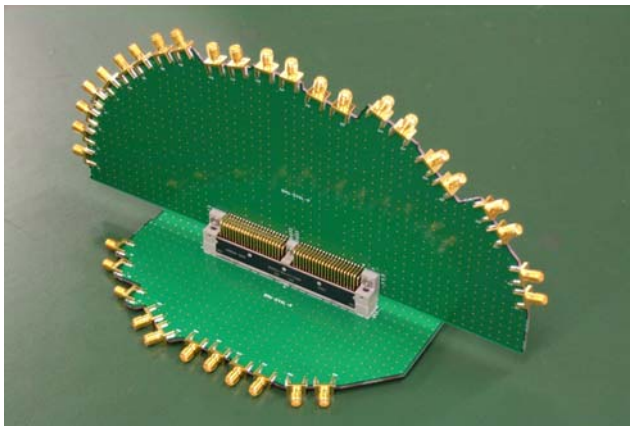


Fig.1: PCBs for Evaluation of a backplane connector

The pin assignment of the connector can cause the changes of the transmission characteristics and the crosstalk. Eight differential mode signal paths with different pin assignment of signal and ground are arranged within the backplane connector. Fig.2 shows the variations of the pin assignment. Fig.3 shows another differential mode signal path within the same connector surrounded by five single pin signal paths. The arrangement of Fig.3 is used for the evaluation of crosstalk (the differential path is a victim and single pin of a, b, c, d, and e are aggressive).

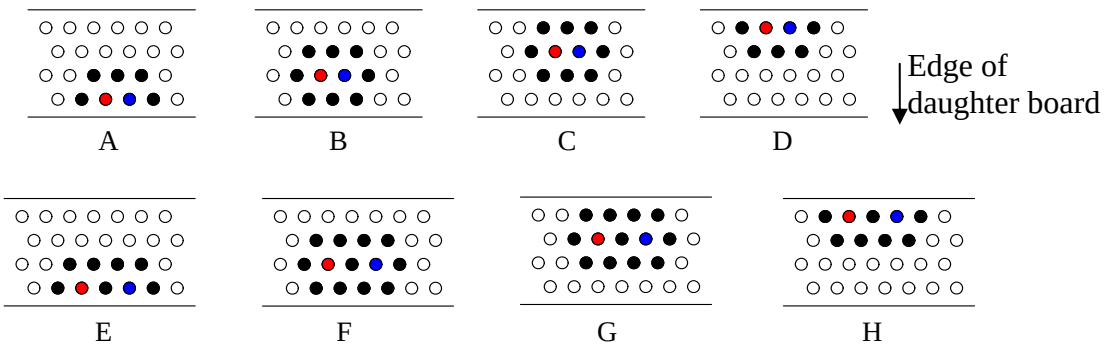


Fig.2: Pin assignment for the transmission characteristics evaluation.
(Red and blue circles are differential pairs. Black and white circles are GND pins.)

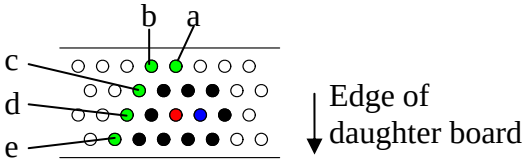
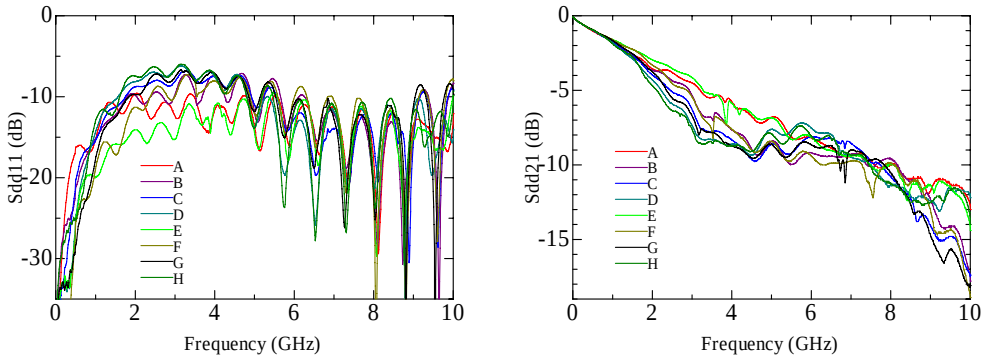


Fig.3: Pin assignment for the crosstalk evaluation.
(Red and blue circles are differential pairs. Black and white circles are GND pins.
And, Green circles are the aggressive signal pins.)

2 MEASUREMENT RESULT OF FREQUENCY CHARACTERISTIC

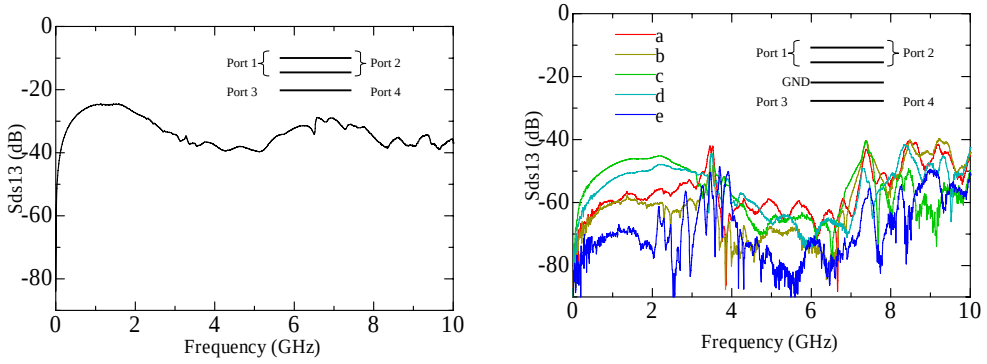
The evaluation PCBs with the mated backplane connector are measured with a four port network analyzer, and mixed-mode S parameters of differential mode are obtained by converting the standard S-parameters [1][2]. Fig.4 shows the measurement results of the differential mode return loss (Sdd11) and insertion loss (Sdd21) including the characteristics of micro-strip lines on PCBs and SMA connector. From Fig.4, it is found that the differences of S-parameters between different pin assignments become remarkable in frequency range higher than 1GHz. Moreover, the pin assignments using the higher row pins (far from edge of daughter board) for signals cause characteristic degradation. Also, the pin assignment with ground pin between two differential signal pins leads to better characteristic than that without ground pin between two signal pins.

Fig.5 shows the measurement results of the near-end crosstalk (Sds13). Fig.5(a) shows the crosstalk using a pin assignment without ground pins around victim (differential signal pins are a and b, and aggressive is c in Fig.3), and Fig.5(b) shows the crosstalk with ground pins around the victim. Crosstalk can be dramatically reduced by ground pins around victim.



(a) Differential return loss (Sdd11) (b) Differential Insertion loss (Sdd21)

Fig.4: Measurement results of the S-parameters of the connector



(a) No GND pins around the victim (b) GND pins existing around the victim

Fig.5: Measurement results of near-end crosstalk.

3 EYE-PATTERN WAVEFORM ANALYSIS

In order to evaluate performance of high speed signal transmission system, an eye-pattern waveform analysis is used because the results can directly compared with eye-

mask specifications of SerDes receivers. The eye-pattern waveform can be obtained by the analysis of SPICE simulation using the analytical model such as the signal source, connectors, the transmission line of the PCB traces and through-hole vias. Fig.6 shows the simulated eye patterns of the SpaceWire backplane system with 50 cm total trace and 3.125Gbps data rate using the model of the evaluated backplane connector. In Fig.6, all eye-pattern waveforms are good for transmission because they clear the eye-mask specification. In detail, the waveform is a little different according to the selection of pin assignment of the backplane connector.

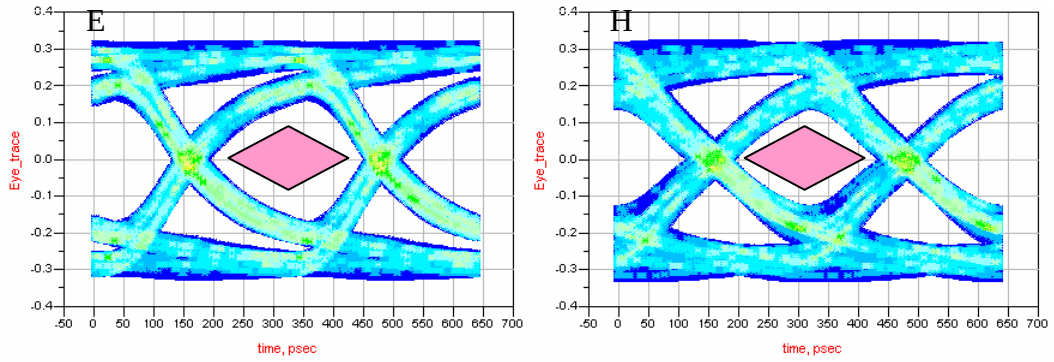


Fig.6: Eye patterns of the analytical result.

4 CONCLUSION

A SpaceWire interconnection system via the backplane using the connector selected by the space qualified design and the compactness was evaluated by the crosstalk measurement and the eye-pattern waveform obtained by the SPICE simulation using the analytical model of the system.
The results of the evaluation are in the followings:

(1) The backplane connector was modelled for various cases of the signal and ground pin assignment by the measurement of the S-parameters.
(2) It was found by the measurement that the pin assignment of the ground pin around the differential pair is effective to reduce the crosstalk from the other signals.
(3) The eye-pattern waveforms were shown by the simulation for the backplane system. It was found that the 3Gbps backplane transmission is possible using the connector selected by the space qualified design.
(4) Totally, it was confirmed by the crosstalk measurement and the waveform simulation that the connector selected by the space qualified design can use for the system of over 1Gbps high speed signal transmission via the backplane by suitable pin assignment design.

5 REFERENCES

1. K. Kurokawa, “Power waves and the scattering matrix”, IEEE Trans. Microwave Theory Tech, vol.13, 1965.

2. David E. Bockelman, William R. Eisenstadt, “Combined Differential and Common-Mode Scattering Parameters”, IEEE Trans. Microwave Theory Tech, vol.43, 1995.

SPW-10X ROUTER ASIC TESTING AND PERFORMANCE

Session: SpaceWire Components

Short Paper

Steve Parkes[1], Chris McClements[1], Gerald Kempf [2], Christian Gleiss[2],
Stephan Fischer [3], Pierre Fabry [4]

[1]*School of Computing, University of Dundee, Dundee, DD1 4HN, Scotland, UK.*

Email: sparkes@computing.dundee.ac.uk.

[2]*Austrian Aerospace*

[3]*Astrium GmbH*

[4]*European Space Agency*

ABSTRACT

The SpW-10X SpaceWire routing switch ASIC is a radiation tolerant device with eight SpaceWire ports and two parallel ports. It supports both path and logical addressing, includes various failure detection and power saving features, operates at up to 200 Mbits/s and switches packets in less than 0.5 µs. LVDS drivers and receivers are included on-chip to save board area, mass and power.

The SpW-10X device is now being sold by Atmel as the AT7910E and is being designed into space missions. This paper introduces the SpW-10X device, describes how it was tested and summarises its performance characteristics.

1 SPW-10X OVERVIEW

The SpaceWire routing switch is a key element in any SpaceWire network [1].

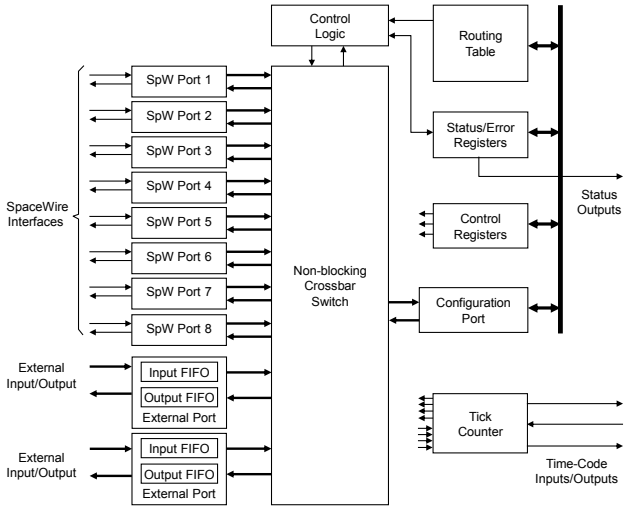


Figure 1 SpW-10X SpaceWire Routing Switch

The SpW-10X routing switch IP core was designed for ESA by University of Dundee and Austrian Aerospace, validated by EADS Astrium GmbH and manufactured by Atmel. It is now available as a radiation tolerant ASIC from Atmel (part no. AT7910E) [2]. The architecture of the SpW-10X is illustrated in Figure 1.

It has the following main features:

- Eight SpaceWire ports.
- Two external parallel ports, each comprising an input FIFO and an output FIFO.
- A non-blocking crossbar switch connecting any input port to any output port.
- An internal configuration port accessible via the crossbar switch from the external parallel ports or the SpaceWire ports.
- A routing table accessible via the configuration port which holds the logical address to output port mapping.
- Logic to control the operation of the switch, performing arbitration and group adaptive routing.
- Control registers that can be written and read by the configuration port using the RMAP protocol [3] and which hold control information e.g. link operating speed.
- An external time-code interface used for sending and receiving time or synchronisation information over a SpaceWire network.
- Watchdog timers on all ports.
- External status/error signals.
- Supports both path and logical addressing.
- Provides priority routing based on logical address.
- Supports group adaptive routing.
- Includes several power saving features to reduce overall power consumption.
- Provides ‘disable on silence’ and ‘start on request’ capability.
- Implemented in 0.35µm MH1RT radiation tolerant technology from ATMEL.
- Radiation tolerant: total dose 300Krad(Si), no latchup up to 70 MeV/mg/cm^2.

2 SpW-10X Verification, Validation and Characterisation

The SpW-10X device has been tested in several different ways:

- During design by University of Dundee a VHDL tests bench was used for initial testing.
- An independent test bench was developed by Austrian Aerospace providing extensive tests and identifying several issues with the initial VHDL code.
- The Router IP was implemented in several STAR-Dundee devices and has been widely used by many organisations.
- The SpW-10X device was implemented in a Xilinx FPGA with the design kept as close as possible to the final VHDL code used for the ASIC design. This SpW-10X FPGA was extensively tested by EADS Astrium GmbH.

- A second SpW-10X FPGA device was implemented as a mezzanine board in preparation for final ASIC prototype testing.
- A similar board was designed to carry the SpW-10X ASIC. As soon as the SpW-10X device was available it was mounted on this test board.

The prototype ASIC devices were delivered in November 2007 and an extensive test campaign was then performed to fully characterise the devices:

- Functional validation by EADS Astrium,
- Performance testing by University of Dundee,
- Characterisation by Austrian Aerospace.

3 VERIFICATION

The verification of the SpW-10X is based on the following cornerstones:

- VHDL Testbench with self-checking scenarios
- RTL and Netlist (FPGA and ASIC) verified with the test bench
- Code coverage checked for RTL simulations
- Analysis of requirements which were not possible to simulate
- Timing of ASIC verified with static timing analysis

The verification of the SpW-10X started in parallel to the VHDL development of the SpaceWire Router and was used continuously to debug the design. The formal verification was run with the final FPGA RTL design, FPGA netlist, ASIC RTL design and ASIC netlist. The results are:

- All functions have been checked and show that there are no errors
- Simulation and analysis have been used for the verification
- FPGA and ASIC netlist verification showed that the implementation is correct

4 SPW-10X VALIDATION

Validation was performed against the SpW-10X Specification checking first the FPGA implementation and then the ASIC against each of the requirements in the specification. All the validation tests were performed successfully. There were two clarifications required during FPGA testing: path addressing with different priority levels was not required and the exact value for the Output Port Timeout Interval was specified. During ASIC testing a clarification was made by Atmel that the LVDS I/O cells perform disable rather than tri-state.

5 SPW-10X NETWORK TESTING

Extensive testing of the SpW-10X ASIC in various network configurations was carried out by University of Dundee. No anomalies were found in the ASIC device.

6 SPW-10X CHARACTERISATION

Austrian Aerospace performed extensive characterisation of the SpW-10X ASIC device including supply voltage, temperature and power consumption measurements. A summary of the results of these tests are provided below:

- SpW data rate configurable up to 200Mbps
- Single supply voltage of 3.3V (3.0 to 3.6V)
- Temperature:
  - Operational ambient temperature -55°C to +125°C
  - Maximum junction temperature +175°C
  - Maximum lead temperature (soldering 10 sec) +300°C
  - Storage temperature -65°C to +150°C
- Radiation
  - Total dose 300Krad(Si)
  - No latchup up to 70 MeV/mg/cm^2
  - Package MQFP 196 with 25 mil pin spacing
- Power consumption (max):
  - Static Pst: 1W
  - OFF condition Poff: 1.6W
  - Total operational all SpW IF active Pop:
    3.7W @ 200Mbps, 3.0W @ 100Mbps, 2.4W @ 10Mbps
  - Deactivated (Clk and LVDS buffer) SpW link: reduction of power by (Pop - Poff) × 0.1 + 0.06; E.g. with two SpW links deactivated operating at 200Mbps the power consumption is 3.16W
  - Data flow has very little influence on power consumption
  - For lower supply voltage (<3.6V):
    resistive model can be used, e.g. 69.4% of power at 3.0V

7 SUMMARY

The SpW-10X ASIC operates at up to 200 Mbits/s on all links. Packet switching times are approximately 0.5 µs with a maximum latency of 0.55 µs from a SpaceWire input, through the switch, to a SpaceWire output. All goals for the performance of the ASIC were achieved.

Comprehensive documentation for the SpW-10X device is available including a summary data sheet [2] and a comprehensive Users Manual [4] which allows users to understand and apply the extensive set of features in the device for their particular mission. Front line support is provided by Atmel and detailed technical support by STAR-Dundee Ltd [5].

8 ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of ESA for supporting the design and development of the SpW-10X router ASIC which was done under ESA contract number 15803/01/NL/JA.

9 REFERENCES

[1] ECSS, “SpaceWire: Links, nodes, routers and networks”, ECSS-E50-12A, January 2003

[2] Atmel, “SpW-10X SpaceWire Router, AT7910E Data Sheet”, August 2008, available from http://www.atmel.com/dyn/resources/prod_documents/doc7796.pdf

[3] ECSS “SpaceWire Protocols”, ECSS-E-ST-50-11C, Draft 1.3, July 2008

[4] C. McClements, S. Parkes, G. Kempf, “SpW-10X SpaceWire Router User Manual”, Issue 3.4, July 2008 available from http://www.atmel.com/dyn/products/product_card.asp?part_id=4339

[5] http://www.star-dundee.com

Networks & Protocols 1

Wednesday 5 November
11:45 – 12:10

SPACEWIRE PLUG-AND-PLAY: A ROADMAP

Session: Networks and Protocols

Long Paper

Peter Mendham, Albert Ferrer Florit, Steve Parkes

Space Technology Centre, University of Dundee, Dundee, DD1 4HN, UK

E-mail: [petermendham, aferrer, sparkes]@computing.dundee.ac.uk

ABSTRACT

SpaceWire Plug-and-Play has emerged as a topic of widespread interest, with organisations from across the world requiring some plug-and-play features in a wide variety of contexts. The plug-and-play sub-group of the SpaceWire working group developed the principles of a plug-and-play protocol for SpaceWire, with the aim of producing a concrete implementation. The authors of this paper further developed the concept by shifting the implementation to RMAP, providing the ability to leverage existing IP and experience, and simplifying the proposals

The paper begins by reviewing the history of SpaceWire plug-and-play, the current position of the fledgling standard, and its implementation both on new and existing hardware. A key development affecting the future of plug-and-play is the forthcoming SpaceWire-RT protocol. This paper presents a roadmap indicating how plug-and-play will interoperate with SpaceWire-RT, and the migration path that should be taken to ensure maximum functionality and flexibility, whilst retaining support for existing hardware.

1 INTRODUCTION

Since its official publication as an international standard in 2003 [1], SpaceWire has become the *de facto* standard for spacecraft onboard communication, at least within the payload. The rise in popularity of SpaceWire has resulted in a growth in the available instruments, onboard computers and other devices using the standard as the main method of data handling. By constructing equipment that uses SpaceWire for communication, a certain degree of interoperability is ensured (roughly corresponding to the data-link layer of the OSI model). Use of the Protocol ID standard [2] permits identification of standard protocols such as the Remote Memory Access Protocol (RMAP) [3] and a further level of interoperability. However, a SpaceWire network must still be constructed, and configured, carefully for a given application, usually requiring customised software and/or hardware. The lack of standardisation for simple tasks required on almost all SpaceWire networks limits the level of interoperability between devices and software, necessitating custom development for each application and resulting in higher development costs, longer development schedules and greater risk.

In the commercial, non-space, arena so-called 'Plug-and-Play' technologies have been developed to ease the integration of standard components and provide standardised mechanisms with which to carry out common tasks. The past three years have seen the development of a draft Plug-and-Play standard for SpaceWire. This paper discusses the current draft standard, presenting a service-oriented view of the document, highlighting its key features, and the use cases for their application. Given the existing popularity of SpaceWire, it would be difficult to immediately incorporate many of the features of SpaceWire Plug-and-Play (SpaceWire-PnP) into existing systems. The paper therefore presents a method for implementing a subset of services on existing hardware.

Perhaps the single largest imminent change to the way in which SpaceWire is used is the SpaceWire-RT (for Real Time, or Reliable and Timely) standard, which will provide quality of service for SpaceWire. SpaceWire-RT and SpaceWire-PnP are highly related, this paper discusses the connection between these two protocols, and the relationship they will have in systems of the future.

Finally, the paper discusses what the future might hold for SpaceWire systems, and for SpaceWire-PnP in particular, indicating the ways in which SpaceWire-PnP can be adapted and extended to cover new developments.

2 BACKGROUND

As described in a previous paper [4], from the perspective of a user of commercial equipment, application of the term 'plug-and-play' indicates that it should be possible to interface two or more arbitrary devices without the need for configuration. Plug-and-play generally involves two key aspects:

1. Automatic discovery and configuration of hardware and software systems in response to changes in hardware interfacing or availability including whilst the system is running.
2. Detection and configuration of the services that a plug-and-play enabled device provides.

SpaceWire does not offer a standard mechanism for detecting the topology of a network, or what devices are attached to it. Nor does it offer a standard mechanism for configuring the various aspects of a SpaceWire network, such as links and routers. SpaceWire also lacks and standard features to assist detection or configuration beyond the network, in the service domain. The absence of these features became increasingly important as the popularity of SpaceWire grew, and particularly in two application domains: the Operationally Responsive Space (ORS) programme [5] in the USA and the general use of laboratory test and development equipment and electrical ground support equipment (EGSE), especially within the European SpaceWire community.

The history and drivers behind the creation of a plug-and-play protocol for SpaceWire (SpaceWire-PnP), have been covered by the authors in previous papers [4,6] and will not be discussed here.

3 PRINCIPLES OF SPACEWIRE-PNP

3.1 BASIC PRINCIPLES

The central goal of the SpaceWire-PnP standard is interoperability at the network level. As such SpaceWire-PnP provides services to discover, identify and configure the features of a SpaceWire network, as covered by the SpaceWire standard, plus a few more corresponding to only the most common use cases. SpaceWire-PnP does not require devices to support more of the SpaceWire standard than is required to achieve their objectives: if something is optional in the SpaceWire standard, SpaceWire-PnP does not require that it be implemented.

SpaceWire-PnP takes the same view of a network as the SpaceWire standard, being entirely composed of links, nodes and routers. Nodes and routers are both referred to as *devices*. Nodes fall into one of two categories: active nodes, which may discover portions of the network and may attempt to manage other resources; passive nodes, which do not do either and expect to be managed by an active node. All devices have a configuration port (port zero) to which SpaceWire-PnP operations are addressed. Each network device is managed by one active node which is referred to as that device's *owner*. A device's owner may change, but at any given time a device has only one owner. Any node may read the settings of a device, but only the owner may modify them. If a node wishes to modify the settings of a device it does not own, it must do this through the device owner. This is discussed in more detail below.

3.2 NETWORK DISCOVERY AND DEVICE OWNERSHIP

The discovery of a SpaceWire network is driven by an active node. The network is discovered through an iterative exploration of links, routers and nodes, starting with those directly connected to the driving node and working outwards. Simulation work carried out by the authors [4] has determined that in most practical cases, a breadth-first search is fastest and uses the fewest resources. In this type of discovery a node should discover all of the resources attached to a router before attempting to discover what devices are, in turn, attached to those. As an active node discovers the network, it can claim ownership of each of the devices it finds.

A network may be discovered, and owned, by multiple active nodes. The possibility for concurrent discovery and attempts at device ownership requires that claiming ownership is an atomic operation. Claiming a device must identify the owner in such a way that they may be contacted, both to ensure their continued presence on the network and to permit the device to be configured by non-owners. In claiming a node, the claiming operation must atomically set the address (either logical or path) of the owner. If a logical address is used, routing resources may need to be configured to ensure that the owner can be reached. If a logical address is used in claiming a router, the router must automatically configure the routing tables when a device is claimed such that the logical address refers to the port on which the claiming operation took place.

A network with more that one active node presents the potential for competition in device ownership. To resolve a competition situation, each device is assigned a priority. Typically, this priority would be set *a priori* by the system designer such that devices are owned and managed in a deterministic manner. However, SpaceWire-PnP is also designed to deal with the case of a truly open system, in which nothing is known or assigned before devices are added to the network. In this case,

there is the potential for competition between devices with identical priorities. The conflict is resolved by using the physical port number with which the competing actives nodes are using to access the device.

3.3 Device Configuration and Proxy Identifiers

A device may only be configured by its owner. However, many network devices, especially routers, represent a shared resource, and many nodes may have an interest in their configuration. This problem is solved through the use of *owner-proxies*. When an owner claims a device, it atomically sets its address to identify itself. It must also, in the same atomic operation, assign the device a *proxy identifier* which may be used by any other node wishing to configure the device.

When carrying out any discovery or configuration operation the proxy ID is specified. A proxy ID of zero indicates no proxy, or that the discovery or configuration operation is addressed to the same device as the target of the transaction. A non-zero proxy ID allows a node to proxy the discovery or configuration for another device, one that it owns. This permits an owner node to vet requested configuration operations, disallowing ones that may cause issues with the correct operation of the network.

3.4 Principles of Implementation and RMAP

SpaceWire-PnP has a unique protocol ID which clearly identifies the packet as plug-and-play. With the exception of the protocol ID, SpaceWire-PnP uses a 100% compatible implementation of the RMAP protocol [3] for its basic semantics. The RMAP standard gives a great deal of flexibility to implementations, and SpaceWire-PnP standardises many areas that RMAP leaves open. For example, SpaceWire-PnP defines:

1. The RMAP operations that must be implemented: read, verified acknowledged write and RMW.
2. The RMW operation is defined as being a conditional write: the write only takes place if the read returns a value which matches the mask operation (see [8] for a discussion of this).
3. The addressing size and mode: 32-bit wide and incrementing only; all operations must take place on a multiple of 4-bytes. Non-incrementing addressing must be implemented if electronic datasheets, data sources or data sinks are implemented (see below).
4. The device must support a minimum of a 4-byte write and a 16-byte read.

All operations are addressed to the configuration port of a device and the return address should not include the port on the device that is being used, as this is added to the return path automatically, essential for network discovery.

4 SpaceWire-PnP Services

4.1 Service Summary

The facilities that SpaceWire-PnP provides can be grouped together into the following services:

1. **Device Identification and Status.** Provides core information to allow the identification of the device, its type (node/router, and device type field), and vendor information. Additionally, the device status, ownership details and port activity parameters are accessible. Error reporting is also part of this service: SpaceWire-PnP, Protocol ID and SpaceWire-RT error recording can be accessed, if implemented.
2. **Capability Discovery.** Provides a summary of the capabilities of this device. As SpaceWire-PnP is concerned only with interoperability up to the network level, *capabilities* are defined as protocols that this device supports, and the ways in which they can be transported (e.g. using SpaceWire-RT). A device may also use the capability service to provide one or more electronic datasheets, the format of which is identified by a type field but is not standardised by SpaceWire-PnP. This permits ORS xTEDS files [7] to be provided.
3. **Device Ownership.** Provides an atomic mechanism for claiming ownership of a device, identifying and contacting an owner and resolving conflicts.
4. **Owner-Proxy.** Device owners must use the proxy service to provide access to the devices they own. All claimed devices must identify their proxy ID.
5. **Link Status Configuration.** Provides the ability to query the status of any device link and configure its state and speed.
6. **Router Status and Configuration.** This service is applicable to routing devices only. Provides support for the configuration of routing tables and routing mechanisms, as well as time-code propagation.
7. **Time-Code Source.** This service, if implemented, provides a standard method for exposing a time-code source, allowing time-code generation to be enabled/disabled, the frequency of time-codes to be configured, and ticks to be generated manually, if appropriate.
8. **Generic Data Source.** Provides a standard way for a device, such as an instrument, to source data.
9. **Generic Data Sink.** Provides a standard way for a device (such as an actuator) to sink data.

Note that all devices should implement services 1-5, service 6 should be implemented by all routers. Services 7-9 are optional, to be implemented if appropriate.

4.2 Generic Data Sources and Sinks

A device, such as an instrument, may serve the function of sourcing data. SpaceWire-PnP provides a standard mechanism to enable this. A device may implement zero or more data sources which provide data in packets of a bounded size, with the maximum specified by the device. The device also identifies a type, indicating the content of the data. Three different mechanisms are provided by which data can be sourced:

1. Basic reads, where the device provides a data-ready status indicator in addition to a field from which the data can be read. Reads when the source is not ready return an error.

2. Delayed response reads, where the device does not respond to a read until data is ready. A timeout can be specified to ensure that the source responds, even if data is not ready.
3. Initiated RMAP writes, where the device transfers the data as an RMAP verified or un-verified, un-acknowledged write operation, targeting a specified SpaceWire address with a specified local RMAP address.

SpaceWire-PnP also provides a standard mechanism for the implementation of data sinks, such as actuators. Again, a device may implement zero or more data sinks which accept packets up to a specified maximum size. Two different mechanisms are provided by which data can be sunk:

1. Basic (unacknowledged) writes, where the device provides a sink-ready status indicator, in addition to a field to which the data can be written. Writes when the sink is not ready are discarded.
2. Queued acknowledged writes, where the device will accept one or more acknowledged writes. If more than one write is accepted concurrently, the writes are placed in a FIFO queue. If the queue is full, the sink returns an error.

Data sources and sinks most obviously apply to nodes; however, router-driven network discovery, in which a router notifies interested active nodes of changes to link status, can be implemented as a generic data source attached to a routing device.

5 LEGACY SUPPORT IN SPACEWIRE-PNP

SpaceWire-PnP is a new protocol which is transported by a compliant implementation of an existing standard (RMAP). As such, hardware and software for handling SpaceWire-PnP (RMAP) packets already exists and is well proven. However, as the encoding of the various parameters (the RMAP address space) is new, this is not implemented by current hardware, nor is support for the SpaceWire-PnP protocol ID.

In contributing to SpaceWire-PnP, the UoD was able to draw from its extensive experience in designing SpaceWire routers, culminating in the design for the SpW-10X (Atmel device AT7910E). The SpW-10X implements many features that correspond to the core SpaceWire-PnP services. These are implemented using RMAP with a device-specific address space. The SpW-10X supports SpaceWire-PnP services as summarised below:

1. **Device Identification and Status.** Vendor and device ID are provided.
2. **Capability Discovery.** Capability information can be inferred.
3. **Device Ownership.** Limited atomicity is provided by the SpW-10X implementation of RMAP RMW. By combining this with suitable timeouts, the device ownership mechanisms required for SpaceWire-PnP can be implemented.
4. **Owner-Proxy.** The SpW-10X can support the specification of a proxy ID.
5. **Link Status Configuration.** Full link status and configuration is provided.
6. **Router Status and Configuration.** All standard features are provided.

The SpW-10X does not implement generic data source or sink services as these are not relevant. Although the SpW-10X can source time-codes, there is no method for accessing the time-code source via RMAP.

With a different low-level implementation, a SpW-10X-aware version of SpaceWire-PnP can implement all of the core, required services in a standard way, allowing full interoperability with later SpaceWire-PnP devices. Providing the mechanisms are publicly available, any legacy device which supports the core features of SpaceWire-PnP, especially ownership, can be incorporated into a network in this manner.

The Remote Terminal Controller (RTC) (Atmel device AT7913E) is intended as a highly capable processing unit with SpaceWire interfaces (in addition to CAN and MIL-STD-1553 interfaces). The RTC also has a fully-featured RMAP target core; however, this cannot be used for SpaceWire-PnP as it maps the RMAP address space directly onto the host address space without any provision for address translation. The most appropriate way to implement SpaceWire-PnP on the RTC is therefore is software, where a full implementation can be achieved.

6 IMPLEMENTATION OF SPACEWIRE-PNP

By using a compliant implementation of RMAP as the transport for SpaceWire-PnP, the protocol aims the leverage existing hardware and software as much as possible. A device must support two potential differences from a standard RMAP implementation in order to comply with SpaceWire-PnP:

1. SpaceWire-PnP packets are addressed to the configuration port (port zero) of every device, therefore the device must recognise a zero byte at the beginning of the packet. For routers, this is likely to be handled by the routing fabric.
2. Although SpaceWire-PnP uses an implementation of RMAP, it does not use the RMAP protocol ID as this would fail to specify the standard nature of the address space being used. The device must therefore recognise the SpaceWire-PnP protocol ID.

A standard RMAP core (either hardware or software) must therefore be adapted to recognise both the leading zero (if applicable) and the protocol ID. A simple adapter layer can be added before the core to handle these features and provide a flag to users of the RMAP core to indicate that a transaction is for SpaceWire-PnP rather than RMAP. In this manner, an RMAP implementation can support both SpaceWire-PnP and standard RMAP transactions with the same implementation, this means that devices that currently implement RMAP require very little additional logic (again, either hardware or software) in order to support SpaceWire-PnP. For devices that don't currently implement RMAP, the set of RMAP commands that must be supported for a basic implementation of SpaceWire-PnP is sufficiently small to limit the overhead required for SpaceWire-PnP support.

7 SPACEWIRE-PNP AND SPACEWIRE-RT

SpaceWire, as an asynchronous network, does not offer any quality of service (QoS), either in respect to reliability or in respect to timeliness. The forthcoming SpaceWire-RT protocol [8] adds a, largely transparent, layer on top of SpaceWire to provide a full range of qualities of service. These features, along with those provided by SpaceWire-PnP, will form an essential part of the CCSDS SOIS (spacecraft onboard

interface services) SpaceWire sub-network mapping [9]. The relationship between SpaceWire-RT and SpaceWire-PnP is threefold:

1. SpaceWire-PnP may be used on a SpaceWire-RT network which means that active nodes must obey certain rules in order to preserve the QoS provided by SpaceWire-RT.
2. SpaceWire-RT uses SpaceWire-PnP mechanisms to initiate, configure, manage and close *channels*, the virtual end-to-end links over which SpaceWire-RT transports data.
3. SpaceWire-RT can be used to provide reliable and/or timely channels over which SpaceWire-PnP transactions can take place. Whilst RMAP offers very basic reliability in the form of CRCs and replies, these alone are unlikely to be sufficient. SpaceWire-RT offers the *assured* and *guaranteed* services which both provide reliable transport (on an asynchronous or scheduled network respectively)

SpaceWire-RT support can be viewed as a further (tenth) service provided by SpaceWire-PnP.

A channel may be opened by specifying all channel parameters in an acknowledged, verified SpaceWire-PnP write. The channel is successfully opened if a non-error return code is given in the reply. If a channel number is not known *a priori*, status information for all channels can be queried and used to find potentially free channel numbers. A channel may be opened by a node which is neither the source nor the destination. A channel can be closed using a simple SpaceWire-PnP operation which clears all channel information and releases the associated resources.

SpaceWire-PnP provides a simple table which lists the status of all open SpaceWire-RT channels. This permits an active node to check all open channels for errors, and to verify the presence and correct operation of channels. Additionally, a status query can include requests from the device being queried for SpaceWire-RT channels to be opened on its behalf. This permits both passive nodes (and routers), and active nodes not assigned management bandwidth on a scheduled network, to request channels with appropriate QoS. Once the request has been satisfied, a network node manager can write back to the requesting node with the new channel number.

On joining a SpaceWire network, an active node must account for the fact that the network may be using SpaceWire-RT, and it may be scheduled. To ensure scheduling rules are not broken, a new node must wait before commencing network discovery, or any other SpaceWire-PnP operation. If, during that time, a time-code is received, the node must wait until timeslot zero, which is reserved for management operations. At that point the node may proceed to discover the network. On an asynchronous (un-scheduled) SpaceWire-RT network, SpaceWire-PnP may proceed as normal.

8 THE FUTURE

Through its various drafts, SpaceWire-PnP has evolved to cover the most important features of a SpaceWire network, including some of the key use cases, such as data sources and data sinks. A guiding principle has been efficiency, which would be improved by including a standard mechanism for the exchange of topology information between those nodes carrying out network discovery. SpaceWire-PnP is

well positioned to take account of changes including any additions. One possible such addition is the real-time signalling mechanism proposed by Sheynin *et al* [10].

9 CONCLUSIONS

The SpaceWire-PnP draft standard has come a long way since it was created as part of the work of the SpaceWire Working Group plug-and-play sub-group. The proposals in this paper seek to provide a highly flexible and extensible standard which leverages existing technology and provides appropriate opportunities for the support of legacy systems such as the sophisticated SpW-10X router and the RTC device. This is accomplished using an implementation of RMAP as a transport and providing standard mechanisms to permit the discovery and management of network resources by multiple nodes simultaneously in an interoperable manner.

The standardisation and wide-spread implementation of SpaceWire-PnP will provide the basis for device and vendor interoperability, lowering spacecraft development time, costs and risk, and expanding the market for SpaceWire devices.

10 REFERENCES

1. ECSS, "SpaceWire - Links, nodes, routers and networks", ECSS-E-50-12A, 24th January 2003.
2. ECSS, "Protocol Identification", ECSS-E-50-11 Draft F, 4th December 2006.
3. ECSS, "Remote Memory Access Protocol", ECSS-E-50-11 Draft F, 4th December 2006.
4. Mendham *et al*, "Plug-and-Play Technology for SpaceWire: Drivers and Alternatives", IAC-07-B4.7.06, Proceedings of the 58th International Astronautical Congress, Hyderabad, India, 2007.
5. US DOD, "Plan for Operationally Responsive Space", A Report to Congressional Defense Committees, National Security Space Office, USA, 17th April 2007.
6. Mendham *et al*, "Interoperability And Rapid Spacecraft Development using SpaceWire Plug-And-Play", IAC-08-B4.7.06, Proceedings of the 59th International Astronautical Congress, Glasgow, UK, 2008.
7. Lyke *et al*, "Space Plug-and-Play Avionics", Proceedings of the 3rd Responsive Space Conference, Los Angeles, CA, USA, 2005.
8. Parkes and Ferrer-Florit, "SpaceWire-RT Initial Protocol Definition", Draft 2.1, SpaceNet Report No. SpW-RT WP3-200.1, ESA Contract No. 220774-07-NL/LvH, October 2008.
9. Fowell and Taylor, "The SOIS Plug-and-Play Architecture and its Proposed Mapping onto SpaceWire", Proceedings of Data Systems in Aerospace, DASIA 2008, Palma de Mallorca, Spain, 2008.
10. Sheynin *et al*, "Real-Time Signalling in SpaceWire Network", Proceedings of the 1st International SpaceWire Conference, Dundee, UK, 2007.

Networks & Protocols 2

Wednesday 5 November
13:40 – 15:20

PROTOTYPE IMPLEMENTATION OF A ROUTING POLICY USING FLEXRAY FRAMES CONCEPT OVER A SPACEWIRE NETWORK

Sev Gunes-Lasnet[(1)], Olivier Notebaert[(1)]

[(1)] *Astrium, 31 avenue des Cosmonautes, 31402 Toulouse Cedex - France*
Phone: +33 (0)5 6219 7663 - Fax: +33 (0)5 6219 7158
Email: sev.gunes-lasnet@astrium.eads.net

ABSTRACT

The benefit of SpaceWire as an efficient data-link to transmit science data on board spacecraft is now demonstrated through the growing development of SpaceWire in major space science projects such as Bepi-Colombo, Gaia and the James Webb Space Telescope. The efficiency of SpaceWire becomes even more obvious when used in a network configuration. Although not yet really popular onboard spacecraft, a network configuration not only reduces the overall data links mass, it also enables flexible implementation of scalable distributed systems which can be of great interest for future applications. However, state of the art SpaceWire networks cannot offer sufficient levels of communication services quality with the existing protocols which limits their field of application to non critical applications from the dependability or real-time standpoint.

The FlexRay communication concept developed in the automotive industry takes into account dependability and real-time properties to provide guaranteed access to the network for critical data transfers through parallel channels and time slotting. The remaining bandwidth is shared on a best effort basis for non-time critical data transfers. A study performed at Astrium makes an adaptation of FlexRay protocol elements such as communication cycle and routing strategy over a SpaceWire network.

1. CONTEXT

Several drivers for onboard data links and I/Os upgrade can be identified, such as new missions requiring autonomous manoeuvres for space exploration or the development of robotics in space. The emergence of new spacecraft control mode, e.g. using vision and star trackers instead of accelerometers and gyroscopes, induced new needs in terms of data processing and in particular for the on-board data links capabilities [1] [2].

As a consequence, enhanced capabilities are required for on-board data links in particular due to the increase of data exchange volume (towards the Gigabits/s range), the real-time command and control requirements and the need for in-orbit software maintenance, together with a need for lower power consumption to perform solar system exploration and to allow to reach further spacecraft autonomy.
Moreover, in the context of optimisation of on-board data links to satisfy to new needs in terms of data processing, there is a growing interest for decentralised/distributed architectures implying peer-to-peer communications rather than master/slave ones.

In order to face the identified challenges for on-board data processing and in particular for on-board data links in an effort of cost reduction, the re-use rate, maturity and interoperability of the technology developments should be optimised; this is why standardisation is pushed forward by the different space agencies and industries [3].
Other constraints such as harness optimisation are pushing for network or buses configurations to simplify the integration and operations of spacecrafts.

2. ON-BOARD DATA LINKS OPTIMISATION

2.1. Current use of high speed data links

1355 links, which are the precursors of SpaceWire, are used in several spacecrafts such as Science data to Mass Memory (Cryosat, Rosetta, Mex/Vex) and for telecom signal dynamic switching (Inmarsat4).
Other high data rate links are used for Gbit performance requirements in missions such as Pleiades, TerraSAR-X. High speed data links are also used as on-board computer ground test interface for software instrumentation.
SpaceWire has been considered in most studies on future data processing architecture and Leon System-on-Chip prototypes:

- In ESA studies such as SCoC and the A3M demonstrator, A3SysDef, Gamma, Disco…[4] [5][6][10]
- And also in national agencies and EADS-Astrium internal projects such as ALF3, Unionics, MAEVA, PADAPAR… [7] [8]

SpaceWire/ECSS-E50-12 supports the need for high performance data processing on future spacecrafts and is recommended for any point-to-point high speed link with a data rate fewer than 200 Mbps; it is viewed as a future solution towards more generic payload data processing systems through on-board data networks [9]. SpaceWire networks could also be extended to the whole data processing system (e.g. for small vehicles, robotics…).

2.2. On-Board data links optimisation strategy

The current typical on-board data links architecture is challenged by new needs in particular in terms of Payload data processing, as described in the first paragraph of this paper.

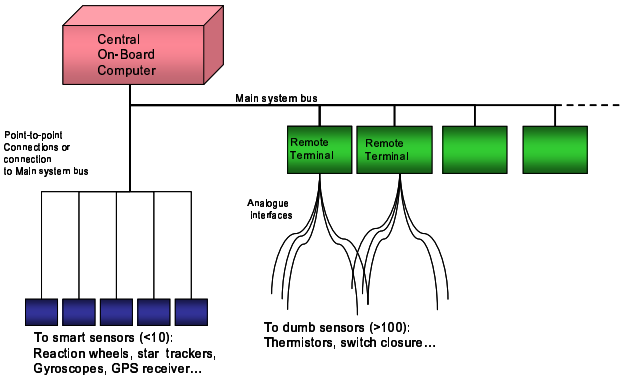


Figure 1: Typical on-board data links architecture

The optimization of on-board data links targets the use of on-board networks, in particular for payload data processing. On-board networks could answer the anticipated needs in terms of data throughput, determinism and reduced harness mass. SpaceWire is in this context considered to be of high interest for the development of payload data handling building blocks, in particular because it could act as a multi-node backbone, see Figure 2.

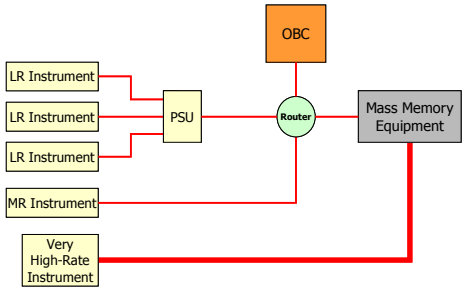


Figure 2: SpaceWire as a multi-node backbone

Such SpaceWire networks are considered to be able to optimise on-board resources and performances; to insure their development, the consolidation of identified points presented in the following paragraph is necessary.

2.3. Critical elements for SpaceWire networks development

As identified by the SpaceWire working group and discussed during the first SpaceWire International conference in 2007, critical aspects for the development of SpaceWire networks on-board spacecraft are in particular:

- The availability of SpaceWire Network building blocks elements for embedded systems such as space qualified components and IP's for integration in System on Chips;
- Ground support software/hardware, e.g: adapters to ground networks (usb, ethernet…), Traffic monitor/simulator, architecture model, network administration SW, simulation…
- Standardised protocols ensuring time deterministic packet delivery, (e.g. derived from existing aeronautics standards).

The activity described in this paper focuses on the deterministic protocols issue.

3. PROTOTYPING AND TESTING OF A ROUTING POLICY USING FLEXRAY FRAMES CONCEPT OVER A SPACEWIRE NETWORK

High level protocols implemented over SpaceWire will allow ensuring the respect of timing constraints, while respecting the SpaceWire standard.

To perform a first iteration analysis of how a high level protocol could be implemented over SpaceWire, the prototyping of concepts from the automotive protocol FlexRay has been performed under the form of a pilot implementation over a SpaceWire network. This implementation will allow further analysis of the concept and in particular performance assessment.

3.1. FlexRay consortium overview

The Flexray consortium originally chartered in end of December 2006, with a new charter established in January 2007. It targets the automotive market, with automotive control applications.
The core members are BMW, Bosch, DaimlerChrysler, FreeScale, GM, Philips and Volkswagen. The consortium is composed of 7 Core, 13 Premium, 47 Associate, and 27 Development members so far.
Initially, the consortium was only open to automotive industry players (semiconductor manufacturers, equipment and system suppliers, tools and services suppliers and manufacturers). However in late 2006, the consortium agreed the membership to EADS Innovation Works Germany as a move to non-automotive companies. Technology use and licensing is still not cleared for non-automotive applications.

3.2. FlexRay technology overview

FlexRay's main characteristics are directly resulting from the automotive X-by-Wire requirements: fault-tolerant clock synchronization via a global time base, collision-free bus access, guaranteed message latency, message oriented addressing via identifiers, scalable system fault-tolerance, and speed of one order of magnitude higher than CAN [11].
There are two interesting properties of FlexRay for supporting the deterministic communication and reliability required by space systems: the first one is the division of the bandwidth between a deterministic time triggered communication part and a user defined run-time scheduled communication part; the second property is the possibility of using redundant channels if this is desirable.
FlexRay uses the OSI-model as reference model. The layers that FlexRay uses are the Application, Data Link and Physical layers (Figure 3). The Network and Transport layer are typically placed in the second layer, and the Session and Presentation layers in the seventh layer.

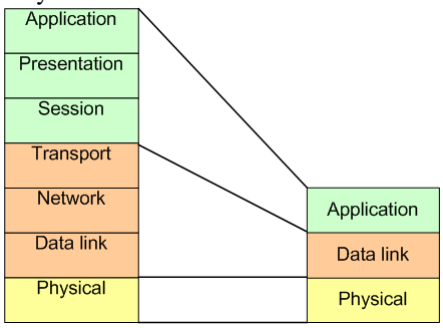


Figure 3: Layer mapping between OSI and FlexRay

3.3. Reuse of automotive protocols concepts

Given the position of the FlexRay consortium towards non-automotive applications, the prototyping implementation of a routing policy using its concepts has been performed, rather than the direct evaluation of fully supported FlexRay features products.

3.4. Prototyping platform

The prototyping platform chosen is depicted in Figure 4: It is composed of a SpaceWire router from 4Links, a DSI (Diagnostic SpaceWire Interface) also from 4Links which has been programmed for analyzing the SpaceWire traffic, and 3 boards with SpaceWire ports and LEON2 processors in FPGAs. Two boards are AVNET boards, while the last one, acting as time master, is a Pender board. Two SpaceWire ports have been used on the AVNET2 board (see Figure 4).
Moreover, a SpaceWire IP tunnel [10] from Star Dundee has been used between the router and the different boards to monitor the traffic and check for eventual communication problems. The DSI has been used to analyze the traffic which arrives at its ports. On the contrary of the router, it is possible to control the DSI by programming the API provided by the constructor.

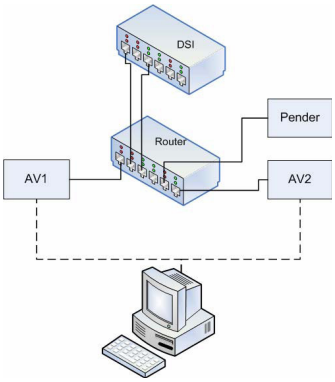


Figure 4: prototyping platform

3.5. Communication model prototyping

The Rosetta spacecraft on-board communication model has been chosen in order to allow the simulation of real payload data to be transmitted over the SpaceWire network. Within Rosetta data handling architecture, 8 modules use IEEE-1355 serial links. The traffic chosen for this pilot implementation has been derived from the traffic forecasted on the 1355 links between the Solid State Mass memory and the avionics processor, between one instrument and the processor, and between one instrument and the memory.

During the prototyping, a time master on the network distributes a global time reference; this is the Pender board task. The different modules use this time reference for sending packets over the network.

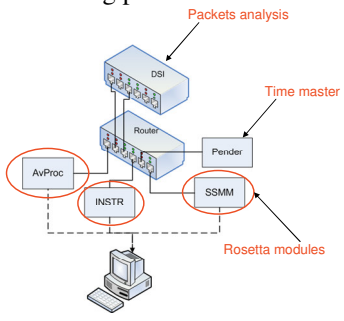


Figure 5: Prototyping configuration

The Pender board has one six-bit time counter (0 to 63), which is incremented each second. The generated time code signal is broadcasted to all the output ports of the router. All nodes connected to the router (all AVNET boards) have one six-bit counter. When the interface receives the broadcasted time-code, it updates its associated time-counter with the new time. This common time reference is used for the transmission of the packets. Each board has been programmed to transmit packets which are based on

the Rosetta payload packets: One board acts as Avionics processor board, while the other corresponds to the Instrument. Their flow diagram is depicted in the following figures.

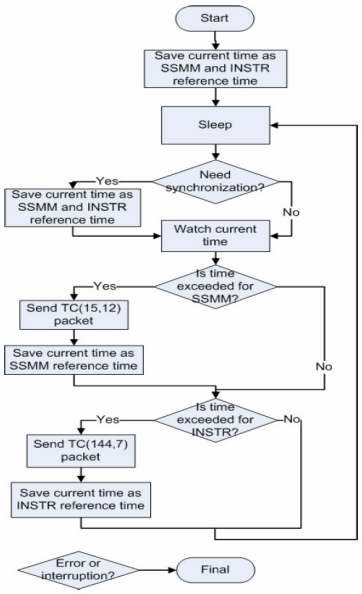


Figure 6: Flow diagram of the avionics processor board send packet task

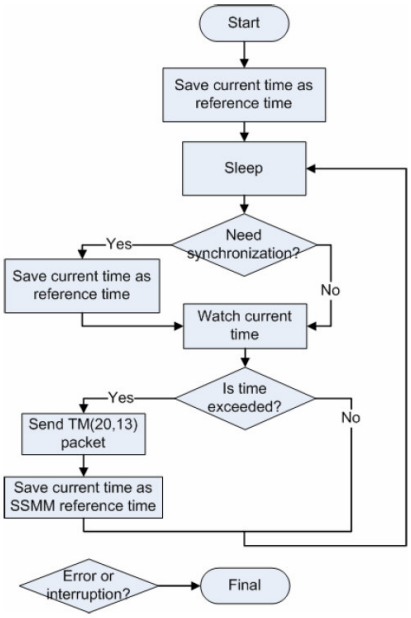


Figure 7: Flow diagram of the instrument board send packet task

The configuration implements a priority based bandwidth reservation: The packets with high priority are guaranteed to be transmitted first during the static part of the cycle. In order to re-use concepts from the automotive, two time zones have been defined for each communication cycle, as depicted in Figure 8;

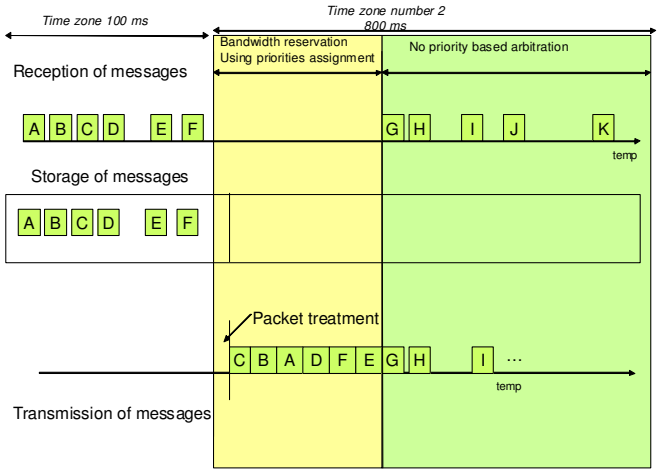


Figure 8: Communication cycle

A communication cycle of 900ms has been defined by dividing it into a priority time frame of 100ms and a secondary time frame of 800ms:

- The first time zone takes as reference the first packet received, and lasts until 100ms after the reception of the first packet. The packets which arrive in this time zone will be considered as simultaneously arrived, and they will be managed in base of their priorities. This is considered to be the priority zone, and is designed to implement similar capabilities as the time-triggered part of the communication cycle. Due to the lack of synchronization between the different modules in the pilot implementation, this time zone can however not be considered as fully time-triggered.
- The second time zone takes as reference the first packet received after the end of the priority zone and lasts 800ms. In this time zone, there is no priority based packet processing; the packets are managed with the original SpaceWire driver. This time zone has been implemented with the aim of allowing event driven communications during each communication cycle.

Such a communication cycle model has been chosen in order to correspond to the identified needs of bandwidth reservation and event triggered packets transmission over a SpaceWire network for payload data processing.
The aim of the pilot implementation was to implement a first iteration of a communication cycle design over a SpaceWire network rather than to implement a full time-triggered approach; the routing is done through an analyse-and-transmit routing approach. The prototyping platform will allow further analysis and performance assessment of high level protocols over SpaceWire networks.

3.6. Packet analysis

The DSI acts as packets analyser unit, as depicted in the flow diagram in Figure 11, and has two of its ports connected to the router. The priority packets are stored during the 100ms time slot and once the allocated time is over, a packet treatment thread sorts the messages by descending priority (inferior number, higher priority). The packets are then sent one after the other, after which the secondary time frame of 800ms is started. Finally, the message list is deleted. The ports and priorities used for each link are specified in Figure 9.

	Port	Port DSI	Port exit DSI	Priority
AV1	7	3	2	7
AV2.0	5	2	2	5
AV2.1	8	2	2	8

Figure 9: ports and priorities used

For this prototyping, the priority has been introduced in the second octet of the SpaceWire packets as depicted in Figure 10.

Destination id	Cargo	End of packet	
SpaceWire packet format defined in ECSS SpaceWire standard			
Destination id	Priority	Cargo	End of packet
SpaceWire packet format proposed by this communication model			

Figure 10: SpaceWire packet format used in the pilot implementation

The flow diagram in Figure 11 shows how the analysing task manages the packets.

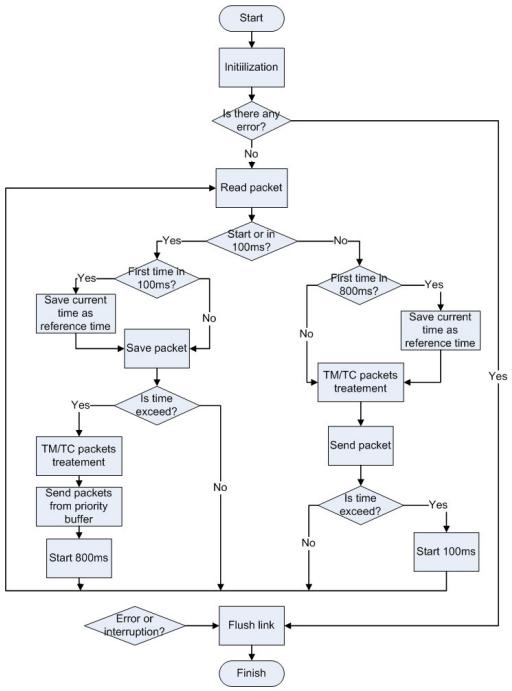


Figure 11: Flow Diagram of the DSI function as packet analyser, using Rosetta packets

4. PROTOTYPE TESTING RESULTS ANALYSIS

The reorganisation of the packets during the first part of the communication cycle has been monitored, as well as the routing of packets without priority based bandwidth attribution during the second part of the cycle. Figure 12 presents an overview of the results obtained for the priority based treatment of the Rosetta-like packets:

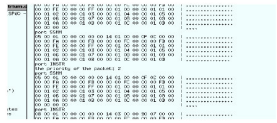
 Figure 43: AvProc sending TC messages	The AvProc module sends TC packets.
 Figure 44: The messages sent by the DSI after the treatment	The DSI receives the packets, it treats them, and then the packets are sent to the logical port.
 Figure 45: the AvProc receiving TM messages	The AvProc receives a TM packet. That means that it has received an answer to a request from another module.

Figure 12: Rosetta payload packets simulation overview

The transmission of packets and its correct processing by the programmed DSI has been observed during a test procedure which parameters are presented in Figure 13.

Parameters of the test	*values*
Number of messages transmitted	1 message each 100 ticks
Duration of the test	More than 100 messages sent
Number of messages received in time frame 1	Average of 5 messages
Number of messages received in time frame 2	Average of 40 messages

Figure 13: Test procedure parameters

The communication cycle concept presented in paragraph 3.5 has been successfully implemented; during the first part of the communication cycle, each board transmits different packets which are re-organised by the DSI and routed by the SpaceWire router. During the second part of the cycle, no priority based routing is used, and the DSI does not re-organise any packets arriving at its input ports.

Several improvements can be made to the prototyping developed for this pilot implementation. It will be needed in another iteration of the prototyping to increase the traffic between the modules. A higher number of modules can be used to match the Rosetta configuration. For this implementation, default speeds of 100 mbps have been used for the SpaceWire links, this can be adapted in further iteration.

In order to perform a further test of a bandwidth reservation policy based on priorities, a synchronisation method between the different modules clocks is needed. This will insure precise attribution of communication slots to each node connected to the SpaceWire network.
The optimisation of the communication cycle definition needs to be performed; the length of the packets and of the communication cycle can be adapted.
In addition, for an implementation based on the FlexRay concept, bandwidth reservation relies on the intelligent routing strategy, which is located in the router. The DSI has been used as an intelligent router in order to ease the implementation given that the DSI could easily be programmed. Such intelligent routing strategy requires the specification of dedicated router capabilities such as slots reservation for each user on the network, control of the packets transmission to prevent babbling idiot errors, and possible further capabilities.

5. CONCLUSIONS AND PERSPECTIVES

The prototyping presented is a first implementation of concepts developed in the automotive industry, which are of interest for the development of SpaceWire networks building blocks for future on-board payload data processing. The implemented model is to be further iterated in particular in terms of synchronization, communication model optimization and representative traffic.
Further developments are on-going; the prototype will be used for the measurement of the sensitivity of the latency with respect to the length of packets transmitted over the network, and for the optimal dimensioning of the network elements: Size of an internal router buffer, optimal length of the communication cycle…
Moreover, the platform used for this pilot implementation can be used for the evaluation of further concepts of real time protocols over SpaceWire networks, thus contributing in the development of future on-board applications using SpaceWire networks.

6. REFERENCES

[1] B. Frapard, S. Mancuso, Astrium Satellites "Vision Navigation for European Landers and the NPAL Project", *Proceedings of the 6th ESA International Conference on Spacecraft, Guidance, Navigation and Control Systems, 2005*

[2] "Autonomous Vision-based navigation demonstration », Bernard Polle, Anthony Villien, Jacques Lheritier, Benoit Frapard, Astrium Satellites, *Proceedings of GNC 20087th International ESA Conference on Guidance, Navigation & Control Systems*

[3] "Benefits of the standardisation efforts for on-board data interfaces and services », Olivier Notebaert, Astrium Satellites, *Proceedings of SpaceOps 2006 conference*

[4] Marc Le Roy - Astrium Satellite, "Advances Avionics Architecture and Modules (A3M) Final Report", ESA Contract 13024/98/NL/FM (SC), September 2003.

[5] Philippe Chevalley-ESA/ESTEC, Luc Planche, Astrium Satellites, Stuart Fowel-Scisys, DisCo – A Space-Oriented Middleware for future Payload Data Handling Equipments Proc. 'DASIA 2006 – DAta Systems In Aerospace Conference', Berlin, Germany, 22-25 May 2006 (ESA SP-630, July 2006).

[6] Christophe Honvault - Astrium Satellite, "Generic Architecture for Mass Memory Access (GAMMA) executive summary" ESA Contract 17437/03/NL/AG.

[7] « MAEVA for very integrated, low power plat-form systems », Marc Lefebvre, Jean-François Coldefy , Astrium Satellites, *Proceedings of DASIA 2005*

[8] Christophe Honvault, Olivier Notebaert – Astrium Satellites, "Prototyping a Modular Architecture for On-Board Payload Data Processing - PADAPAR", *Proc. 'DASIA 2007 – Data Systems In Aerospace Conference', Naples, Italy, 29 May - 1 June 2007 (ESA SP-638, August 2007).*

[9] Christophe Honvault, Olivier Notebaert - – Astrium Satellites, SPACEWIRE NETWORKS FOR PAYLOAD APPLICATION*, International SpaceWire Conference 2007, Dundee, United Kingdom, 17-19 September 2007*

[10] Christophe Honvault (Astrium Satellites), Raffaele Vitulli (ESA/ESTEC) Topnet Pilot Operation Implementation: A First Step To Virtual Satellite Integration, *Proc. 'DASIA 2008 – data Systems In Aerospace Conference', Palma de Mallorca, Spain, 27-30 May 2008 (ESA SP-665, August 2008)*

[11] "Flexray - An ANSWER TO THE CHALLENGES FACED BY SPACECRAFT ON-BOARD COMMUNICATION PROTOCOLS" Sev Gunes-Lasnet, Gianluca Furano, *Proceedings of DASIA 2007 conference*

SPACEWIRE-RT PROTOTYPING

Session: SpaceWire Networks & Protocols

Long Paper

Albert Ferrer and Steve Parkes

School of Computing, University of Dundee, Dundee, DD1 4HN, Scotland, UK.

E-mail: aferrer@computing.dundee.ac.uk, sparkes@computing.dundee.ac.uk

ABSTRACT

SpaceWire Reliable-Timely (SpW-RT) aims to provide a consistent quality of service (QoS) mechanism for SpaceWire networks. It is intended to provide reliable, high data rate communication services and support for control applications where timely delivery is essential. This paper presents the ongoing prototyping activities related with SpW-RT. It introduces the most important protocol functions and how they may be implemented in a wormhole switching network. The lessons learned from the prototypes built were an important contribution to the design of the current SpW-RT protocol specification.

1 INTRODUCTION

SpaceWire Reliable-Timely (SpW-RT) is being designed to provide quality of service mechanisms over SpaceWire [1,2]. Some important design guidelines and objectives are summarized below:

- Achieve high performance and reliability, while being simple to implement and understand.
- Reduce the protocol complexity required by enforcing and standardizing good network design practices.
- Support for software (slow processing power but big buffers) and hardware implementations (fast processing power but small buffers)
- Support for processor-based intelligent nodes, and dumb nodes interfacing instruments and other devices.
- Compatible with most significant SpW compliant devices.
- Provide enough flexibility to accommodate multiple user cases under a unique protocol definition.

SpW RT targets asynchronous networks and scheduled networks. This includes multiple user cases such as:

- Asynchronous communication with dedicated links. It may require high throughput, reliability and flow control.

- Asynchronous communication using shared links that tolerates variable message latency. It may require reliability and flow control.

- Synchronous system for periodic status messages transferred using a guaranteed service.

- Synchronous system for sporadic control messages transferred using a guaranteed service, with opportunistic use of otherwise unused timeslots.

For easy understanding, in the following chapters the term “message” refers to a complete user data unit. The term “packet” refers to an actual SpW packet. The term “data packet” refers to a packet containing a segment of a user data unit.

2 SPW-RT RELATED TOPICS

This chapter provides background information and presents some concepts involved in the development of the SpW-RT protocol.

2.1 ROUTING

SpaceWire networks use wormhole switching so the packets are not completed buffered in the routers before they are routed. Instead, packets are immediately routed depending on the content of a single byte header. This provides very low latency when there is no congestion in the network and it is usually very simple to implement. The disadvantage is that a packet will block all network resources (SpW links) used by the packet until the transfer is completed.

Path and logical addressing

SpW routers implement logical and path addressing; optionally using Group Adaptive Routing (GAR). Logical addressing allows the packet routing to be changed by modifying the routing tables of routers, without requiring the SpW nodes being notified. With path addressing, nodes must explicitly provide the path to the destination. Both techniques allow using one or more bytes of the header for the routing. Logical addressing is usually implemented with a single byte that matches the logical address of the destination node. However, it is possible to have multiple routes to the same destination using the router's header deletion feature, in the same way that regional addressing is implemented.

Routing priority

Furthermore, some routers may implement a routing priority scheme to deal with multiple packets waiting to be routed through the same output link. For example, SpW-10X allows two priorities levels within a round robin scheme. However it is not possible with standard SpW routers to pre-empt low priority packets.

2.2 RELIABILITY AND REDUNDANCY

SpW standard provides error detection mechanisms for point to point links but as other link layer standards, it does not provide any end-to-end reliability. This capability must be provided by a suitable transport layer such as SpW-RT.

Reliability can be obtained by acknowledging received packets, optionally retrying in case of an error. Multiple redundant paths can be used simultaneously or sequentially. User data is acknowledged using sequence numbers for the data packets or for the data bytes correctly received. This also ensures that data is not out-of-order.

Sending window

The quantity of data or data packets that can be sent without being acknowledged is usually called the sender window. Note that when flow control is considered, the sender window must take into account the receiver window.

SpW networks typically do not buffer packets in the routers so the delay in the reception of the acknowledgement is introduced by the receiver and the possible network congestion while sending the acknowledgement. The sending window should be adjusted so that the maximum throughput is achieved while minimizing the sender buffer required.

Fault cases

SpW links have theoretically, a very low bit error rate. Packets are more likely to contain errors or be lost because of network congestion, permanent faults or other transient faults. Network congestion may force the routers to timeout the blocked packets and spill them. Network congestion may occur because of an inadequate network analysis or by an error in any part of the network. Use of a retry mechanism over the same path is inadequate in case of permanent faults on this path, but may be appropriate when the fault is not located on the path to the destination.

Redundancy

The GAR mechanism can cope with permanent link errors and blocked links but is not suitable to deal with faulty routers, as they may contain invalid routing tables. Therefore, multiple redundant paths are necessary to provide fault tolerant capabilities. High critical applications running in a scheduled network may have reserved redundant paths. In this case it is recommended to implement the simultaneous retry technique, so packets are sent simultaneously on the primary and redundant paths and the receiver discards duplicated packets. On the other hand, non critical applications may use a redundant path only when the primary one fails.

2.3 TIMELINESS

Packet delivery time or packet latency is variable in asynchronous SpW networks without dedicated links for each data transfer. Packets going to nearby destinations will tend to have lower latency than packets going to more distant destinations. Higher latency implies higher use of bandwidth, as links are being used for longer time. Synchronous networks allow deterministic latency for packets and messages but

have a penalty in terms of flexibility and performance. For some applications, the knowledge of the maximum message latency may be sufficient.

Worst case packet latency

Packet latency in asynchronous networks can be very high in some topologies and network traffic. Computation for three simple cases is presented below assuming equal routing priority and round robin mechanism.

a) All links in the destination path are not shared with any other sending entity. Then, the latency will be constant in absence of errors and will depend on the maximum packet size (M), number of hops to the destination (H), link speed (S), and switching delay (Ts):

$$lc = M / S + H \bullet \mathrm{Ts}$$

b) All sending entities send simultaneously a single packet to a unique destination in a network with a simple line topology (Fig 1). The maximum latency corresponds to the furthest possible source destination pair and its value is:

$$l\big|_{\max} = \sum_{r=1}^{H} (L_r - 2) \bullet \left(M / S + H_r \bullet \mathrm{Ts} \right) + \left(M / S + H_r \bullet \mathrm{Ts} \right)$$

Where *Lr* denotes the number of links in the router *r* and *Hr* is the number of hops to the destination for the nodes attached to router *r*.

c) Same case as b) except that packets are generated continuously. The maximum latency corresponds to the furthest possible source destination pair and its value is:

$$l\big|_{\max} = \prod_{r=1}^{H} (L_r - 1) \bullet \left(M / S + H_r \bullet \mathrm{Ts} \right)$$

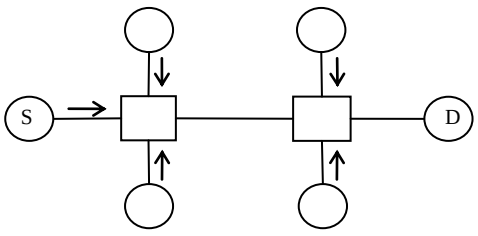


Figure 1: Linear topology with four port routers. The maximum packet latency from S to D is five times the minimum latency if a single packet is sent by each node, and nine times if packets are sent continuously. Note that the worst case is assumed; all nodes send packets to destination D.

Note that it is assumed that packets are consumed at the destination.

The worst case packet latency is not trivial to obtain for arbitrary topologies. Although it may not be possible to express it analytically, it may be computed if the expected network traffic is known. The message latency is usually easily derived from the packet latency.

Time Division Multiplexing (TDM)

Packet latency can be made deterministic and constant by dividing the transmission time in multiple time-slots grouped into epochs. For each timeslot multiple sets of unidirectional links are defined. Each set of links is reserved to be used in this timeslot by a unique source node in such a way that there is no link that is included in the destination path of two different source nodes. Multiple packets can be sent during a timeslot provided that they are delivered before the beginning of the next timeslot. This technique ensures that there is no packet blockage caused by other packets using the same network resources (links).

Scheduled networks using TDM are very efficient when the expected network traffic is periodic and known. Depending on the application it can provide higher throughput and lower message latency than asynchronous networks. However when traffic is sporadic and unknown it tends to provide lower throughput and a message latency that is higher than the average one in asynchronous networks. Nevertheless, the worst case message latency in asynchronous networks may not be acceptable for some applications.

TDM can be easily implemented in SpW networks using time-codes. There are sixty-four possible time-code values that allow up to sixty-four different timeslots to be defined. Time slot configuration may be preset or configured at network setup time. Reconfiguration should be done by a master network manager to ensure the robustness of the system.

Message latency and throughput can be adjusted by assigning more time-slots. They should be equally distributed in the epoch when they are not synchronized with the message production.

2.4 PRIORITIES

Messages usually have different levels of priority. Priority is applied at node and network level over packets containing a message.

Node level

The simplest priority mechanism that can be implemented at node level is to try to send higher priority packets before lower ones. In synchronous networks the highest priority packet within a specific timeslot is resource reserved in this timeslot. Packets with lower priority may be sent only if the higher ones do not need to use the timeslot.

More complex priority mechanisms can be implemented on the top of the previous one. For example, for asynchronous wormhole switching networks, the elastic round robin method [3] provides better fairness, as it takes into account the time used by a channel to send a message.

Network level

Some routers may perform the arbitration taking into account packet priorities. In asynchronous networks this reduces packet latency for high priorities packets but does not provide deterministic timely delivery.

For synchronous networks without a schedule table the following mechanisms may provide determinism using priorities at network level:

a) Master arbitration approach, somehow similar to MIL-STD-1553, based on a master controller periodically polling the terminals.

b) Master arbitration approach in which all nodes are required to send a request to the network master before performing any transaction.

2.5 SEGMENTATION AND ENCAPSULATION

Message segmentation is required to ensure that packets sent over the network have a maximum size. This is a necessary condition to have bounded packet latency and deterministic delivery. Message segments are encapsulated in the transport protocol packet (SpW-RT). The message segments should not contain SpW routing information as this is already handled by the transport protocol.

2.6 END TO END FLOW CONTROL

End to end flow control ensures that there is always space at the destination buffer before sending a data packet. Therefore packet blocking cannot occur when the sender sends data faster than the receiver can process. This is critical to avoid very high network congestion with wormhole switching.

End to end flow control is also important for applications that already provide memory control capabilities, like RMAP, but cannot process multiple requests in parallel. The mechanism guarantees that an application is not deadlocked because of this limitation. When necessary it ensures that a high priority application waits until an ongoing low priority application is processed.

Finally, end to end flow control provides a simple detection mechanism for the sender entity when the destination application is busy or unavailable.

2.7 NETWORK MANAGEMENT

To enforce robustness and reliability, an intelligent node called network manager should be responsible for the whole operation of the network. The network manager may periodically poll routers and nodes to obtain their status, requests and possible notifications. In a scheduled network nodes are usually not allowed to send unscheduled error reports and may need to wait to be polled for the network manager, which should have the most complete and updated network information. Multiple redundant network managers may be present to detect and recover from an error, for example in case the active network manager stops sending Time-Codes.

The network manager is also responsible for configuring the routing tables and required channels parameters. It may also implement FDIR techniques and the Plug and Play protocol.

3 SPW-RT PROTOTYPES

Two software prototypes have been developed for the PC platform using SpW interfaces. The first one is a complete implementation of the first draft of the SpW-RT protocol [4]. It proved that the underlying concepts where valid and it provided and

approximation of the performance figures expected. The second prototype was developed to try as many new techniques as possible, making the protocol more efficient but still providing a complete working protocol. The most important results were:

- Bidirectional channels instead of unidirectional channels. It allowed implementing piggybacking for the user data, the acknowledgement and the flow control. It made possible to execute RMAP commands with acknowledgement using only two packets.

- Connection-less protocol. Sequence numbers were reset with a special flag.

- Support for dummy receivers with zero configuration. The return path information was embedded in the packet structure.

- For synchronous systems, the destination buffer space of multiple channels was encapsulated in a single packet to increase the protocol efficiency.

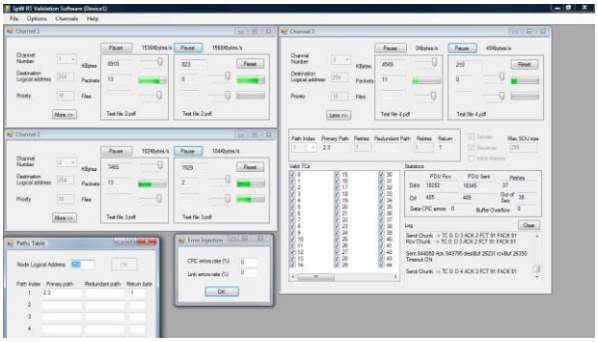


Figure 2: Protocol application layer tool developed to test the SpW-RT prototypes. It can send and receive multiple files concurrently in asynchronous and scheduled networks. It has error injection and the data rates for the generation and consumption of user data can be adjusted dynamically.

4 LESSONS LEARNED

The following considerations were derived from the prototype work and further theoretical analysis:

- Forcing the asynchronous and scheduled networks to have the same unique packet structure for control PDUs increases the complexity of the protocol and reduces its efficiency.

- Bidirectional channels are not efficient for scheduled systems. Moreover, they are more difficult to handle for the final user.

- A connection oriented protocol provides more robustness for high critical applications.

- For better efficiency in scheduled networks, flow control should be send by the receiver just before the sender performs the arbitration. The acknowledgement should be received as soon as possible. Therefore piggybacking the acknowledgement and flow control is not optimum.

- In scheduled networks, sending multiple data packets per timeslot increase the protocol efficiency. 256 bytes per segment and six data packets per timeslot are the optimum values at 200Mbit/s link speed. The efficiency is increased if the data packets are encapsulated in a single SpW packet. Note that typically only one destination per source node is available for each timeslot.

- In order to distinguish between different error cases, it is recommended to send an acknowledge packet even if a SpW error end of packet marker is received. Of course, the sequence number would not acknowledge the data packet with errors.

5 CONCLUSIONS AND FUTURE WORK

SpW-RT prototyping efforts have played an important role in the design of the SpW-RT protocol, providing important information about the advantages and disadvantages of different strategies.

Future efforts will target the prototyping of the lastest version of the SpW-RT protocol for EGSE and space qualified components. Moreover, network design tools and reference architectures with different user cases will be developed to facilitate the deployment of systems based on SpW-RT.

6 REFERENCES

[1] ECSS, "SpaceWire: Links, nodes, routers and networks", ECSS-E50-12A, January 2003 SpW RMAP

[2] S.M. Parkes, "SpaceWire-RT Requirements", SpaceNet Report No. SpW-RT WP3-100.1, ESA Contract No. 220774-07-NL/LvH, February 2008

[3] S.M. Parkes and A. Ferrer-Florit, "SpaceWire-RT Initial Protocol Definition", Draft 1.0, SpaceNet Report No. SpW-RT WP3-200.1, ESA Contract No. 220774-07-NL/LvH, March 2008.

[4] S.S. Kanhere, H. Sethu and A.B. Parekh, "Fair and Efficient Packet Scheduling Using Elastic Round Robin". IEEE transactions on parallel and distributed systems, vol 13, no 3, march 2002.

TIME-TRIGGERED SERVICES FOR SPACEWIRE

Session: Networks and Protocols

Long Paper

Wilfried Steiner, Reinhard Maier
TTTech Computertechnik AG, Vienna, Austria
David Jameux
European Space Agency ESTEC, Noordwijk, The Netherlands
Astrit Ademaj
Vienna University of Technology, Vienna, Austria
E-mail: wilfried.steiner@tttech.com, reinhard.maier@tttech.com, david.jameux@esa.int, ademaj@vmars.tuwien.ac.at

ABSTRACT
In a real-time network Time-Triggered Services allow a set of individual components to work as a coordinated whole with two main resulting benefits: firstly, a strong system-wide determinism is established and, secondly, the given physical resources can be high-efficiently utilized. The clock-synchronization service is a core time-triggered service that brings the local clocks of the individual components into agreement. The synchronized local clocks can then be used to trigger system-wide coordinated actions, such as the transmission of messages, which are then said to be time-triggered. In addition to time-triggered communication only, the synchronized local clocks can also define intervals in which event-triggered communication is allowed, which even enables mixed real-time non-real-time communication on a single physical network.

1 INTRODUCTION

Modern computer network architectures introduce dedicated network components like routers to reduce the number of communication links in the system. Nodes will then connect to a router, for example, instead of connecting directly to each other with individual point-to-point communication links. Besides the obvious weight and cost reduction of this architectural approach, additional media access logic has to be realized in order to establish a mutually exclusive access of the nodes to the communication links, which become a shared network resource.

In the simplest form, the media access logic implements an event-triggered principle, in which a node is free to access the network at arbitrary points in time and in which the nodes are serviced on a first-come first-served basis. An immediate drawback of this event-triggered principle is the cumulative transmission delay and jitter, when several nodes need to communicate onto the same shared communication link. The time-triggered principle constitutes a media access logic that uses a system-wide synchronized time-base to provide coordination between nodes in a distributed computer system, such that transmission delay and jitter are kept within low bounds.

This paper presents the outcome of an ESA-funded study on investigating the general applicability of time-triggered services for the SpaceWire protocol as well as identifying resulting constraints on SpaceWire Nodes and Routers. As a general outcome of this study we conclude that time-triggered services seamlessly integrate with the SpaceWire protocol which already provides synchronization primitives, so

called Time Codes that can be leveraged to establish a system-wide synchronized time-base.
A communication network consists of end systems that are connected via communication channels. Communication channels usually consist of passive wires and network components. In the case of SpaceWire the communication network consists of SpaceWire end systems (also called communication nodes) and SpaceWire Routers. SpaceWire is a communication protocol that defines low-level communication paradigms. The objective of this paper is to conceptually discuss how SpaceWire could be extended via time-triggered communication services and to identify possible constraints and restrictions in the specification of SpaceWire Links and SpaceWire Routers.

2 TIME-TRIGGERED SERVICE CLASSES

The number of time-triggered communication protocols is increasing and while the time-triggered protocols differ significantly in the algorithms they implement to realize time-triggered communication, there is a common set of problems that has to be solved. We call this common set of problems the Time-Triggered Service Classes.

Scheduled Dispatch Service Class: This class specifies methods for time-triggered dispatch of messages according to an off-line specified schedule table. This includes the representation of the schedule in the components, e.g. how the schedule is stored in local memory.
Clock Synchronization Service Class: This service class represents services that ensure that the local clocks of the components in the communication infrastructure stay synchronized to each other once synchronization is established.
Startup Service Class: The startup service class covers methods and services to initially synchronize the components in the communication infrastructure. This can be a coldstart procedure or an integration/reintegration procedure.
Clique Detection and Resolution Service Class: This service class defines measures that detect clique scenarios. These are unintended scenarios where disjoint subsets of components are synchronized within the subset but not over subset boundaries. Clique Resolution services define methods that re-establish synchronization when cliques have been formed and detected
Membership Service Class: Membership services are low-level diagnosis services that continually monitor the system's health state. In particular such services could reflect which end systems are present in the systems and which are not – for example because of transient/permanent failures.
External Synchronization Service Class: This service class specifies methods that allow the communication infrastructure to synchronize to an external time source.
Configuration and Maintenance Service Class: This service class defines services on how a communication infrastructure can be configured and maintained. Such services include for example configuration download procedures.
Dataflow-Integration Service Class: This service class defines measures on how message classes with different characteristics can be integrated such that all those message classes can use the same physical medium. In particular the integration of event-triggered and time-triggered messages classes is of interest in this service class.
Legacy Service Class: Existing protocols have interoperability requirements. This service class aims to identify these requirements and provide glue functionality to allow interoperability.

Integrity Service Class: This service class defines services that enhance the integrity of the communication infrastructure. In particular we are interested in two types of integrity measures: a guardian measure that can be central, local, or both, and end-to-end arguments, such as sequence numbers and timestamps.
Availability Service Class: This service class defines services that enhance the availability of a communication infrastructure. Such services include redundancy management of communication channels and redundancy management in case of fault-tolerant computation entities such as TMR configurations.

The complexity of the actual services that are realized for the service classes above, heavily depends on the system requirements. A master-based system, for example, will allow the realization of very simple services, and a single function may be sufficient to address multiple service classes at the same point in time. A master-less system will require services to be realized in form of distributed algorithms, which are inherently more complex. On the other hand, master-less systems provide higher system reliability as the failure of a single device will typically not result in an overall system loss.

This paper discusses the Clock Synchronization Service Class, the Scheduled Dispatch Service Class, and the Dataflow-Integration Service Class in particular for compliance with SpaceWire. More service classes are discussed in the final report of the study "Time-Triggered Techniques for Quality of Service over SpaceWire" [1].

3 CLOCK SYNCHRONIZATION SERVICE CLASS

This service class represents services that ensure that the local clocks of the components in the communication infrastructure are synchronized to each other once synchronization is established.
Each oscillator, as a physical component, has slightly different characteristics. One of these characteristics is the *Drift Rate*, which is defined as the difference to an oscillator perfectly aligned with real-time. Note that in this context of real-time data communication and distributed control, relativistic effects in time are not considered. According to Kopetz [2], typically Drift Rates are in the range of 10^{-2} to 10^{-7} sec/sec. It is the aim of the clock synchronization service to compensate for this inherent drift of local clocks. One straight forward clock synchronization services is Master-Slave, which off-line declares a node as Master which is used as reference clock in the network. SpaceWire inherently supports Master-Slave via SpaceWire *Time-Codes*.
SpaceWire specifies Time-Codes at the character level. A Time-Code is formed by: an Escape Character (ESC), consisting of 1 parity bit and 3 control bits, and a single Data character, consisting of 1 parity bit, 1 data-control flag, and 8 data bits. The structure of the SpaceWire Time-Codes is depicted in Figure 1.

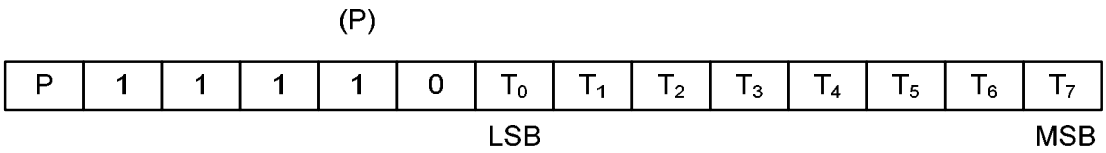


Figure 1: SpaceWire time-code

In the SpaceWire specification [3], Section 7.3(d), the Time-Code is further specified as "Six bits of time information shall be held in the least significant six bits of the

Time-Code (T0 – T5) and the two most significant bits (T6, T7) shall contain control flags that are distributed isochronously with the Time-Code."

Hence, six bits allow for sixty-four different Time-Codes. However, as a minimum only a single Time-Code is needed for time-triggered communication. In general, the number of different Time-Codes required is a function of

- the size of the communication schedule and the number of required integration points in the schedule: as the communication schedule may temporarily become long it may be required that a node can integrate at specified points inside the communication schedule instead of only at the communication schedule start, and
- the required precision in the system: again, as the communication schedule may temporarily become long, it may be necessary to schedule multiple Time-Codes for re-synchronization of the local clocks.

Time-Codes have highest priority and are transmitted interleaved with the regular dataflow. This means a Time-Code is sent immediately or immediately after the transmission of an ongoing Character is finished. Hence, on a per SpaceWire link basis, the latency jitter of a Time-Code is bounded by the maximum character in transit, which is the length of one data code (10 Bits). For the Master-Slave clock synchronization process the precision is a simple function of the latency jitter and the drift offset (where R.int is the re-synchronization interval, i.e. the duration in between two consecutive resynchronization attempts):

*Precision = Latency Jitter + Drift Offset = Latency Jitter + 2 * Drift Rate * R.int*

The end-to-end latency jitter is calculated from the link latency jitter and is discussed in the SpaceWire specification [3] under 8.12 (p) Note 2: ST.jitter = 10 N/R, where, N is the number of Links traversed and R is the average link operating rate.
The best theoretically achievable precision in the system is therefore:

*Precision = Latency Jitter + Drift Offset = (10 * N / R) + (2 * Drift Rate * R.int)*

Note, that this does not include additional Latency Jitter imposed by a SpaceWire Router as this additional Latency Jitter is implementation dependent.

A Master-Slave clock synchronization algorithm is attractive due to its simplicity and the resulting low overhead in specification, implementation, testing, and certification. On the negative side, a pure Master-Slave clock synchronization algorithm does not provide fault-masking. This means that, if the master fails (a) no time may be generated at all or (b) a malicious timeline may be generated or, (c) an interrupted timeline may be generated. Hence, if fault masking is not required the Master-Slave approach is a good solution. However, if fault masking is a requirement, Master-Slave has to be enhanced via distributed algorithms.

One straight-forward fault-masking extension is a backup clock synchronization master. This means that instead of a single node, a second node is configured with an active TICK_IN signal. Note that this is already a violation of the guideline in the SpaceWire specification that suggests assigning only one node an active TICK_IN signal. This backup master could run in warm standby or hot standby:

- Warm Standby: the backup master continuously checks the status of the primary clock synchronization master, potentially via checking if it receives valid Time-Codes. If the backup master does not receive valid Time-Codes for a specified duration it starts sending Time-Codes itself. This approach is very simple and could be argued to be in-line with the guideline suggesting only one master as there actually is only one master at any point in time.
- Hot Standby: both the primary and the backup master can send Time-Codes with an offset to each other. The identity of the primary and the backup master can be encoded via a static distribution of name space via the six least significant bits of the Time-Code (for example Time-Codes with T0 = 0 are sent by the primary, Time-Codes with T0 = 1 are sent by the backup master). As there are two sources of Time-Codes present the fail-silence failure of one of the masters is immediately masked. In order to mitigate a drift in the timeline of the primary and the backup master, these timelines have to be synchronized to each other. A simple synchronization procedure would be that the backup master always follows the primary, if the primary is present.

It has to be mentioned that the failure models that can be covered with these types of standby only cover benign failure modes like fail-silent failures of the clock synchronization masters. Failure models like babbling idiots, that are faulty nodes or faulty routers that do not fail silently, but send at arbitrary times demand the implementation of guardian instances. Guardian instances can be for example local at the node/router or centralized at a router to protect against babbling nodes.

4 SCHEDULED DISPATCH SERVICE CLASS

This class specifies methods for time-triggered dispatch of messages according an off-line specified schedule table. This includes the representation of the schedule in the components, e.g. how the schedule is stored in local memory. In a Schedule Master-based system, the Schedule Master will execute a request-response protocol: the Schedule Master sends a request to a slave, which in turn will respond with the requested information. The MIL-BUS 1553 is an example of a Schedule Master-based system. On the other hand, the schedule information can be distributed in the components and time-triggered services can be used instead of a master-driven request-response protocol. The time-triggered services ensure that the distributed schedule tables are synchronously executed. While, the time-triggered services reduce the communication overhead introduced by the request-response protocol, the Schedule Master-based approach is more flexible, as schedule changes can be done locally, without a dedicated mode change protocol.

In both approaches, Schedule Master-based, and Distributed Schedule-based, it is required that the dataflow will not be dependent on the receiver of a message. Hence, even in case of the failure of a receiver (or in case of a faulty destination address), there must be some (HW) mechanism ensuring that the messages will be delivered such that the bus/network will not be deadlocked. Therefore, non end-to-end flow control is required.

Figure 2 presents a communication network consisting of four nodes and a single router on the left hand side. On the right hand side an example of a communication schedule is depicted.

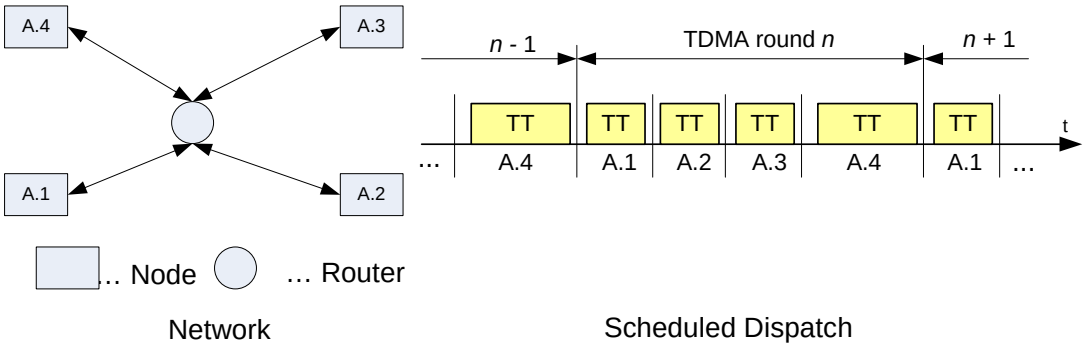


Figure 2: Communication network of four nodes, one router and communication schedule

4.1 Traditional Time-Division Multiple Access

In a protocol using Time-Division Multiple-Access, time is split up into pieces of not necessarily equal durations, which are called *slots*. These slots are grouped into sequences called TDMA rounds (see Figure 2), in which every node occupies one or more slots. The knowledge, which node occupies which slot in a TDMA round is static, available to all components a priori, and equal for all TDMA rounds. When the time of a node's slot is reached, the node is provided exclusive access to the communication medium for the duration of the slot. After the end of one TDMA round, the next TDMA round starts, that is, after the sending of the node in the last slot of a TDMA round, the node that is allowed to send in the first slot sends again. Consequently, the sending slots of each node are repeated with each TDMA round.

As any two local clocks in the network will always be slightly offset, a node may not use the full assigned timeslot for message transmission. This is a restriction that comes from bus-based networks, in which concurrent bus-accesses of different nodes must be avoided as they would result in physical signal interferences on the wire. Durations between two consecutive transmissions are called *inter-frame gaps*. The inter-frame gaps have to be chosen with respect to the precision in the system and the different propagation delays of the messages on the channels.

4.2 Novel Concepts in Time-Division Multiple-Access

The traditional TDMA as discussed in the previous paragraphs was designed for bus-based network protocols. With the movement from bus-based to network-based topologies, the rather static principle of time-triggered communication can be improved. Some of these improvements are as follows:

- Time-Triggered Multi-Cast Communication: in network-based communication topologies the switches, or routers, can be used to route messages only to subsets of nodes (or routers) instead of sending all messages in broadcast. Note that in SpaceWire, messages are only sent in uni-cast (except Time-Codes).
- Inter-Frame Gap Reduction: As discussed under traditional Time-Division Multiple-Access, in order to avoid communication conflicts, the inter-frame gap is a function of the precision (the quality of synchronization) in the communication network. In network-based networks the switches, or routers, can give priority to the transmission in progress, when a second node starts its transmission early. Hence, the router can intermediately buffer parts of the second node's communication data, until the first node finishes its transmission.

- Dynamic Slot-Assignment: a schedule master can be implemented in the network that dynamically adjusts the communication schedule to optimize the bandwidth utilization when nodes enter or leave the network. However, in space applications, modification of the communication scheme of nodes happens rarely enough (e.g. at phase change or at stage separation) and is always predictable, so that the different configurations can be considered as "static" modes.
- Support of non-harmonic message periods: in traditional TDMA, harmonic message periods can be supported. For example in TTP this is achieved by defining a Cluster Cycle that consists of a configurable number of TDMA rounds. As the TDMA round is the shortest period, this is also the highest frequency of message transmission. Messages that have to be sent with a higher period can be sent in every other TDMA round instead of every TDMA round. New schedule dispatch services also allow contention-free communication schedules with non-harmonic periods. An example of a communication schedule is given below in Figure 5. The communication schedule depicts a message with a period of 2 ms and a message with a period of 3 ms.

5 DATAFLOW-INTEGRATION SERVICE CLASS

In order to efficiently utilize the given physical resources the data-flow integration service class addresses methods that allow using the same physical network for both, scheduled time-triggered traffic (TT) and non-scheduled event-triggered traffic (ET). This service class, hence, enables payload data and command & control data on the same physical network. Similar as required for time-triggered traffic only, we assume that the delivery of a time-triggered message will never be blocked by a faulty or non-present receiver as the network would deadlock otherwise. For event-triggered messages, either readout at destination node is also ensured or end-to-end flow control must be implemented.

In this section we present three dataflow-integration services and their suitability to SpaceWire. We name these services according to the protocols that realize them, namely, the "DECOS-Approach", the "FlexRay™-Approach" and the "TTEthernet-Approach".

5.1 DECOS-APPROACH

The dataflow-integration service may specify a "tunnelling"-mechanism of ET data over TT communication: all transmissions on the network follow the scheduled-dispatch principle. ET messages are not directly sent to the network, but a middleware layer places the contents of ET messages inside a dedicated part of a TT message. We could call this mechanism "allocated transport of ET messages over a scheduled network". As the DECOS-Approach appears at the network interface as regular scheduled dispatched traffic, it is suitable for SpaceWire.

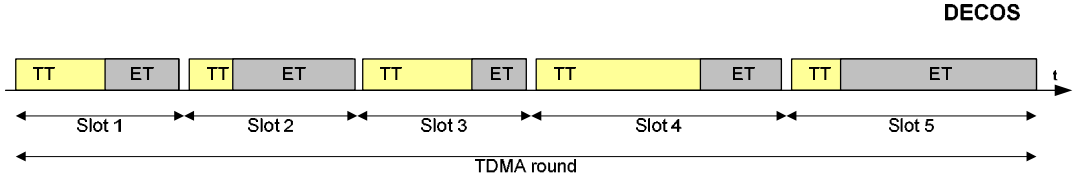


Figure 3: DECOS-Approach

The time-triggered slot may be dynamically split between time-triggered end event-triggered data or may even be fully used for event-triggered data. In the latter case we call the time-triggered slot "sporadic", which will only be used for time-triggered data

if new data is present and free for node-local event-triggered communication otherwise. We could call this mechanism “opportunistic traffic”.

5.2 FLEXRAY™-APPROACH

The dataflow-integration service may specify a “meta” time-division multiplexing approach: the bandwidth is split into (a) a static segment in which only TT messages are communicated and (b) a dynamic segment, in which only ET messages are communicated. The static and the dynamic segments are alternately executed.

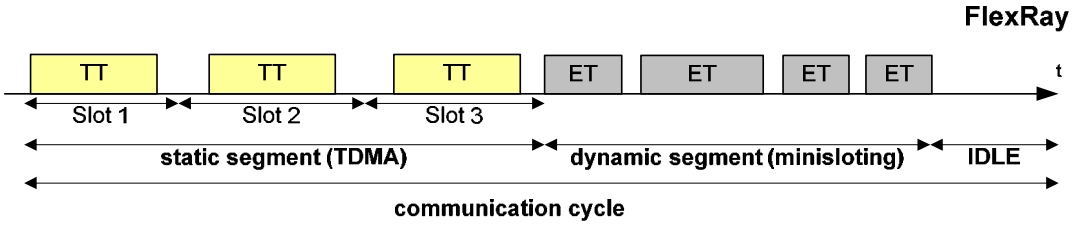


Figure 4: FlexRay-Approach

For SpaceWire, the FlexRay-Approach can be realized by offline-specifying the dynamic segment as a “long” slot in which ET communication is allowed. The start and end instant of this “event-triggered slot” can be derived from the synchronized local clocks or from dedicated Time-Codes. The latter approach is attractive, as SpaceWire nodes that communicate ET messages only do not have to be synchronized.
When a node identifies the end instant of the event-triggered slot, it may finish a pending or active transmission. However, as this increases the IDLE phase before the next static segment can be started, the node may also immediately remove pending transmissions and stop active transmissions by sending an Error end of packet (EEP) control code. In both cases, the necessary IDLE time between the end of the dynamic segment and the start of the next static segment is bounded, given the network MTU (both for TT and ET messages) and the maximum number of nodes allowed to send ET messages.

5.3 TTETHERNET-APPROACH

The dataflow-integration service is realized inside the switches in a TTEthernet network. To a TTEthernet switch, dedicated TTEthernet nodes, as well as, arbitrary standard Ethernet nodes (e.g. a laptop) can be attached. While the TTEthernet nodes execute a clock synchronization service and are therefore able to communicate TT messages, the standard Ethernet nodes will typically send ET messages only. The TTEthernet switch privileges the TT messages over the ET messages. For this a TTEthernet switch realizes two configurable modes: in case of a conflict of a TT and an ET message the switch either (a) pre-empts the ET message and relays the TT message with a constant latency, or (b) finishes the transmission of the ET message and relays the TT message immediately after the ET message (that means the TT message is treated with highest priority in case of other messages waiting for relay; as TT messages are scheduled, there will never be more than one TT message waiting for relay).

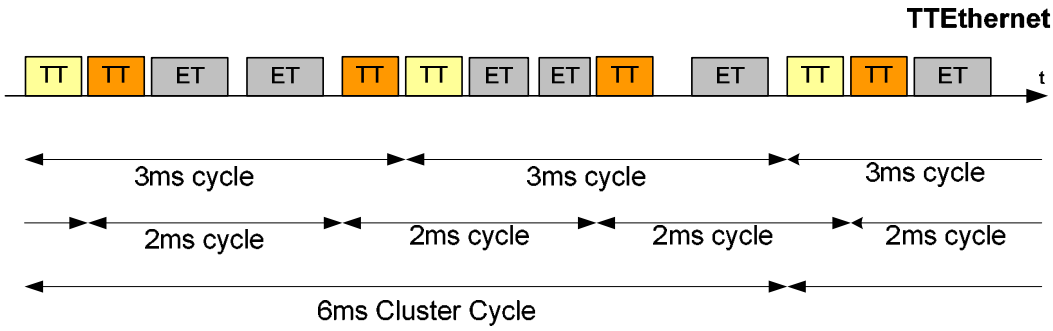


Figure 5: TTEthernet-Approach

For SpaceWire the first privilege mode will be difficult to achieve, as SpaceWire does not specify store-and-forward switching behaviour, but cut-through (also called "Wormhole Routing"). However, the second approach seems realizable, if the SpaceWire Router is able to distinguish a TT message from an ET message, which can be done by an appropriate higher layer identifier inside a SpaceWire packet.
In the TTEthernet-approach time-triggered slots can also be defined as sporadic slots in analogy to the DECOS-approach. The TTEthernet-approach is, however, more powerful, as the decision to free a time-triggered slot for event-triggered communication is done in the switch and not node-local. Hence, the sporadic time-triggered slot can be used also by event-triggered traffic coming from other nodes.

6 CONCLUSION

In this paper we discussed the feasibility of time-triggered services on top of SpaceWire. We conclude that the SpaceWire Time Codes inherently allow the realization of a Master-Slave clock-synchronization service. This clock-synchronization service is the enabler to synchronize the sending actions of the SpaceWire nodes and ensure that communication conflicts are avoided. We also discussed three approaches for SpaceWire to integrate time-triggered and event-triggered communication on a single physical SpaceWire network. We conclude that the FlexRay-approach is attractive for SpaceWire as it is simple to realize. However, it is not optimal as time-triggered slots cannot be made sporadic and, hence, be reclaimed for event-triggered. The TTEthernet-approach on the other hand would allow reclaiming bandwidth reserved for time-triggered traffic, but is more complex to realize.
From a fault-tolerance perspective, the Time-Code mechanism tolerates only simple failure modes of SpaceWire components. Fault-tolerance capabilities that go beyond simple fail-silent or detectably-faulty failure modes would cause a significant enhancement to SpaceWire as is (involving the concept of Guardian) and is not addressed in this paper.

7 REFERENCES

1. W. Steiner, R. Maier, A. Ademaj, "Time-Triggered Techniques for Quality of Service over SpaceWire", ESA Contract Number 21050/07/NL/LvH
2. H. Kopetz, "Real-Time Systems" Kluwer Academic Publishers, 1997, p.59
3. SpaceWire - Links, nodes, routers and networks, ECSS--E--50--12A, January 2003

ETHERNET FOR SPACE APPLICATIONS: TTETHERNET

Session: Networks and Protocols
Long Paper

Wilfried Steiner, Günther Bauer
TTTech Computertechnik AG, Vienna Austria

David Jameux
European Space Agency ESTEC, Noordwijk, The Netherlands
E-mail: wilfried.steiner@tttech.com, guenther.bauer@tttech.com, david.jameux@esa.int

ABSTRACT

With the growing complexity of avionics for spacecrafts it may be beneficial to rethink architectural styles in general and adopting network architectures from other application areas in particular. Common network architectures from the civil aerospace domain implement a reliable high-speed backbone bus that integrates lower-speed field-busses. This paper discusses an integrated architecture featuring Ethernet as backbone bus that integrates SpaceWire networks.

1 INTRODUCTION

On-board spacecraft computer networks have to be robust against harsh environments including high-level radiation, which can cause transient upsets, like bit-flips, in a integrated circuits. As a result, one of the driving requirements in hardware deployment for spacecrafts lies in small memory sizes and footprints, which typically leads to specific space products.

Standard Ethernet networks, on the other side, are primarily developed with a focus on consumer electronics and office requirements and do not impose said limits on memory and footprint. On the contrary, in order to avoid message drops the requirements on message buffer memory in Ethernet switches and routers become excessive.

TTEthernet closes the gap between restricting hardware requirements from space applications and excessive hardware requirements from modern Ethernet networks. The key is the introduction of a time-triggered paradigm on top of Ethernet that allows a coordinated and pre-determined usage of the resources present in the network. As a result TTEthernet is scalable for cross-domain usage, which gives a vast economic benefit and significantly accelerates the maturity process of the TTEthernet technology.

This paper gives an introduction to the TTEthernet concepts and arising benefits from TTEthernet deployment in mixed-criticality systems. Furthermore, we present integrated network architectures using TTEthernet as deterministic high-speed backbone that interconnects individual SpaceWire networks. In particular, we discuss dataflow and the flow of synchronization in such a hybrid network, where dataflow is considered in both directions, while the flow of synchronization is considered from TTEthernet to SpaceWire, if required at all.

2 OVERVIEW OF TTETHERNET

2.1 DATAFLOW IN TTETHERNET

TTEthernet specifies services that enable time-triggered communication on top of Ethernet, the TT (Time-Triggered) Services. As depicted in Figure 1, TT Services can be viewed parallel to the common OSI layers: a communication controller that implements the TT Services is able to synchronize its local clock with the local clocks of other communication controllers and switches in the system. The communication controller can then send messages at off-line planned points in this synchronized global time. These messages are said to be time-triggered and it is the task of the off-line planning tool to guarantee that the time-triggered message schedule is free of conflict. By conflict-free we mean that it will never be the case that two time-triggered messages compete for transmission and, hence, no dynamic arbitration actions for the communication medium (for time-triggered messages) are required.

TTEthernet supports communication among applications with different real-time and safety requirements over a single physical network. Therefore, three different traffic classes are provided (see Figure 1): Time-Triggered Traffic (TT), Rate-Constrained (RC) Traffic, and Best-Effort Traffic (BE). If required, the corresponding traffic class of a message can be identified based on a message's Ethernet Destination address.

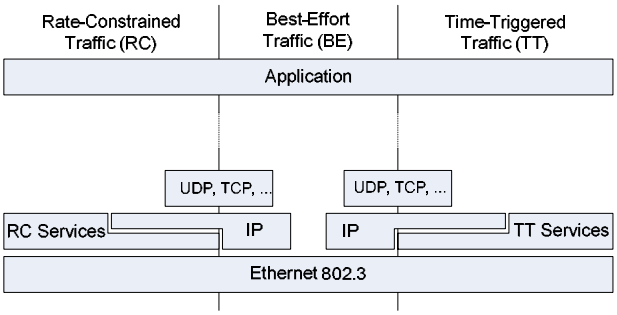


Figure 1: Interaction of standards

As depicted in Figure 1, messages from higher layer protocols, like IP or UDP, can easily be "made" time-triggered without modifications of the messages' contents itself. This is because the TTEthernet protocol overhead is transmitted in dedicated messages, called Protocol Control Frames, which are used to establish system-wide synchronization upon those components that need to be synchronized. In short, TTEthernet is only concerned with "when" a data message is sent, rather than with specific contents within a data message.

Time-Triggered (TT) messages are used for applications with tight latency, jitter, and determinism requirements. All TT messages are sent over the network at predefined times and take precedence over all other traffic classes. TT messages are optimally suited for communication in distributed real-time systems.
Rate-Constrained (RC) messages are used for applications with less stringent determinism and real-time requirements than strictly time-triggered applications. RC messages guarantee that bandwidth is predefined for each application and delays and temporal deviations have defined limits. In contrast to TT messages, RC messages are not sent with respect to a system-wide synchronized time base. Hence, different communication controllers may send RC messages at the same point in time to the

same receiver. As a consequence, the RC messages may queue up in the network switches leading to increased transmission jitters. As the transmission rate of the RC messages is bound a priori and controlled in the network switches, an upper bound on the transmission latency can be calculated off-line and message loss is prevented.
Best-Effort (BE) messages implement the classical Ethernet approach. There is no guarantee whether and when these messages can be transmitted, what delays occur and if BE messages arrive at the recipient. BE messages use the remaining bandwidth of the network and have less priority than TT and RC messages.

TTEthernet realizes the dataflow-integration service of messages of different traffic classes in the TTEthernet switches. A more detailed description of time-triggered communication and general dataflow-integration services can be found in [1].

2.2 FAULT-TOLERANT SYNCHRONIZATION STRATEGY IN TTETHERNET

In addition to a non-fault-tolerant Master-Slave clock synchronization service, TTEthernet specifies a fault-tolerant Multi-Master synchronization approach as depicted in Figure 2. In the first step Synchronization Masters send Protocol Control Frames to the Compression Masters. The Compression Masters then calculate an average value from the relative arrival times of these Protocol Control Frames and send out a new Protocol Control Frame in a second step which is used for re-synchronization of the local clocks. The central role of the Compression Master suggests its realization in the switch in the computer network, though this is not mandatory.

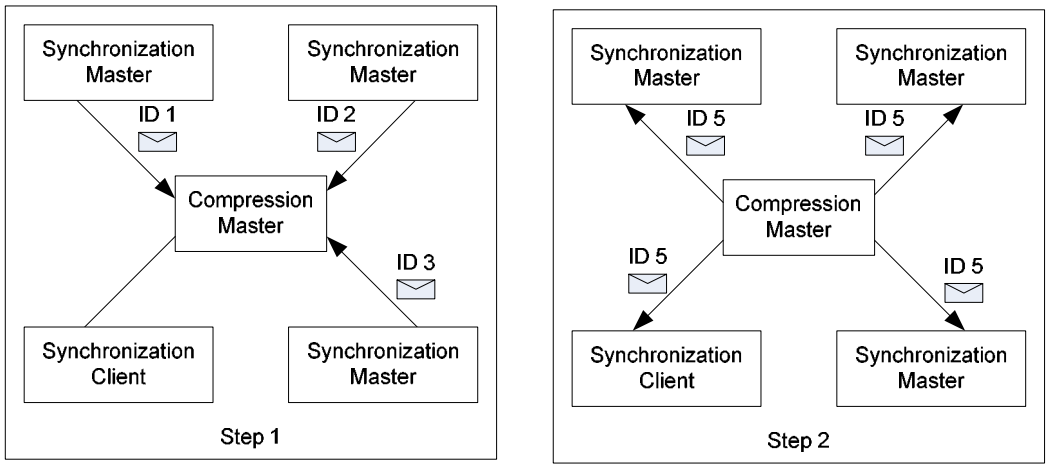


Figure 2: TTEthernet two step fault-tolerant synchronization approach

The scalability of TTEthernet from Master-Slave to Multi-Master gives a vast economic benefit: as TTEthernet can be used throughout different application domains, the cost of the realization of TTEthernet can be decreased significantly. Likewise, the cross domain usage of TTEthernet increases the probability of remaining failure detection in the realization of TTEthernet and by this matures the realization of TTEthernet significantly. This is also called ”proof-by-million” following the concept, that the probability of correctness is directly linked to the number of its implementations. Likewise the cross-domain usage contributes to the ”service history” of TTEthernet, when deployed in systems with a comparable level of criticality.

TTEthernet tolerates multiple inconsistent faults. When configured to a Multi-Master mode, TTEthernet tolerates a fully inconsistent-omission faulty communication path and even an inconsistent-omission faulty end system at the same point in time. This failure mode means that each faulty device can arbitrarily drop messages on any of its incoming communication links and on any of its outgoing communication links with potential inconsistent dropping behavior for each message. TTEthernet therefore allows a more cost-efficient realization of system architectures that require tolerance of multiple concurrent failures in the system. For example said inconsistent failure mode can even be tolerated in a system architecture that consists of only two independent communication channels. Previous realizations of communication architectures that tolerate this failure mode required at least three independent communication channels.

TTEthernet supports arbitrary multi-hop networks and even provides hooks for power-down modes of sub-networks. This can be achieved by implementing more than one Compression Master per communication channel. A system-inherent priority mechanism can be used to switch between Compression Masters, or even use the redundant set of Compression Masters per Channel as hot standby redundancy.

3 MOTIVATION FOR AN INTEGRATED ARCHITECTURE

In this section we discuss benefits of an integrated network architecture that is realized from SpaceWire, Ethernet, and Time-Triggered Services.

3.1 GENERAL BENEFITS

Design Diversity: SpaceWire and Ethernet can be argued as truly diverse network designs. This means that in the case of a design error in one of the two networks, it is highly unlikely that the same design error also occurs in the respective other network.

3.2 BENEFITS OF SPACEWIRE

Simplicity: One of the key design requirements of SpaceWire was simplicity. A simple design reduces the probability of design errors and eases the test and verification process. Furthermore, simplicity contributes to shorten the training efforts.

Efficient Memory Utilization: SpaceWire flow-control provides a natural method of high-efficient memory utilization. As messages do not have to be buffered intermediate in network components, such as routers, memories can be kept small. This, in turn contributes to a low failure rate of transient upsets.

Space-Proofed Technology: As SpaceWire is already used in active space-missions, SpaceWire-based projects can rely on well-established implementation processes.

3.3 BENEFITS OF ETHERNET

Bandwidth: SpaceWire is currently defined up to 400 Mbit/sec and typical realizations of SpaceWire will not exceed 200 Mbit/sec. Ethernet meets higher bandwidth requirements: Ethernet specifies already 1 Gbit/sec and 10 Gbit/sec bandwidths. Ethernet is, thus, suited as backbone network for individual SpaceWire sub-networks.

Group Communication: SpaceWire is a unicast protocol, besides SpaceWire Time-Codes, which are broadcasted. Hence, messages that have to be received by multiple nodes have to be repeatedly sent, leading to inefficient bandwidth utilization. Ethernet

supports multicast and broadcast communication, in which the Ethernet switches are capable of relaying the same message to several ports.

Widely-Used Standard: Ethernet is the dominating standard in office communication and current market trends show that Ethernet is getting rapidly adopted for distributed control throughout the application domains, e.g.: Profinet, Powerlink, EtherCAT in industrial controls, ARINC 664-p7 for avionics controls, or upcoming Ethernet-based solutions for automotive applications. There are multiple reasons for this trend, which are not only focusing on components or chip cost: using Ethernet allows utilizing a huge knowledge and user pool. A naïve web-search reveals 103,000,000 Google-hits for Ethernet vs. 114,000 hits for SpaceWire. Furthermore, the development for critical application domains as civil avionics demands a requirements-based development that meets the highest certification standards. Re-use of the certified components for space products certainly increases their quality.

3.4 Benefits of Time-Triggered Services

Time-triggered services establish and maintain a global time, which is realized by the close synchronization of local clocks of the components. The global time forms the basis for other system properties, as for example temporal partitioning, precise diagnosis, efficient resource utilization, or composability.

Temporal Partitioning: The global time can be used as powerful isolation mechanism when components become faulty; we say that the global time operates as "temporal firewall". In case of a failure it is not possible for a faulty application to untimely access the network. Depending on the location of the failure, either the communication controller itself or the switch will block faulty transmission attempts. Failures of the switch can be masked by powerful end-to-end arguments such as CRCs or by high-integrity designs.

Efficient Resource Utilization: The global time allows the individual components in a network to operate as a coordinated whole. This coordination directly contributes to high-efficient resource utilization. In systems that apply an event-triggered paradigm, as for example an ARINC 664-p7 network, high safety margins on resources, such as memory in switches, must be assumed. This is to guarantee that even in worst-case scenarios, when a multitude of components send at approximately the same point in time, no message will be dropped due to buffer overflows in the switches. The time-triggered paradigm allows the components to coordinate their network utilization and, hence, minimizes such safety margins.

Precise Diagnosis: A global time stamping service simplifies the process of reconstruction of a chain of distributed events. On the other side, the synchronous capturing of sensor values allows to build snapshots of the state of the overall systems.

Composability: The global time allows the specification of components not only in the value domain, but also in the temporal domain. This means that already during the design process of components, the access pattern to the communication network can be defined. The components can then be developed in parallel activities. Upon integration of the individual components, it is guaranteed that prior services are stable and that the individual components operate as a coordinated whole.

4 SPACEWIRE AND TTETHERNET INTEGRATION

Figure 3 presents four options with increasing functionality on how SpaceWire and TTEthernet may be integrated, (a) from a dataflow perspective and (b) from a synchronization flow perspective. Ethernet links are represented by bold lines, SpaceWire links by thin lines. The TTEthernet domain is represented by four switches (A…D). The SpaceWire domain is represented by four sub-networks (A…D). Within each SpaceWire sub-network, a SpaceWire Router and three SpaceWire nodes (1…3) are depicted. The interface between the TTEthernet and the SpaceWire domain can either be asynchronous (ASYNC) or synchronous (SYNC) and may require a dedicated gateway component (GW). In cases where no dedicated gateway component is present, the gateway functionality is realized within the TTEthernet switches. TTEthernet nodes are not depicted in the figure. However, for a fault-tolerant synchronization of TTEthernet (Multi-Master Mode) also TTEthernet nodes that operate as Synchronization Masters have to be present.

In addition to standard SpaceWire, we also consider SpaceWire Reliability and Timeliness (SpaceWire-RT). For SpaceWire-RT we assume that a Master-Slave clock synchronization service is realized, such that a single SpaceWire node will periodically send Time-Codes. These Time-Codes will then be used by at least a subset of nodes for synchronizing their local clocks as discussed in [1] or directly trigger a remote node to send a message as proposed as isochronous communication in [2]. Note that the objective of this example is rather to present the different integration options, than defining an overall network architecture. However, as discussed in the following (Section 4.5), communication between any two nodes in this network is possible.

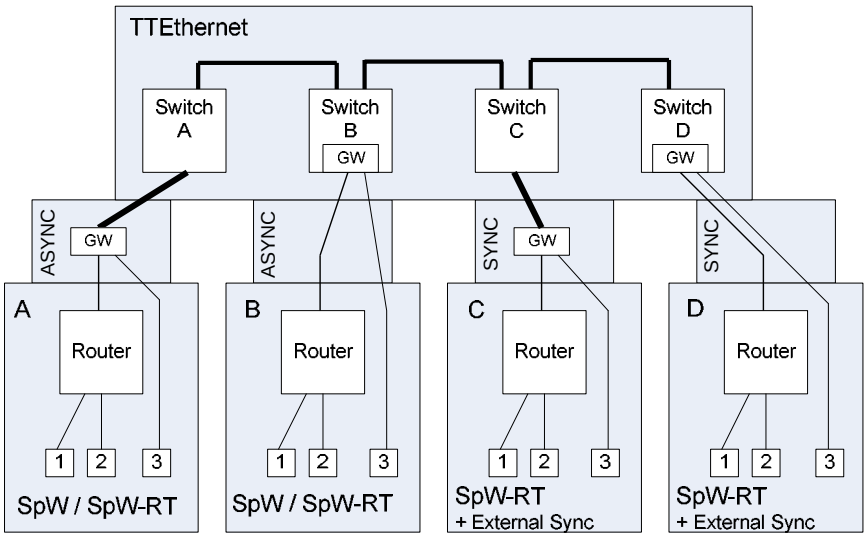


Figure 3: Four Options of SpaceWire - TTEthernet Integration

4.1 INTEGRATION OPTION A: ASYNCHRONOUS SPACEWIRE / SPACEWIRE-RT AND TTETHERNET (EXTERNAL GATEWAY)

The simplest form of integration is the implementation of an asynchronous gateway component. The functional requirements on the gateway are in the value domain of converting Ethernet frames to SpaceWire messages and vice versa.

In the temporal domain SpaceWire messages that are relayed into the TTEthernet domain can be sent as either Best-Effort (BE), Rate-Constrained (RC), or Time-Triggered (TT) Ethernet frames, where only the TT type would required the gateway

to be synchronized to TTEthernet. The temporal end-to-end guarantees from a SpaceWire node to a TTEthernet node have to be carefully examined and, of course, depend on the traffic type chosen. In case, where the SpaceWire message is mapped to Best-Effort traffic, no delivery guarantees for this message can be given, as this message will be treated with lowest priority in the TTEthernet system. In case, where the SpaceWire message is mapped to Rate-Constrained traffic, the delivery of this message is guaranteed. However, this message may be delayed by other Rate-Constrained and Time-Triggered traffic inside the TTEthernet domain, leading to high transmission latencies. In case, where the SpaceWire message is mapped to Time-Triggered traffic, the delivery of this message is guaranteed with short transmission latency and a jitter, which is significantly less than in Rate-Constrained traffic.
However, as the SpaceWire domain is not synchronized to the TTEthernet domain (in this Integration Option), the SpaceWire message will be delayed already in the gateway until the next sending slot of the gateway is reached. By assigning sending slots with short period and, thus, granting more bandwidth to the gateway, this delay can be minimized.
Furthermore, the gateway can operate as a data concentrator, which means that the gateway collects a magnitude of SpaceWire messages and forms a single Ethernet frame. Of course, this data concentration would not be restricted to messages from a single SpaceWire node, but may very well compress data from different nodes that may or may not be attached to the same SpaceWire router (e.g. messages from node 1 and messages from node 3). In principle, this data concentrator function can be also reversed from the TTEthernet to the SpaceWire domain. This may be of particular use, if only fractions of Ethernet frames communicated in the TTEthernet domain are of relevance in a respective SpaceWire sub-network.

4.2 Integration Option B: Asynchronous SpaceWire / SpaceWire-RT and TTEthernet (Integrated Gateway)

In contrast to Integration Option A, Integration Option B realizes the gateway functionality inside the TTEthernet switch device. The merge of the switch device with the gateway leads to a direct reduction of weight and power and, hence, indirectly to a reduction of cost.
From a fault-tolerance perspective we observe three emerging properties. Firstly, the system reliability increases with the reduction of the number of components in the system. Secondly, the gateway function benefits from design decisions and methods that have been chosen for the TTEthernet switch: in order to restrict the failure modes of a TTEthernet switch it can be implemented as a self-checking pair. In case of TTEthernet this is done by a so called COM/MON design, in which a switch is constructed out of a dual-chip solution. One chip, the commander (COM) acts as switching device, the second chip, acts as monitor (MON) that controls the output of the commander. When the monitor detects a failure of the commander, the monitor resets the COM/MON device, or parts of the COM/MON device, e.g. one outgoing port of the switch. Thirdly, the semantic filter and the leaky-bucket protection mechanisms present in the switch for the TTEthernet domain can be leveraged for SpaceWire. The semantic filter can analyse the semantics of a SpaceWire message and discard this message in case of semantic failures or syntactic inconsistencies (e.g. a checksum failure). The leaky-bucket mechanism checks that a minimum distance between two SpaceWire messages is respected. For example, assume that SpaceWire node 3 becomes faulty and starts to behave as babbling idiot (starts to send arbitrary messages). The leaky-bucket mechanism will silently discard messages if they arrive

too close to each other at the gateway and, thus, prevents a monopolization of the network by the faulty node 3. Note, that in such a configuration, the TTEthernet switch can be seen as Central Guardian for a SpaceWire network (in case where the messages from node 3 are sent back to node 1 and 2).

4.3 Integration Option C: SpaceWire-RT with External Synchronization and TTEthernet (External Gateway)

The gateway functionality can even be extended by an external clock synchronization service that synchronizes the SpaceWire domain to the TTEthernet domain. For this, the gateway will synchronize to the TTEthernet domain and send SpaceWire Time-Codes itself into the SpaceWire domain, which are aligned to the TTEthernet global time. This overall synchronization allows the most efficient usage of resources in the integrated architecture, as even the over-sampling effects as discussed in Section 4.1 are avoided.

4.4 Integration Option D: SpaceWire-RT with External Synchronization and TTEthernet (Integrated Gateway)

Again, the gateway can be integrated within the TTEthernet switch with perpetuation of the functionality discussed under the previous integration options. However, as system-wide synchronization is established, also the time-triggered protection mechanism can be implemented. This mechanism extends the leaky-bucket mechanism in that not only a minimum distance between messages is controlled, but also the absolute timing of a message, as the sending slots are a priori defined. This is, in principle, also possible under Integration Option B, but would cause the maintenance of two global times (the TTEthernet and the SpaceWire global time).

4.5 SpaceWire Tunnelling Over Ethernet

The integrated SpaceWire – TTEthernet architecture not only allows a dataflow from SpaceWire to TTEthernet and vice versa, but also tunnelling of SpaceWire messages from SpaceWire over TTEthernet to SpaceWire. This may be of particular usage, (a) to establish spatial partitioning between two SpaceWire sub-networks, and (b) allows TTEthernet to be used as high-speed backbone bus for SpaceWire.

5 Conclusion

This paper has given a short introduction to the TTEthernet technology with a focus on dataflow and synchronization. We discussed benefits of an integrated SpaceWire – TTEthernet architecture and presented four integration options. These integration options allow seamless dataflow and time synchronization between SpaceWire and TTEthernet.

6 References

1. W.Steiner, R.Maier, D.Jameux, A.Ademaj, “Time-Triggered Services For SpaceWire”, SpaceWire Conference, Nara, Japan, 2008
2. S. Parkes. “The Operation and Uses of the Time – Code”, International SpaceWire Seminar, 2003

Networks & Protocols 3

Wednesday 5 November
15:50 – 17:05

Proposal of CSP based Network Design and Construction

Session:Network and Protocol

Short Paper

Kazuto Tanaka, Satoshi Iwanami, Takeshi Yamakawa, Chikara Fukunaga*

Tokyo Metropolitan University, Hachioji, 192-0397 Japan

Kazuto Matsui

Prominent Network Inc., Tokyo, 104-0032 Japan

Takashi Yoshida

Smartscape Inc., Tokyo 151-0051 Japan

Abstract

We have designed a network router suited to be used in a SpaceWire network. The design was based on a formal method using CSP (Communicating Sequential Processes). The performance of the router has been verified by implementing it in a network with several processors [1]. In this paper we discuss the reason why we use CSP as a formal design method for the router-network system, actual design and refinement procedure. Discussions are emphasized on necessity of a formal method like CSP to construct and develop a secure system such as devices used in SpaceWire network system.

1 CSP model

As bit serial interconnect protocol adopted by SpaceWire allows us to form flexible network topology, we can construct the best optimized network system for a particular space mission. We can make also a fault-tolerant system on top of that by adding several duplicated network path rather than minimal topology. Many processors in the system should process and analyze data inputted concurrently from the front-end devices, and accurate information as a result of data analysis should be feed-backed to the front-end in a real time manner. Since a network comprises several active processors as well as measurement devices , the network system itself can be regarded as a parallel (concurrent) system with many processes. We have to, therefore, design the hardware system and application system with extremely strict care to avoid resource conflicts among processes.

*Correspondong author:fukunaga@tmu.ac.jp

Formal (design) methods are known to be very effective to establish a model based on the specification in a mathematical way and to verify the model meets the specification consequently. With a formal method we try to express a system specification in algebraic, symbolically logic or graphically theoretical way. Although each method uses different way to design a model, but usually every method has some algorithm to develop the model (namely to express the model in a different expression but consistently in a mathematical manner). The model developed in this way is used for the refinement or design verification. If we describe a model using a some formal method, and the model is verified equivalent as the original specification, then the implemented model after some refinement is thought to be verified in the design description level.

Communicating Sequential Processes (CSP) [2] is one of formal methods, and is known to be effective for description of a concurrent (parallel) system in which many processes (processors) work in parallel. We can express synchronization and concurrent processing in a simple way with CSP. In a parallel system, however, each process is described in a sequential manner, and a parallel processing between processes is achieved with synchronized communication between them with a concept of channels. Processes are shared their commonly used variables through the channel communication but they never keep shared memory. The real time condition of the system is also satisfied because of the inherent event driven nature embedded in CSP as a channel communication.

There is a verification tool to point out the existence of some failures (livelock, deadlock and/or divergence) in a system described in CSP, which is called FDR2 (Failure Divergence Refinement Version 2) [3]. With this refinement application, we can validate a model description automatically from the view point of existence of non-determenistic parts, namely livelocks or deadlocks. FDR2 is, hence, a relevant tool for system designers who use CSP for system modeling. In the following section we demonstrate the usage of the tool for our network design.

2 Crossbar switch example

There are algebraic notations (concepts) of CSP description. Among them relevant concepts are introduced here in order to follow the discussions given below;

1. parallel processing : $p \,[\![a, b]\!]\, q$ which means processes p and q are running in parallel in which common parameters for two processes are exchanged using channels a, b,

2. channel input $c?x : a \rightarrow p$ means variable x of type a is inputted through channel c, then proceeds to process p, and $c!v \rightarrow q$ means variable v is output through channel c, and then proceeds to process q,

3. sequential processing : $p \ ; \ q$ means a sequential processing, namely process q is followed after the completion of p,

4. $p \;|||\; q$ means two processes p and q are running in parallel without channel communication, this is called interleave, and $|||_{i \in T} p_i$ means the extended version of interleave in which all processes $p_i, i \in T$ runs independently in parallel, and

5. data series is expressed as $s = \langle a, b \,..\, n \rangle$, and two functions; *head* and *tail* work as $tail(s) = \langle b \,..\, n \rangle$ while $head(s) = a$.

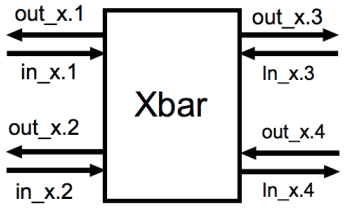


Figure 1: Four port crossbar switch with input/output channel naming

Based on these building blocks, we can design, for example, a primary crossbar switch (process xbar) that is indicated in fig. 1. In order to keep discussion simple, we assume the number of bi-directional ports is restricted to four($Tag = \{1, 2, 3, 4\}$) and the destination port number is stored in the header part of data packet ($packet = \langle port\ number, data \rangle$).

$$xbar(in, out) = |||_{i \in Tag}\, INOUT(i, in, out), \tag{1}$$

$$\begin{aligned} INOUT \quad & (i, in, out) = in.i?packet \rightarrow \\ & out.head(packet)!tail(packet) \rightarrow INOUT(i, in, out) \end{aligned} \tag{2}$$

Then we implement the actual system model as

$$SYSTEM = xbar(in\_x, out\_x), \tag{3}$$

where both $in\_x(in)$ and $out\_x(out)$ are arrays of range $Tag = \{1, 2, 3, 4\}$ and the k-th element is specified like $in.k$. The model has been checked with FDR2 and was found neither deadlock, divergent failure (livelock) nor non-deterministic process.

As the next step of the development, we have added a broadcast besides one-to-one switching. The broadcast means the same data(message) from a particular channel (i) is relayed to all the ports except the source ($Tag \setminus \{i\}$). This broadcast

can be achieved in CSP by modifying (2) as

$$
\begin{aligned}
INOUT \quad & (i, in, out) = in.i?packet \rightarrow \\
\text{if} \quad & head(packet) == 0 \text{ then} \\
& |||_{j \in Tag \setminus \{i\}} \; out.j!tail(packet) \rightarrow Skip;\; INOUT(i, in, out) \\
\text{else} & \\
& out.head(packet)!tail(packet) \rightarrow INOUT(i, in, out), \qquad (4)
\end{aligned}
$$

where *Skip* indicates that it reaches (successful) termination. We introduce the port number zero for the broadcast. We bring an "*if* ... *else*" statement to judge one-to-one or broadcast switching. In this way (4) has been naturally derived from (2). We have also confirmed the validity of this model with FDR2.

If it is necessary not to send the same message to all the port (broadcast), but to transfer it to some selected channels, we should use a cascade link of two crossbar switches as shown in fig. 2. Even if we specify the broadcast connection in the second router, we send the message to three out of five channels. The CSP description of this model can be written from (3) by simply taking into account the parallel processing of two crossbar switches as

$$SYSTEM = xbar(in\_x, out\_x) \; [\![\, Left\_ch, Right\_ch \,]\!] \; xbar(in\_x', out\_x') \quad (5)$$

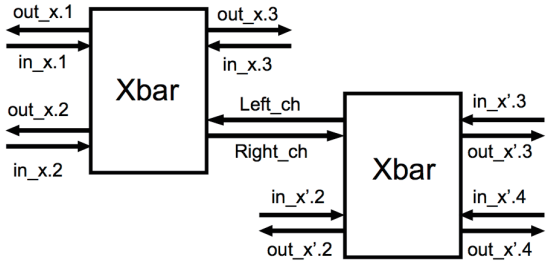


Figure 2: Cascade link of two crossbar switches. We can make one-to-many switching to the channels x'.2, x'.3 and x'.4 from any other channels with this circuit.

References

[1] K.Tanaka et al., "The design and performance of SpaceWire Router-network using CSP", presented in this conference.

[2] C.A.R.Hoare, "Communication Sequential Processes", Prentice-Hall Inc., 1985

[3] Formal System Inc., "Failure Divergence Refinement (FDR) 2", http://www.fsel.com

SoCWire: A SpaceWire inspired fault tolerant Network-On-Chip for Reconfigurable System-on-Chip Designs in Space Applications

Session: Networks and Protocols

Long Paper

B. Osterloh, H. Michalik, B. Fiethe

IDA TU Braunschweig, Hans-Sommer-Str.66, D-38106 Braunschweig, Germany

E-mail: b.osterloh@tu-bs.de, michalik@ida.ing.tu-bs.de, fiethe@ida.ing.tu-bs.de

Abstract

Configurable System-on-Chip (SoC) solutions based on state-of-the arte FPGA have successfully demonstrated flexibility and reliability for scientific space applications like the Venus Express mission. Future space missions demand high-performance on board processing because of the discrepancy of extreme high data volume and low downlink channel capacity. Furthermore, in-flight reconfigurability and dynamic reconfigurable modules enhances the system with maintenance potential and at run-time adaptive functionality. To achieve these advanced design goals a flexible Network-on-Chip (NoC) is proposed which supports in-flight reconfigurability and dynamic reconfigurable modules. The Configurable System-on-Chip solution is introduced and the advantages are outlined. Additionally we present our newly developed NoC approach, System-on-Wire (SoCWire) and outline its performance and applicability for dynamic reconfigurable systems.

1 Introduction

Configurable System-on-Chip (SoC) solutions based on FPGAs have already been successfully demonstrated in Data Processing Units (DPUs) for scientific space applications like the Venus Express mission. This approach provides the capability of both flexibility and reliability for system design. For future space missions the demand for high performance on-board processing has drastically increased. High resolution detectors with high data rate and data volume need intensive data processing (e.g. image compression) on-board because of low downlink channel capacity. A significant advantage would also be in-flight reconfigurability and dynamic reconfigurable modules for maintenance purposes and run-time adaptive functionality. To achieve such an enhanced Dynamic Reconfigurable System-on-Chip (DRSoC) approach a flexible communication architecture is needed, which provides high data transfer rates, is suitable for dynamic reconfigurable modules as well and guarantees the qualification of the system even after a module update.

In this paper we will focus on our newly developed Network-on-Chip (NoC) architecture approach System-on-Chip Wire (SoCWire). First we will introduce the configurable System-on-chip approach for the Venus Express Monitoring Camera (VMC), successfully demonstrated in space. We will then outline the essential for a

NoC and introduce our SoCWire based architecture and outline its performance, which has been measured in a demonstration implementation

2 CONFIGURABLE SOC APPROACH

Available radiation tolerant SRAM-based FPGAs (e.g. Xilinx XQR Virtex-I, Virtex-II and the Virtex-4QV series) enable the integration of special functions (e.g. data compression) and processor system completely in a single or few high density FPGAs. The major advantages of such a system are flexibility, (re)programmability and module re-use, which can be easily adapted to specific requirements. SRAM based FPGAs are sensitive to Single Event Upset (SEU) but with dedicated SEU mitigation techniques (configuration memory scrubbing, Triple Modular Redundancy TMR) the SEU rate can be reduced to negligible or at least tolerable value. The Venus Express Monitoring Camera (VMC) is one example of such a flexible SoC approach. It is based on a "LEON-2" processor, which is provided as highly configurable VHDL model, achieving a computer power of 20 Million Instructions Per Second (MIPS) in our system. Additionally the FPGA includes all peripheral logic and interfaces to different sensors and communication units (e.g. memory controller, spacecraft RTU interface logic,1355 SpaceWire interface controller)[1]. VMC science operation has been started in the mid May 2006. Since then VMC was switched on for an accumulated time of 6800 h, taking more than 103,000 images and is running well. So far only four non-correctable SEUs in the Xilinx FPGAs have been observed, which is in the expected range and shows the suitability of this approach for space applications.

3 REQUIREMENTS FOR FUTURE SPACE MISSIONS

VMC demonstrated a successful and space suitable configurable SoC approach. For future space mission advanced data processing needs to be done on-board. With high resolution detectors of >2Mpixel the data rate and data volume increases drastically but the average data rate to spacecraft remains at 20…60 kbps. Therefore classical ground processing steps like scientific parameter extraction and subsequent data evaluation need to be performed on-board. In-flight update of processing modules enhances the system with maintenance potential and performance improvement; an image compression core could be replaced by a sophisticated core to calculate scientific parameters directly on-board. Dynamic partial reconfiguration enhances the system with run-time adaptive functionality, which is an improvement in terms of resource utilization and power. Our framework for in-flight reconfigurability is based on the VMC approach with the additional features of dynamic partial reconfiguration. The FPGA is subdivided into static and Partial Reconfigurable Areas (PR-Areas) which can be updated during operation. The static area remains unchanged during the whole mission and comprises all critical interfaces (e.g. processor, communication interfaces to S/C). This offers the advantage that only the updated module has to be qualified in a delta-qualification step, not the whole system [2]. The architecture model we use for our instrument DPU designs is usually a macro-pipeline system with pre- and post-

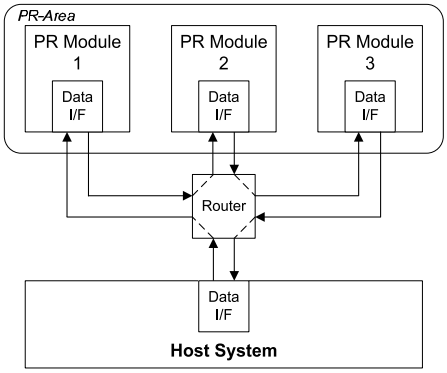


Fig.1:DPU architecture macro-pipeline system

processing steps in a dedicated structure as depicted in Figure 1. This architecture covers the typical processing chain requirements in classical instruments up to complete payload data handling.

4 PARTIAL RECONFIGURATION IN VIRTEX-4 FPGA

The Xilinx Virtex-4 Pro-V family is available as space suitable radiation tolerant FPGA. In contrast to earlier Virtex families the internal hardware architecture has changed. The FPGA is now divided into clock regions each comprising 16 CLBs, which equals one frame the smallest addressable segment of the configuration memory space. The advantage of this hardware architecture is that logic left, right, top and bottom of a Partial Reconfigurable Module (PRM) can be used for the static area [3]. Xilinx provides new un-directional Bus-Macros in the Virtex-4 family which can connect modules horizontal and vertically. These bus-macros are suitable for hand shaking techniques and bus standards like AMBA or Wishbone. As mentioned before, classically the instrument DPU architecture is a macro-pipeline system with high data rate point-to-point communication. To realize this architecture in a bus structure, multi master and bus arbitration are needed. Furthermore the partial reconfiguration process does not have an explicit activation. New frames become active as they are written. If frame bits change, those bits could glitch when the frame write is processed. Additional some selections (e.g. the input multiplexers on CLBs) have their control bits split over multiple frames and thus do not change atomically. Therefore a fault tolerant bus structure with hot-plug ability is necessary to guarantee data integrity. With this limitation a bus structure based system would encounter the following disadvantages: (i) an SEU in the PRM bus interface logic could stop the system, (ii) failure tolerant bus structure (high efforts) is needed to guarantee data integrity and (iii) during the reconfiguration process a PRM could block the bus and stop the system. With the issues mentioned before we consider instead a networked architecture with a Network-on-Chip (NoC) approach providing: (i) reconfigurable point-to-point communication (ii) support of adaptive macro-pipeline and (iii) hot-plug ability.

5 SYSTEM-ON-CHIP WIRE (SOCWIRE)

Our approach for the NoC communication architecture, which we have named SoCWire, is based on the ESA SpaceWire interface standard [4]. SpaceWire is a well established standard, providing a layered protocol (physical, signal, character, exchange, packet, network) and proven interface for space applications. It is an asynchronous communication, serial link, bi-directional (full duplex) interface including flow control, error detection and recovery in hardware, hot-plug ability and automatic reconnection after a link disconnection. These Spacewire features are ideal for our NoC approach.

5.1 SPACEWIRE

Spacewire uses Data Strobe (DS) encoding and the performance of the interface depends on skew, jitter and the implemented technology. Data rates up to 400 Mb/s can be achieved. The SpaceWire character level protocol is based on the IEEE Standard 1355-1995 with additional Time-Code distribution. The character level protocol includes data character, control character and control codes. A data character (10bit length) is formed by 1 parity bit, 1 data-control flag and 8 data bits and includes

data to be transmitted. The data-control flag indicates, if the current character is a data (0) or control character (1). Control characters (4-bit length) are used for flow control: A flow control token (FCT), end of packet markers (EOP or EEP) and an escape character (ESC) is used to form higher level control codes (8-14bit length) e.g. NULL (ESC+FCT) and Time-Code (ESC + Data character).

5.2 SoCWire CODEC

As mentioned before, SpaceWire is a serial link interface and the performance of the interface depends on skew, jitter and the implemented technology. For our NoC approach we are in a complete on-chip environment with up to 6 reconfigurable modules, which can be operated by one switch. The maximum character length in the SpaceWire standard without time code, which is not needed in our NoC, is 10bit (data character). Therefore we have modified the SpaceWire interface to a 10bit parallel data interface. The advantage of this parallel data transfer interface is that we can achieve significantly higher data rates as compared to the SpaceWire standard. Additionally, we have implemented a scalable data word width (8-128bit) to support medium to very high data rates shown in Table 1. For bi-directional data transfer every eight data character a FCT need to be included in the parallel data transfer and therefore the maximum data rate can be calculated by: *DRate[Mb/s]= fcore[MHz]*DWord Width[Bit] *7/8*. For the unidirectional data transfer the FCT can be processed in parallel and the maximum data rate is therefore: *DRate[Mb/s]= fcore[MHz]*DWord Width[Bit]*. On the other hand we keep in our implementation the advantageous features of the SpaceWire standard including flow control and hot-plug ability. Also the error detection is still fully supported making it suitable even for an SEU sensitive environment. Furthermore link initialization time could be reduced from 19,2 µs to 400 ns @ 100 MHz and link error recovery time from 20,05 µs to 1,13 µs @ 100 MHz.

DWord Width	***f_{Core} (MHz)***	***DRate [Mb/s]***	
		Unidirect.	***Bi-direct.***
8	100	800	700
32	100	3200	2800

Table 1. SoCWire CODEC data rates for given core clock frequency

5.3 SoCWire Switch

The switch enables the transfer of packets arriving at one link interface to another link interface on the switch, and then sending out from this link. The SoCWire Switch and its protocol are again based on the SpaceWire standard and is a fully scalable design, supporting data word width from 8-128bit and 2 to 32 ports. It is a totally symmetrical input and output interface with direct port addressing, including header deletion and wormhole routing. The SoCWire Switch basically consists of a number of SoCWire CODECs according to the number of ports and additional fully pipelined control machines and a cross-bar. The maximum data rate is therefore equivalent to the SoCWire CODEC. With additional SoCWire Switches and path routing the network could be extended to support even complex systems. Time Code distribution has not been implemented because it is used to synchronise a Spacewire network to a global time base. Since we are in a complete on-chip environment, each node is synchronised with the system clock and therefore Time Code distribution is not needed.

6 TEST AND RESULTS

We have implemented four SoCWire CODECs, one in the Host system, three in the PRMs and one SoCWire Switch in a dynamic reconfigurable macro-pipeline system, see Figure 1. The Host system and SoCWire Switch where placed in the static area and the PRMs in the partial reconfigurable area. All SoCWire CODECs where configured with an 8 bit data word width. The implementation of the system with reconfigurable areas could be easily implemented with the standard unidirectional Xilinx Bus-Macros. Figure 2 shows a cut out of the placed and routed SoCWire macro-pipeline system: the PRMs (PRM1, PRM2 and PRM3) and Bus-Macros in a Virtex-4 LX 60. The static area is distributed over the FPGA. The PRMs were configured as packet forwarding modules. We have tested different configuration of packet forwarding e.g. between modules, through the whole macro-pipeline system, under the condition of parallel communication between nodes. The system runs at 100MHz and the maximum data rates of the simulation could be validated to be 800 Mbps. We dynamically reconfigured one PRM in the system. During the reconfiguration process the communication between the PRM and SoCWire Switch was interrupted, the other PRMs connections were still established. After the reconfiguration process was completed the communication between the two nodes was built up automatically without any further external action (e.g. reset of node or switch). This makes the system ideal for dynamic reconfigurable systems. The Partial Reconfiguration Time (PRT) can be calculated by:

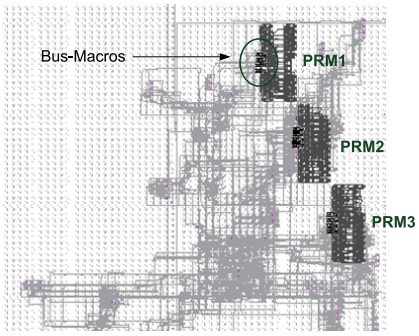


Fig. 2 SoCWire Macro-Pipeline System

$$PRT[s] = \frac{PRM[Bytes]}{CCLK[Hz] \times SelectMap_{DWord\ Width}[Bytes]}$$

The size of one PRM was 37912 Bytes (64 Bytes command + 37848 Bytes data) and therefore the PRT 758µs (SelectMap, 8Bit data word width at 50Mhz). For this test system the area for one PRM was set to utilize 0.6 % of the logic resources.

7 CONCLUSION

Configurable System-on-Chip Data Processing Units based on state-of-the art FPGAs are a proven solution for space applications. For future space applications the demands for high performance on-board processing increases enormously. Additionally, in-flight reconfigurability and dynamic reconfiguration is a further enhancement to support update of hardware functions on-board even on demand. To meet these requirements an enhanced architecture with a NoC approach is needed. In this paper we presented our NoC approach SoCWire. SoCWire meets all requirements for a high speed dynamic reconfigurable architecture. High data rates are achieved with significantly small implementation efforts. Hardware error detection, hot-plug ability and support of adaptive macro-pipeline are provided. The high flexibility of the reference design allows the designer to adapt the SoCWire system quickly to any possible basis architecture. Additional SoCWire Switches can be implemented to extend the network. With SoCWire and PR-Areas system qualification can be guaranteed (with PRM self-testing features).

8 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] B. Osterloh, H. Michalik, B. Fiethe, F. Bubenhagen, "Enhancements of reconfigurable System-on-Chip Data Processing Units for Space Application", AHS'07. pp. 258-262, Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007), Edinburgh, August 2007

[3] Xilinx, Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Functional Description, www.xilinx.com, September 2005

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

DISTRIBUTED INTERRUPTS MECHANISM VERIFICATION AND INVESTIGATIONBY MODELING ON SDL AND SYSTEMC

Session: SpaceWire networks and protocols

Short Paper

Liudmila Onishchenko, Artur Eganyan, Irina Lavrovskaya

Saint-Petersburg University of Aerospace Instrumentation. 67, B. Morskaya, Saint-Petersburg, Russia

E-mail: luda_o@rambler.ru, artfla@rambler.ru, i_lavrovskaya@mail.ru

ABSTRACT

Distributed Interrupt mechanism has been proposed for next SpaceWire standard release. Interrupt codes and Interrupt_Acknowledge codes are low-latency signaling codes, according to specification their distribution does not depend on data flow. That makes it useful for real-time distributed systems interconnections. In this paper we present: the distributed interrupts mechanism parameters and their estimation, verification by using model on the Specification and Description Language (SDL); investigation and analysis of this mechanism by using the SpaceWire Network Functional Model in SystemC and we also give some recommendations about using the distributed interrupts mechanism and choosing the timeout parameters for efficient recovery.

1. DISTRIBUTED INTERRUPT MECHANISM OVERVIEW

The detailed Distributed Interrupt description is in [1]. Interrupt-Code represents a system signal request. It is issued by a node link that will be considered as the source node for this interrupt (Interrupt Source). It is distributed over the network to all other nodes. An Interrupt-Code should be accepted for handling in some node of the SpaceWire network, which will be called the Interrupt Handler. The host of the node is supposed to implement some interrupt processing routine. One of 32 interrupt request signals (interrupt source identifiers) could be identified by the Interrupt-Code.

Interrupt_Acknowledge-Code represents a confirmation that the Interrupt-Code has reached some Interrupt Handler and has been accepted by it for processing. The Interrupt Handler node should send an Interrupt_Acknowledge-Code with the same five-bit interrupt source identifier as in the accepted Interrupt Code. The Interrupt-Code is broadcasted to find an Interrupt Handler node. To eliminate infinite cycling of the broadcasted control code specific mechanisms and rules for its handling in nodes and routers are provided:

Each link controller of a node and each router contain one 32-bit Interrupt Source Register (ISR). When the link interface receives from its host an interrupt request with a five-bit interrupt identifier it sets appropriate bit to '1' in the 32-bit ISR. Then it sends out

the Interrupt-Code with the five-bit interrupt source identifier field. If the correspondent bit in the ISR is in '1'state already then the Interrupt-Code is not sent out. A subsequent Interrupt-Code with the same interrupt source identifier can be sent by the link only after receipt of an Interrupt Acknowledge with the correspondent interrupt source identifier. In a router, when a link interface receives an Interrupt-Code it checks the correspondent bit in the ISR. If the bit is '0' it sets the ISR bit to '1' the signal propagates to all the router output ports (except the port that have issued the signal). But if the correspondent bit in the 32-bit ISR is equal to '1' the Interrupt-Code will be ignored (to prevent repeated Interrupt-Code propagation in networks with circular connections). The router shall not retransmit the Interrupt-Code to its output ports.

The Interrupt-Code handling time $\boldsymbol{T_H}$ shall be more that maximum Interrupt-Code propagation time from Interrupt Source to the farthest node (may be not Interrupt Source) in the network by the longest way in the worst case ($\boldsymbol{T_{IHmax}}$). The Time $\boldsymbol{Tg}$ between getting by Interrupt Source Interrupt_Acknowledge-Code and sending next Interrupt-Code shall be more then maximum Interrupt_Acknowledge-Code propagation time from Interrupt Handler to the farthest node (may be not Interrupt Handler) in the network by the longest way in the worst case ($\boldsymbol{T_{IGPmax}}$). Because of symmetry Interrupt-Code and Interrupt_Acknowledge-Code mechanism propagation $\boldsymbol{T_{IHmax}} = \boldsymbol{T_{IGPmax}} = \boldsymbol{T_{IPmax}}$.

In a SpaceWire network faults and errors may occur: link disconnect error or parity error can cause an Interrupt -Code/Interrupt_Acknowledge-Code loss; there may be spontaneous change of an ISR bit state as a result of intermittent faults in a node or in a router. To ensure tolerance against faults and spontaneous changes in ISR special timers are used. Each ISR in a node or in a router has a timer per ISR bit. A timer starts at the receipt of an Interrupt-Code with correspondent five-bit interrupt source identifier and resets at receipt of an Interrupt_Acknowledge-Code with the same interrupt source identifier. In case of timeout before the timer is reset, the ISR timeout event arises; the correspondent ISR bit should be reset to '0'.

2. DISTRIBUTED INTERRUPT PARAMETERS

To use Distributed Interrupt parameters we should define the following parameters: $\boldsymbol{T_{timeoutN}}$ – a timeout value for ISR in Nodes; $\boldsymbol{T_{timeoutR}}$ – a timeout value for ISR in Routers; $\boldsymbol{Tg}$, $\boldsymbol{T_H}$ – see before. This parameters depends on the network topology, router architecture and link bit rates and they affect to the latency characteristics that important for real-time. We give some expressions for them below.

Let $\boldsymbol{D}$ – network diameter (depends upon the SpaceWire network interconnection topology); $\boldsymbol{T_{bit}}$ – one bit transfer time; $\boldsymbol{T_{wtc}}$– Time-code transport through router delay (ignoring interference with previous characters/codes; depends upon implementation); $\boldsymbol{PLen}$ – the length of the longest way between any two nodes in the network (depends only on the network topology). The $\boldsymbol{T_{IPmax}}$, $\boldsymbol{T_H}$ and $\boldsymbol{Tg}$ values are: $\boldsymbol{T_{IPmax}} = (\boldsymbol{PLen} - 1)*(\boldsymbol{Twtc} + 13\boldsymbol{Tbit} + 31*14*\boldsymbol{Tbit}) + (14\boldsymbol{Tbit})* \boldsymbol{PLen}$; and $\boldsymbol{Tg} = \boldsymbol{T_H} = 1{,}5\ \boldsymbol{T_{IPmax}}$

A maximum Interrupt-Code (and Interrupt_Acknowledge-Code) time delivery $\boldsymbol{T_{Imax}}$ is:

$T_{Imax} = (D\text{-}1)*(Twtc + 13Tbit + 31*14*Tbit) + (14Tbit)*D$

In practice it is hardly ever to distinguish a maximum propagation time, but we should use it for timeout value computation. Timeout value $T_{timeoutR}$ should be less then timeout value $T_{timeoutN}$ on T_{IPmax}. It will provide, that in the node and in all routers timeouts $T_{timeoutR}$ will be over, when timeout $T_{timeoutN}$ is expired. Timeout value for router $T_{timeoutR}$ shall be: $T_{timeoutR} = 2(2T_{Imax} + 1{,}5\ T_{IPmax})$, then $T_{timeoutN} = T_{timeoutR} + T_{IPmax}$. A mean Interrupt-Code and Interrupt_Acknowledge-Code propagation time is much lower then maximum value: $T_{Imean} = (D\text{-}1)*(Twtc + 13Tbit + 2*14*Tbit) + (14Tbit)*D$

If all links in the network have different bit rates, it should be taken into account in computation of timeout values and processing time, for example to use the value of the minimal bit rate in the network.

3. EXAMPLE

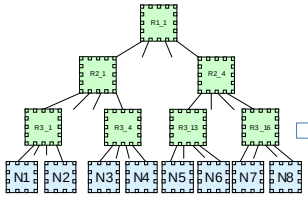


Fig. 1. Tree topology

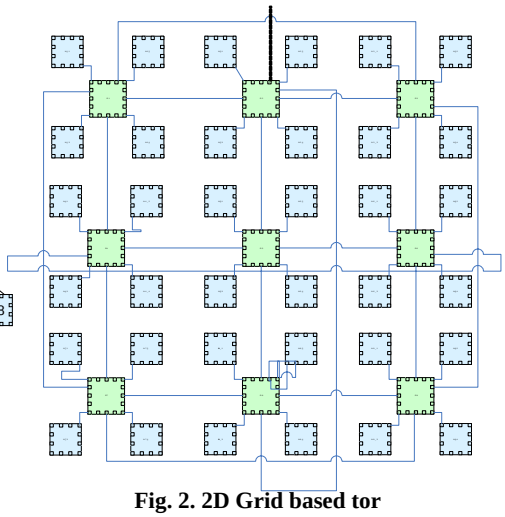


Fig. 2. 2D Grid based tor

For example we use topology shown in Fig.1. and Fig.2 At 400 Mb/s and 10Mb/s we got values (see Table 1.). First two rows correspond to Grid based tor network, last two-correspond tree network

Table 1

PLen	*D*	*Twtc*,*us*	*Tbit*,*us*	T_{IPmax},*us*	$Tg = T_H$,us	T_{Imax},*us*	$T_{timeoutR}$,*us*	$T_{timeoutN}$,us	T_{Imean},*us*
10	4	0.2	0.0025	12.2075	18.31125	4.0925	52.9925	65.2	1.0475
10	4	0.2	0.1	418.1	627.15	140.3	1815.5	2233.6	18.5
6	6	0.2	0.0025	6.7975	10.19625	6.7975	47.5825	54.38	1.7225
6	6	0.2	0.1	232.9	349.35	232.9	1630.3	1863.2	29.9

4. SDL TOOLSUITE

SDL ToolSuite gives an ability to implement specifications. A distributed interrupt system implementation on the SDL allows to have a reference implementation for it and check how Interrupt codes and InterruptAcknowledge codes are sent through the network. This model can be verified and it is very useful to check the correctness of the specification.

The model implemented on the SDL fully conforms to the distributed interrupts mechanism of the SpaceWire network. SDL model is intended for a demonstration of an

interrupt mechanism functionality and for a verification of the SpaceWire specification. This model isn't just an example of a network functionality, but it is in a strict correspondence with the specification.

The SDL model of the distributed interrupts includes the description of general elements of the SpaceWire network. These elements are a node, a router and a link. Communication between node and router is implemented as two unidirectional channels, turned different directions. Using these elements it is possible to create networks of any difficulty and construction. In the presented model it is possible to observe how the Interrupt codes and InterruptAcknowledge codes go through the network. Also there is an opportunity to model some difficult in investigation situations with lost of data, errors in links, distribution recovery in case of errors etc.Using the SDL ToolSuite it is possible to investigate distributed interrupts mechanism, check the correctness of the realization by verification.

5. INVESTIGATION BY SPACEWIRE NETWORK FUNCTIONAL MODEL

The SpaceWire Network Functional model (SpWNM) includes a description of basic SpaceWire network elements like node, routing switch and link, allows to assemble a SpaceWire interconnection system of required structure, implements wormhole routing, time flow and distributed interrupts mechanisms, generation and transmission of data packets. We use this tool for distributed interrupts mechanism investigations. In the sections 2 and 3 we calculate distributed interrupt parameters. We use them for investigation in SpWNM. The 2D Grid based tor is more complicated for distributed interrupts so it is more interesting. Some investigation results are shown below.

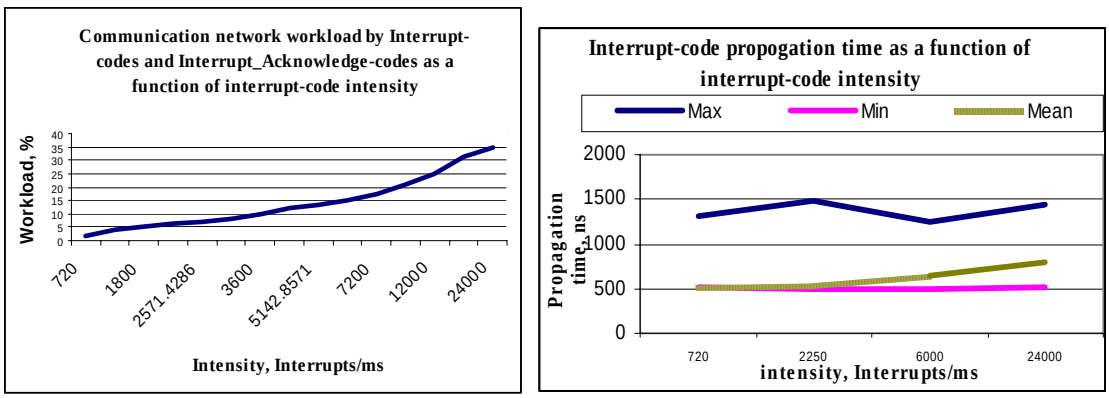


Fig.3. Communication network workload by Interrupt-codes and Interrupt_Acknowledge-codes as a function of interrupt-code intensity. Interrupt-code propogation time as a function of interrupt-code intensity

6. REFERENCES

1. Yuriy Sheynin, Sergey Gorbatchev, Liudmila Onishchenko, “Real-Time Signalling in SpaceWire Networks”. International SpaceWire Conference, Dundee 2007. Conference Proceedings. ISBN: 978-0-9557196-0-8, 4pg.

PROVIDING GUARANTEED PACKET DELIVERY TIME IN SPACEWIRE NETWORKS

Session: SpaceWire networks and protocols

Short Paper

Yuriy Sheynin, Elena Suvorova,

St. Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaya st. 190 000, St. Petersburg RUSSIA

E-mail: sheynin@online.ru, suvorova@aanet.ru

ABSTRACT

SpaceWire interconnections are used in many systems with maximal packet delivery time constraints. We consider some mechanisms for ensuring this feature in the frame of this standard, without updating the basic standard document. Delivery time for a packet depends on the packet length, SpaceWire links transmission rate, which could vary from 2 to 400Mbit/c, on the SpaceWire interconnection topology, on arbitration algorithms and buffering schemes in routing switches. We analyze characteristics of real-time packets delivery using the time-division transmission with global time-slots in SpaceWire interconnection. It is to be based on time-codes distribution mechanism in the SpaceWire standard. We estimate accuracy of global synchronization with time-codes in different interconnection topologies, its skew and jitter (time-triggered communications). Margins for time-slot specification are considered. Methods for terminal nodes distribution over the time-slots for packets transmission are considered. Interference of packets from different time-slots inside a multi-hop interconnection is estimated.

1 THE MINIMUM DISTANCE BETWEEN TWO SEQUENTIAL TRANSMISSION PHASES

The distance between two sequential transmission phases (slot (i) and slot (i+1)) typically need for avoiding situation when packets from slot i arrived to destination node too late, because they are blocked by packets transmitted in slot i+1. In worst case these packets arrives to destination node in slot (i) of next TDMA round. Also the delivery time for these packets will be essentially more than required delivery time. Let's evaluate the minimal between transmission phases in two sequential slots for time-triggered communication. (Time-codes are used for synchronisation of terminal nodes clocks.) For these systems the minimum distance between two transmission phases is equal to the maximal distance of local clocks in the system. This distance appear because in transit switches waiting time before sending Time-code for different port (depends on transmission rate and state of previous symbol sending) is differ.

Correspondingly to system tacks and conditions of usage the transmission rate in different links could be essentially differ. Also for some tacks could be used

nonsymmetrical topologies in which the ways from generator of Time-codes to other terminal nodes could have different length (different number of transit switches).
The maximal distance between sending of Time-code in one switch that is result of waiting to send previous code (Tt) is:

$$Tt = a * Tb \quad (1)$$

Where Tb – time of one bit transmission for most slowly link,
a –the number of bits in previous symbol.
The maximal length of SpaceWire symbol is 14 bits, it corresponds to Time-code, distributed interrupt code or acknowledge code. Correspondingly SpaceWire standard two Time-codes could not go to one port without big time interval. But the distributed interrupt code or acknowledge code could directly precede to Time-code. Also the value of coefficient a will be 14 for systems where the distributed interrupts and acknowledge codes are used, or 10 for systems without these codes.
The maximal delivery time for Time-code could be evaluated with using of next formula:

$$Ttd\max = 14 * Tb_0 + \sum_{i=1}^{N} (Ts + Tt + 14 * Tb_i) \quad (2)$$

where Ts – the Time-code processing time in transit switch
Tbi – the time of one bit transmission in transit link,
N – number of transit switches in the path
The minimal delivery time of Time-code to destination node could be evaluated by next formula:

$$Ttd\min = 14 * Tb_0 + \sum_{i=1}^{N} (Ts + 14 * Tb_i) \quad (3)$$

Then the maximal distance between time codes arrival times for different terminal nodes could be evaluated as:

$$Tt\max = \max_i (Ttd\max) - \min_i (Ttd\min) \quad (4)$$

If Time-codes source node could be source of packets then

$$Tt\max = \max_i (Ttd\max) \quad (5)$$

This parameter determines maximum distance of local clocks in the system and the minimal distance between two sequential transmission phases.

2 INTERVAL BETWEEN START OF LAST PACKET TRANSMISSION AND END OF TRANSMISSION PHASE

Let's evaluate the interval between start of last packet transmission and end of transmission phase. This time is function of packet length (the length of last packet in current slot) – Zp (the number of symbols), of transmission rate in transit links (Tb), of header processing time in transit switches (Tsh), of transit switches number (N). If in one slot the transmission between some pairs of terminal nodes is allowed and transmission paths includes shared links then in corresponded transit switches we need to the – Ta. When the transmission rates of all links of the path are equal the maximal header of packet transmission is:

$$Th = 10 * Tb + \sum_{i=1}^{N} (Tsh + Ta + 10 * Tb) \quad (6)$$

The maximal interval between arriving of header and of packet end to destination node (Td) could be evaluated by next formula:

$$Td = (Zp - 2) * 10 * Tb + 4 * Tb \quad (7)$$

The whole transmission time could be evaluated by next formula:

$$Tp = Th + Td \quad (8)$$

This expression not includes time of waiting in case of header wait when transmission of previous symbol is finished. If packet length is more than 10 – 15 symbols this wait time is negligibly small. If transmission rate in different links is differ then

$$Th = 10 * Tb0 + \sum_{i=1}^{N} (Tsh + Ta + 10 * Tbi) \quad (8)$$

The time interval between arriving of the first and the last symbol of packet to destination node is strongly depends from transmission rate in most slowly link in the path. It could be evaluated by formula (7) when Tb=max(Tbi) for all links included in the path. Also in systems with different link transmission rates in some cases we need to take into account the wait time for sending the NULL inserted because the data symbol is not ready. This wait time is important if data rate of output link is bigger than data rate of input link in less than 2 times. For such transmission rates the NULL symbols will be often appear before data symbol and interval between starting of NULL transmission and time when next data symbol is ready to transmission is very short. As result the real transmission rate for output link will be less than for input link. The time interval between first and last symbol of packet in this case is:

$$Td = (Zp - 2) * 10 * Tbo + 4 * Tbo + 8 * Zp * (Tbi - Tbo), \quad (9)$$

where Tbi – the one bit transmission rate for input link,
Tbo – the one bit transmission rate for output link
The whole packet transmission time is:

$$Tp = Th + \max_i(Td) \quad (11)$$

The parameter Tp show the time before end of transmission phase when the source must start last packet transmission.

3 SOME EXAMPLES

Let's consider the system with three structure represented on figure 1. On this figure circles correspond to terminal nodes, the squares correspond to switches. The transmission rate of links between switches and link T0-S0 is 400Mbit/s (Tb=2,5нс). The node T0 is source of Time-codes and destination node for data flows from all other nodes. The transmission rate of links between nodes T1 – T40 and switches is 10Mbit/s. Let's suppose that Ts=50ns. If T0 is not packet source Ttmax= 1025 ns, in other case Ttmax=2545 ns. Let's evaluate the distance between start of last packet header transmission and end of transmission phase as function of packet size for this example. Let's suppose that every node need to transmit 512 bytes of data during one transmission phase (one slot) and only one source node exists in every phase. It could send these data as one or some packets. In this article we suppose that every packet

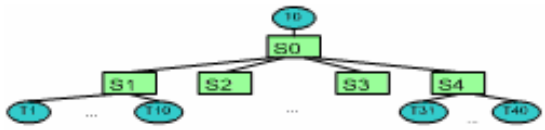
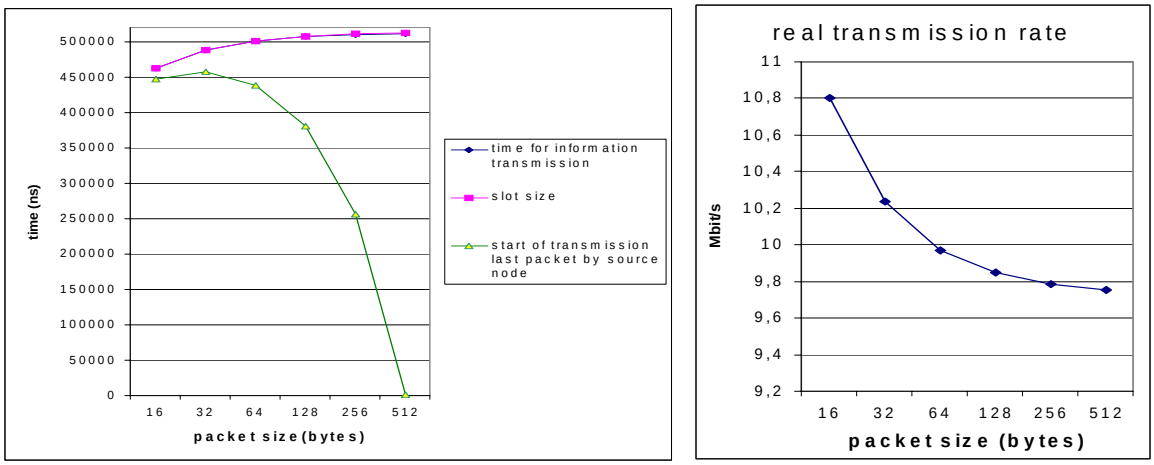


Figure 1. the example of system, based on three topology

includes only header (size is 1 byte), end of packet and data payload (we don't analyze system parameters in case of RMAP packets). On figure 2 a represented dependency between packet size and transmission phase size, whole slot size and start of last packet transmission. For this example the distance between two sequential transmission phases is very small in comparison with slot size (less, than 0.3% of slot size). The slot size grows with growing of packet size but not essentially (the difference for Zp=16 and Zp=512 is 1.1 times). On figure 2 b represented real transmission rate of link between S0 and T0. It is essentially less than physical transmission rate of this link (400Mbit/s). The length of TDMA round is from 18519000 ns to 20503000 ns dependently of packet length. Also the real transmission rate for every node is 0.28Mbit/s. For growing utilization of link between T0 and S0 we can use, for example, packet buffering in S1 – S4. This problem could not be decides only with using of different variants of slot distribution.



a) b)

Figure 2. – timing parameters for example 1

Let's consider other structure represented on figure 3. In this system T3 is Time-codes source. The transmission rate for links marked by dashed lines is 10Mbit/s, the transmission rate of other links is 400Mbit/s. The data flows represented by arrows. Ts=100 ns. For this system Ttmax=3175 ns. For this system this interval between transmission phases is not small. 126 data symbols could be sending between T3 and T2 in this period. Let's consider possible slot distributions for this system. The data paths for T6-T2, T7-T4 and T8-T5 are independent. Remaining pairs are also independent. Thus we could use one slot for every group (TDMA round will be includes two slots). But in case of the transmission time for pairs includes to one slot is essentially differ this distribution is not rational. We can divide transmission for every node pair to some slots for alignment of transmission time for every pair in one slot. But in this case the big number of slots results to bid time will be lost because of inter slots intervals. Because of these problems the global slots are not rational for non symmetrical systems.

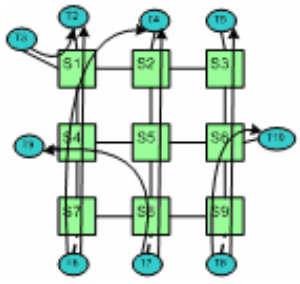


Figure 3 - example of system based on 2D-grid

4 REFERENCES

1. Wilfreed Steiner, "Time-Triggered Techniques for Quality of Service over SpaceWire". 10 SpaceWire Working Group Meeting, ESA/ESTEC February 20/21 2008

DESIGN CONSIDERATIONS FOR ADAPTING LEGACY SYSTEM ARCHITECTURES TO SPACEWIRE

Session: SpaceWire Networks and Protocols

Short Paper

Robert Klar, Christopher Mangels, Sandra Dykes, and Michael Brysch

Southwest Research Institute, San Antonio, TX, 78238

E-mail: robert.klar@swri.org, christopher.mangels@swri.org, sandra.dykes@swri.org, michael.brysch@swri.org

ABSTRACT

Since first developed and standardized, SpaceWire has rapidly gained acceptance for use in space applications. SpaceWire offers many advantages to system designers over other on-board communications technologies. It is rather simple to implement, requiring relatively few logic gates and little interface memory to implement. Built upon low-voltage differential signaling (LVDS), it provides for high-speed communications, supporting link speeds well in excess of 100 Mbps in practical systems. It uses point-to-point links to connect nodes rather than a shared-bus architecture and thus provides much flexibility for incorporating redundancy into spacecraft systems.

Prior to SpaceWire, many spacecraft bus architectures were based on communications technologies such as the MIL-STD-1553B and RS-422 serial links. This paper describes a basic design process and suggests some considerations for adapting such legacy systems to use SpaceWire.

Initially, a system designer must analyze data flows to evaluate direction, volume, criticality, and timing requirements. For time-critical data, a synchronous schedule is often preferable and can be accommodated easily by making use of the time codes feature of SpaceWire. After critical data flows have been identified, redundancy and retransmission can be used to guarantee delivery of important data. Incorporating routing switches into the system can offer much flexibility for architecting robust redundant topologies. Additional reliability can be gained by suitably applying higher-level protocols such as Remote Memory Access Protocol (RMAP) to support data acknowledgement and retransmission. We present some practical considerations from project experience.

1 INTRODUCTION

Since it was standardized by the European Space Agency (ESA), SpaceWire [1] has been proposed for use on many space missions [2]. SpaceWire is a low-power, high-speed communications technology that was designed to be simple to implement in digital logic. It is rapidly replacing widely prolific legacy communication technologies such as RS-422 and MIL-STD-1553B. Since designs are often reused across several missions to reduce development costs and schedule, it is important to consider the characteristics and advantages of these legacy systems when moving

towards a SpaceWire architecture. This paper describes a basic design process and suggests some considerations for adapting such systems to use SpaceWire.

2 LEGACY ONBOARD COMMUNICATIONS SYSTEMS

Originally published by the United States Department of Defense as "Military Standard Aircraft Internal Time Division Command/Response Multiplex Data Bus" [3], MIL-STD-1553B, as the name reveals, was initially developed for application in military aircraft. However, over time, it has found numerous other applications. The United States National Aeronautics and Space Administration (NASA) has found it particularly useful because of its durability and suitability for high-radiation environments. It has been widely adopted for use in command and data handling systems on numerous flying spacecraft including the recent Swift and Fermi (i.e. formerly GLAST) missions [4,5].

MIL-STD-1553B has endeared itself to system designers for many reasons. Since it is based on a shared bus topology, devices must share the transmission medium by time-division multiplexing (TDM), staggering transmission of data on the bus in time. This characteristic of MIL-STD-1553B results in a deterministic real-time schedule for data collection (i.e. the bus schedule). Spacecraft control loops are often designed around this bus schedule. In the terminology of the MIL-STD-1553B standard, the bus schedule is managed by a Bus Controller (BC). A device on the bus with which the BC communicates is a Remote Terminal (RT). MIL-STD-1553B supports half-duplex communication in which the RT responds only to commands directed to it by the BC. Although BC implementation is not defined by the standard, most modern systems use a frame controller which can process a sequence of command messages in a repetitive fashion [6].

A beneficial feature of MIL-STD-1553B is its rich support for fault-tolerance. Developed for use on military aircraft, it was designed with support for redundancy. A common configuration consists of a primary bus and a single hot spare [6]. Although the standard provides for higher levels of redundancy, these are rarely seen in modern systems because it results in reduced bus throughput. Since messages have specific timing requirements and include parity, BCs can detect errors in transmissions. To handle errors, BCs often perform one or more retries on a bus before autonomously switching over to a redundant bus.

Although well suited for command and control applications, MIL-STD-1553B is not well suited for bulk data transfers [13]. With a bus clock rate of 1 MHz, the data rate is limited to only a few hundred kilobits per second in practical use. Consequently, many spacecraft subsystems and instruments producing large quantities of data instead use a dedicated high-speed interface. Because they are relatively simple to implement, serial data interfaces are common. Until more recently, with advent of Low-Voltage Differential Signaling (LVDS), RS-422 signaling was often used to implement serial interfaces in spacecraft systems.

RS-422 commonly refers to ANSI/TIA/EIA-422-B, "Electrical Characteristics of Balanced Voltage Differential Interface Circuits" [7], or the equivalent international standard, ITU-T Recommendation V.11 [8]. As the name reveals, characteristics of RS-422 include a differential driver and receiver. RS-422 supports relatively high data rates, with practical implementations of up to 10 Mbps.

For low-speed data rates up to 115,200 baud, RS-422 is often used to implement an asynchronous serial interface. There is a variety of off-the-shelf hardware to support asynchronous serial communications with a relatively inexpensive personal computer (PC). For higher data rates, a synchronous serial interface is more common. A unidirectional configuration for transferring telemetry is often employed in spacecraft systems and might typically consist of three signal pairs: clock, frame, and data.

3 COMMUNICATIONS SYSTEMS DESIGN

When designing an onboard communications system, it is often preferable to take a top-down approach to solving the problem. A basic design process could include the following steps: 1) Identify top-level requirements; 2) Determine link rates; and 3) Design and size each link.

The first step in the process is to identify top-level requirements. One helpful technique for deriving these requirements is to construct a mission-level data-flow diagram. The diagram can then be analyzed to identify producers and consumers of data. A key part of this analysis is to identify the quantity of data that needs to be transferred on board.

The next step is to determine the required link rates. For some data producers, data rates are easily computed by reviewing the sampling rate and the number of bits in each sample. However, for others, determining data rates can be more difficult. Event detectors, for example, often produce burst data which may be difficult to characterize. In this case, an accurate data model is sometimes needed to estimate the worst case.

After data rates have been determined, the final step is to design and size each link appropriately. Although legacy interfaces may still be viable, SpaceWire can provide similar benefits and is well suited to replace them for many applications. Many vendors produce SpaceWire products and rich variety of offerings is now readily available on the market.

4 SYNCHRONIZED AND UNSYNCHRONIZED ARCHITECTURES

SpaceWire offers significant flexibility to communications systems designers in that it supports both synchronized and unsynchronized architectures. In designing a communication system, it is important to understand the specific nature of the data being produced. If the data is produced at a regular cadence, then a time-synchronized or spacecraft spin-synchronized data system might work well. On the other hand, if the data is produced in bursts or at irregular intervals, then an unsynchronized data system would likely be a better choice. Hybrid architectures are certainly possible and well supported by SpaceWire.

SpaceWire includes the capability to send a low-latency signal called a time code. In practical systems, this can be used for synchronization to an accuracy of a few microseconds without enhancements [9]. Time codes can be used to frame data into discrete time slots to implement time-division multiplexing similar to that inherently provided by MIL-STD-1553B. The time code feature has been further extended to provide even more utility. Researchers at NASA Goddard Space Flight Center (GSFC) developed an intellectual property (IP) core that provides up to 4 unique time codes that can be used to communicate side-band data as well as time [10]. Researchers at St. Petersburg University have developed a scheme for using time

codes as distributed interrupts for support of hard real-time control systems; the protocol provides for up to 16 possible unique codes [11].

High-speed serial interfaces have often been employed to provide unsynchronized telemetry to onboard data storage systems. Significant engineering time and effort can be spent developing and specifying unambiguous, detailed requirements for the required data framing. Consequently, many variations in methods for determining these requirements exist. Substituting SpaceWire for a high-speed serial interface can greatly simplify the interface definition task, as it provides a standard mechanism for defining packet boundaries.

5 IMPROVING RELIABILITY

Proper application of redundancy can improve reliability. SpaceWire provides many options for incorporating redundancy. Several are shown in Figure 1. One option is to include a router in the host system. NASA has developed a router IP core with 4 external ports and 4 internal ports that is sized to fit within a single Actel RTAX2000 with margin. This structure provides an extremely versatile building block that can be used to construct a variety of redundant topologies. For small systems, another option is to simply include redundant links. Link cores require few logic gates and are easily accommodated by modern space-qualified programmable devices. Researchers at Southwest Research Institute ® (SwRI®) developed a link core that consumes less than ten percent of an Actel RTAX2000. A third option is to use a redundant physical layer. Researchers at NASA GSFC have developed a link-and-switch implementation for SpaceWire that allows for an automatic switchover to a redundant link in the event of a link failure. In some ways, this is analogous to the automatic bus switching provided by MIL-STD-1553B.

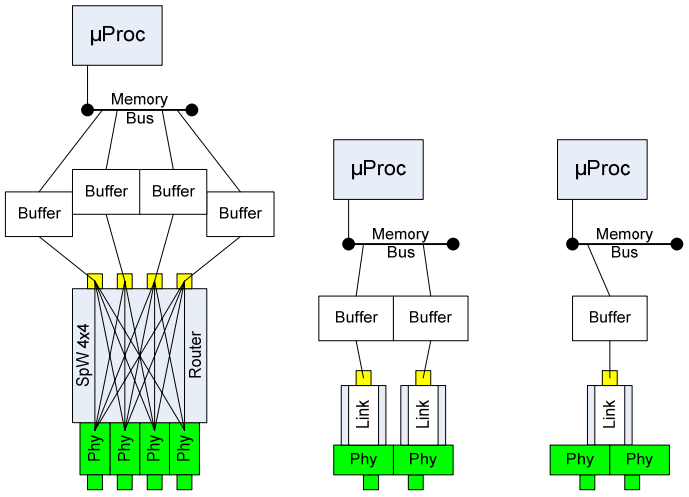


Figure 1. SpaceWire Redundancy Options

The SpaceWire Working Group (SWG) created a specification for a standardized SpaceWire encapsulation header to provide a means to multiplex and demultiplex packets of different higher-level protocols [12]. Included in this standard is the definition of the Remote Memory Access Protocol (RMAP). RMAP supports a command/response mechanism similar in many respects to the operation of MIL-STD-1553B. It provides mechanisms for acknowledged and verified data transfers. RMAP has further advantages in that it incorporates strong error-detection by

inclusion of a cyclic-redundancy code (CRC). When RMAP is used in combination with redundant links, it can emulate the operation of MIL-STD-1553B with redundant busses.

6 CONCLUSION

We have introduced some important considerations for a system designer wanting to use SpaceWire. SpaceWire can be a versatile and robust solution for constructing an onboard communications system.

7 REFERENCES

1. ECSS-E-ST-50-12C, Space Engineering: SpaceWire – Links, nodes, routers, and networks", ESA-ESTEC, July 2008.
2. ESA SpaceWire Website:
   http://spacewire.esa.int/content/Missions/ESA.php
   http://spacewire.esa.int/content/Missions/NASA.php
   http://spacewire.esa.int/content/Missions/JAXA.php
3. "Military Standard Aircraft Internal Time Division Command/Response Multiplex Data Bus," MIL-STD-1553b, United States Department of Defense, 21 September 1978.
4. "Swift 1553 Bus Protocol Interface Control Document." Spectrum Astro Inc., August 2000.
5. "Gamma-Ray Large Area Space Telescope (GLAST) 1553 Bus Protocol Interface Control Document." Spectrum Astro Inc., December 2002.
6. "A MIL-STD-1553B Tutorial," Document #1600100-0028, Condor Engineering, June 2000.
7. "Electrical Characteristics of Balanced Voltage Digital Interface Circuits", TIA/EIA-422-B, May 1994.
8. "Electrical characteristics for balanced double-current interchange circuits operating at data signalling rates up to 10 Mbit/s", ITU-T Recommendation V.11, October 1996.
9. Barry Cook. "Time-code enhancements for SpaceWire", MAPLD International Conference, 2006.
10. NASA GSFC Innovative Partnerships Program Website:
    http://ipp.gsfc.nasa.gov/ft-tech-spacewire.html
11. Yuriy Sheynin, Sergey Gorbatchev, Ludmila Onischenko. "Real-time Signaling in SpaceWire Networks. International SpaceWire Conference, 2007.
12. ECSS-E-ST-50-11C, Space Engineering: SpaceWire protocols", ESA-ESTEC, July 2008.
13. Ronnie Killough. "Integrating CCSDS and MIL-STD-1553B: What You Should Know." IEEE Aerospace Conference, December 2001.

Onboard Equipment & Software

Thursday 6 November
09:25 – 10:40

Modular Architecture for Robust Computation

Session: SpaceWire Onboard Equipment and Software - Short Paper

Dr. W.Gasti,

European Space Agency, Postbus 299, NL-2200 AG Noordwijk, The Netherlands

A.Senior

SEA, Building 660, Bristol Business Park, Coldharbour Lane, Bristol, BS16 1EJ, United Kingdom

Dr. O. Emam, T. Jorden and R.Knowelden

EADS Astrium, Gunnels Wood Road Stevenage, Hertfordshire SG1 2AS, United Kingdom

S. Fowell

SciSys UK, Methuen Park, Chippenham Wiltshire, SN14 0GB, United Kingdom

E-mail: wahida.gasti@esa.int, alan.senior@sea.co.uk, omar.emam@astrium.eads.net, tony.jorden@astrium.eads.net, richard.knowelden@astrium.eads.net,stuart.fowell@scisys.co.uk,

1. Introduction

The classical On-Board Computing System (OBCS) based on one-processor and related multi-drop bus architectures is too limiting for future space missions. The ESA ROSETTA mission has already faced this problem and it was necessary to enhance the network architecture by upgrading the OBCS with an additional processor and incorporate high speed serial links (based on IEEE-1355 Std) to respectively cope with the AOCS and payload needs. Indeed, nowadays the satellite core functions; e.g. AOCS, communication system, power system and the Payloads demand algorithms/techniques requiring the incorporation of Intelligent Units (IUs). IUs are more demanding in terms of power and computation resources and communication speeds are exceeding the capabilities of the large classical OBCS.

ESA anticipated these needs by initiating the SpW Network Technology Program several years ago. In this scope, MARC can be considered as a significant development that will provide to the space user community a robust SpaceWire based OBCS architecture that may be implemented as a Flight Qualifiable design.

2. MARC Requirements

The principal requirements are that MARC shall provide a **Robust and Fault Tolerant Computing System** based on:

- A **Modular and Scalable** network and system architecture
- A **SpaceWire network** capable of handling the future demands of both spacecraft data handling and payload data processing.
- A **hierarchical Fault Detection Isolation and Recovery** (FDIR) principles
- The **CCSDS SOIS layered software architecture** [RF3]
- Flight Qualifiable technologies such that the design can be considered as "**One Step from a Flight Model**".
- The key SpW components and standards [RF1, RF2] developed by ESA: **SpW links, SpW-10X router, RMAP.**

3. MARC FT-Design

3.1. MARC FT-Design Principles

MARC is based on a 5 layer hierarchical FDIR architecture. Each layer either handles any fault in that layer itself, or passes the fault up the hierarchy to the next layer. The layers from the highest to lowest are:

1.Ground
2.TMTC system
3.Hardware Reconfiguration Controller (HRC)
4.Master Core Computing Module
5.General Computing Module (based on COTS processor) and/or Module(s) level FDIR.

The MARC Fault Tolerant (FT) design is based on the following principles:

- No single point of failure:
- No single point of repair: If MARC experiences a single failure; it must be able to continue the handling of key functions without interruption during the repair process by provision of two

identical instances of the same SpW-module and switching (via SpW-Routers) to one of the remaining instances in case of a failure (hot redundancy failover).

- Fault isolation to the failing component, module;
- Fault containment to prevent propagation of the failure;
- Availability of autonomous safe computing system mode and reversion to operational modes on external command;

3.2. MARC FT-Design

To fulfil the MARC requirements and the FT-design principles, MARC has been built starting from a set of HW and SW Building Blocks (BBs). These BBs have been designed to insure the robustness of MARC and are summarised in the following subsections.

3.2.1. MARC Hardware Building Blocks

MARC Highly Reliable HW Components

ESA initiated and funded a set of ASSPs (Application Specific Standard Product) which are devices developed in the same way as an ASIC with the intention to be widely used and covering multi-mission needs. MARC modules are based on 2 ASSPs they are the LEON2FT based processor (AT697F) and SpW_10X router i.e. (AT7910).

MARC Cluster & HW Architecture

The MARC SpaceWire network architecture is based on a building block called a High Flexibility Cluster (HFC) which comprises 2 routers and 4 Modules. The modules may be any function (processing, memory, I/O etc) that requires a network interface. The two 8 port routers provide redundancy and each module interfaces to both routers to allow for failure of one router within the cluster.

The HFC has 4 spare ports, each port comprising a redundant SpW link. These spare ports may be used to connect clusters together or to connect to other system components such as the TMTC system or EGSE.

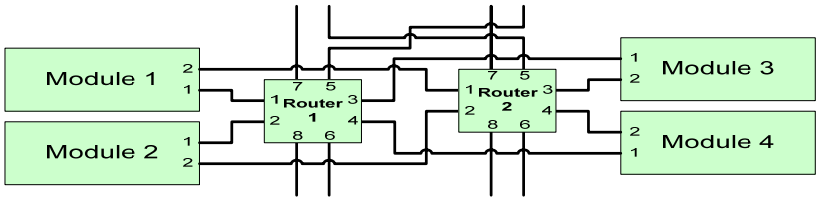


Figure 1: **MARC SpW High Flexibility Cluster (SpW-HFC)**

The HFC is in effect a 4 port router with failure tolerance and thus multiple HFCs may be connected together in any network topology. The simplest HFC based topology is to stack the clusters vertically as shown in Figure 2. If an increased bus bandwidth is required then the number of 8 port routers within a cluster may be increased to 4 or more as shown in Figure 3.

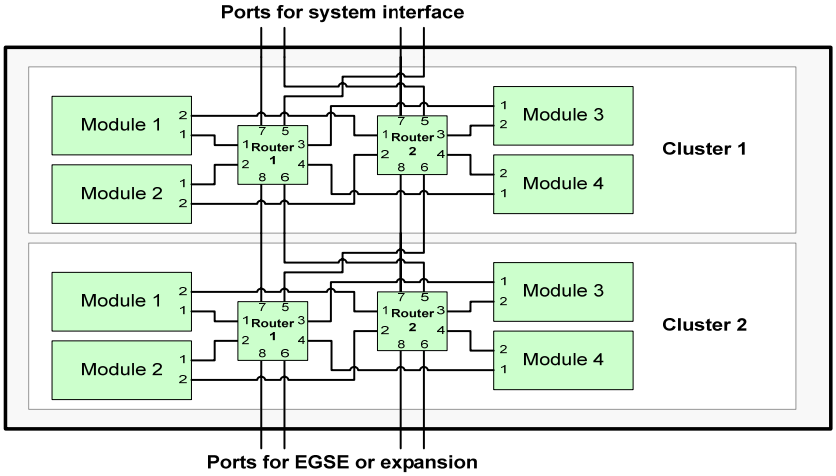


Figure 2: **SpW-HFCs Vertical Assembly**

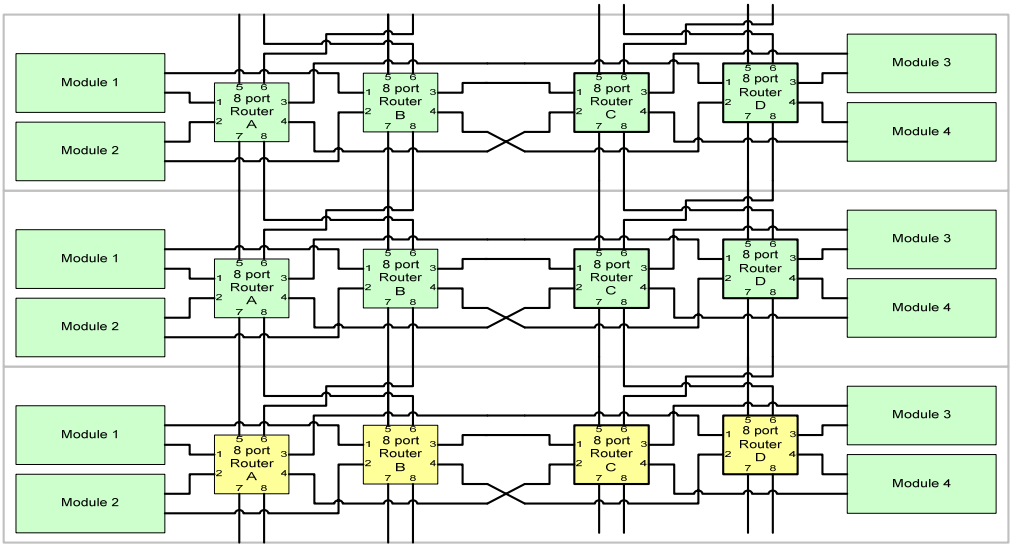


Figure 3: **SpW-HFCs Lateral Assembly**

To fulfil basic Platform and Payload computing needs, MARC can typically be built out of five types of SpW-modules that are:

- Telemetry/Telecommand Module (TTM),
- Core Computing Module (CCM)
- Memory Module (MM)
- I/Os Module (IOM)
- General Computing Module (CGM).

The related SpW-Network is configured and monitored by the master CCM. The SpW-Network is implemented using 8 port SpW-routers that are part of the backplane.

MARC network monitoring will make use of 2 types of heartbeat function as follows:

- SpW-Private Heartbeat Function (SpW-PHF) for each module which monitors its health and status in the cluster. Its report (which can be tailored down to a minimum "I am alive") is under the responsibility of the module and it shall be send regularly to the MARC master CCM.
- SpW-Network Heartbeat Function (SpW-NHF) that is provided by the MARC backplane Hardware Reconfiguration Controller (HRC) which is responsible for ensuring that critical Modules are powered (according to a configuration table), monitoring the health and status of each CCM and the distribution of SpW time codes.

The management of failures will be handled in a hierarchical manner, such that resolution is sought from the lowest possible level (Module) to the highest local system level as follows:

1. Module level (local FDIR e.g. watchdog timer): Module takes internal action (e.g. reboots);
2. SpW-PHF from Module to CCM: CCM takes action if heartbeat is missing;
3. SpW-NHF from CCM to HRC: HRC takes action if heartbeat is missing

RMAP for SpW-Module memory access

All MARC modules have at least two SpW interfaces for redundancy and these incorporate RMAP functionality (via ESA-RMAP-IP-Core). So, for example, it is possible for the CCM to access memory banks housed in any module as an extension of its own memory. The CCM can access software and data resources from at least 3 sources, i.e. local, another CCM or the Mass Memory. Although simple, this memory access capability is very efficient in coping with the Prime CCM failure and permits context saving memory access by the redundant CCM. The access is possible without any system reconfiguration. At configuration and the system initialisation phase, it is possible to distribute boot SW, context saving and data mission storage to the available memory banks of MARC. Nevertheless, each CCM has its own SW boot capability in case MARC is used only for platform command/control application.

3.2.2. MARC SW Building Blocks

MARC SW Architecture: GenFas

GenFas is based on the CCSDS-SOIS layered SW architecture [RF3] using the LEONFT2 processor of the CCM. GenFas is relying on SpW Real-Time Communication Protocol (SpW-RTCP). SpW-RTCP communication model implying:

- a virtual point-to-point connections across SpW network: Virtual points address either source or destination and are represented by channel buffers. A user application (highest SOIS Layer) writes

into one of the source channel buffers available and related data packet is then transferred across the SpW network and becomes available in the corresponding destination channel buffer.

- a scheduled communication: Transmission of data packets is synchronous with each source channel being assigned one or more time-slots when it is allowed to transmit packet(s). Timeliness of delivery is controlled by a schedule table used to specify which source channel can send packet(s) in which time-slot. This provides deterministic delivery.

MARC SpW-RTCP provides 3 QoS according to CCSDS standard definition that are: Basic, Best Effort, Assured.

MARC FDIR

MARC implements a hierarchical FDIR management architecture detecting, isolating and determining the recovery strategy of failed MARC modules, SpW-Routers and SpW-links with failover strategies or switching to a safe computing mode. The hierarchy in terms of computing capability shall consist of 1st class reliable computing modules, i.e. the CCMs, that have the role of MARC system supervisor as well as their own FDIR capabilities, and 2nd class less-reliable computing modules, i.e. the GCMs that are limited to only managing their own FDIR capabilities and only act as a servant to the system supervisor. This hierarchical FDIR strategy is providing Fault Containment Regions (FCRs) possibly confined to modules and/or clusters, with faults autonomously handled or escalated at each level in the hierarchy (with module and system as the minimum defined levels).

MARC FDIR Manager is GenFas embedded software as advocated by SOIS standard. FDIR Manager uses tables which define the initial system configuration and the alternative configurations to be used in response to one or more identified failures of a SpaceWire link, router or system module. The FDIR tables (FDIR-Tabs) are generated by an analysis tool which is conceived to run off-line during the architectural design phase of MARC system. The FDIR Manager Strategy is achieved this by performing its four principal functions that are:

- *Health Check Monitoring*: Health check monitoring is a regular and continuous activity performed through the exchange of Health Status Messages (HSM) between the FDIR manager and every component making up the system modules, routers and links. HSM can be both solicited (pulled) or unsolicited (pushed). Each component will have a programmable time period (T*hs*) associated with it. HSM system is the basis by which the FDIR Manager is able to detect the occurrence of fault within MARC.
- *Fault Detection*: Fault detection is determined by both the arrival time of expected HSM (both solicited and unsolicited) and by the content of the messages themselves. Faults are registered for a specific component if an expected HSM from that component is either early or late or does not arrive at all, or if the content of a message that does arrive on time flags an 'I am not OK' status. Using the expected T*hs*, and three further programmable time constants T0, T1 and T2 (where T0<T1<T2), the FDIR Manager can work out a time range within which T*hs* of the expected HSM can be classified as **Valid**, **Missing, Late** or **Lost**.
- *Fault Diagnosis and Identification:* Following detection of a fault, the FDIR Manager will log the event in its Registered Fault Log. The Fault Diagnosis Procedure is then initiated and during this process, statuses are assigned by the FDIR Manager to the fault events in order to record the seriousness of the fault and the action that should be taken in the event of a recurrence. There are three fault event statuses: **Transient, Intermittent** and **Permanent.**
- *Fault Recovery:* fault recovery is based on 3 procedures that are:

a) *Initial Recovery Procedure*: That is triggered by the FDIR Manager in the event that the Diagnosis Procedure has identified and localised a fault within a component. It represents a final attempt to recover functionality in the component before the use of redundant components and rerouting is performed.

a) Initial Recovery Procedure: That is triggered by the FDIR Manager in the event that the Diagnosis Procedure has identified and localised a fault within a component. It represents a final attempt to recover functionality in the component before the use of redundant components and rerouting is performed.

b) *Router Recovery Procedure*: That is initiated by the FDIR Manager following the Initial Recovery Procedure to recover a faulty router. The Procedure uses the stored FDIR-Tabs to reconfigure and restore the network and varies depending on if the network has cross strapping. Once the new router or routers have been activated, new routing address tables are set and the FDIR Manager sends notifications to all affected nodes that the new routers are available and must now be used.

c) *Node Recovery Procedure*: That is initiated by the FDIR Manager following the Initial Recovery Procedure to recover a faulty module. The procedure is linear, and uses the stored FDIR-Tabs to initiate redundant module(s) and set up new routing address tables. Firstly, the faulty module is

switched off and the selected redundant node (identified using the FDIR-Tabs) switched on. The new module is then configured, using the Configuration Manager, and a new routing address table set up. Finally the FDIR Manager sends notifications to all affected modules that the new module is available and must now be used.

4. MARC Demonstrator

In a first stage, to validate MARC conceptual architecture a prototype called MARC Demonstrator will be developed. As presented in figure 4, it is based on 2 clusters and will encompass the following features:

- Backplane (i.e. SpW_10X routers and HRC)
- Core Computing Module (based on AT697
- Mass Memory Module including file system
- General Computing Module: (based on COTS Power-PC I/Fs)
- Emulation of I/O and TMTC module via PCs and specific SpW EGSEs.
- GenFas software including FDIR Manager

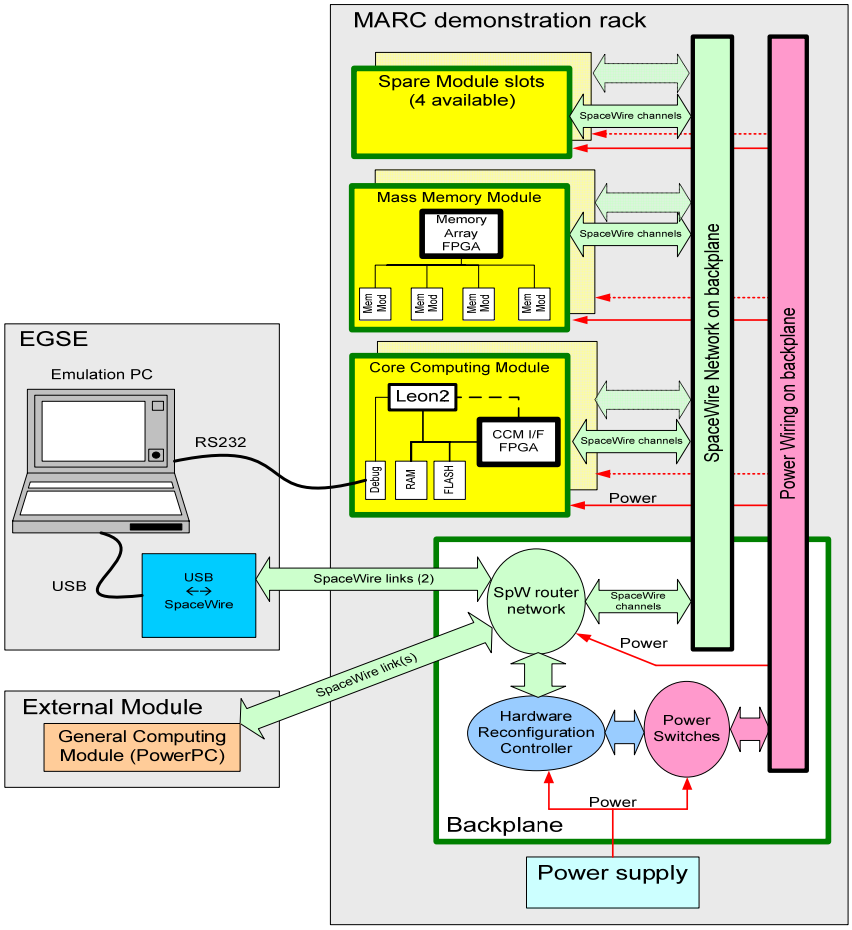


Figure 4: **MARC Demonstrator and Test Bench**

5. Conclusion

As presented in this paper, MARC design is relying on BBs not only ensuring its robustness but also its high flexibility. The MARC development is based on two phases: phase one for preliminary design and feasibility aspect, phase two for detailed design, manufacturing and tests/verification. The first phase has been completed and indicates that:

- The requirements and the FT-Design Principles as defined are clear and feasible,
- MARC preliminary design is consistent with previous requirements.
- This preliminary design of MARC is fulfilling many future ESA mission OBCS needs.
- The time and the cost needed to adapt MARC Demonstrator system to new mission appears to be significantly shorter than a development of a new system.

As a general requirement, it was also required to assess MARC overall performances compared to existing solutions. To this end, MARC Demonstrator will be the first MARC System prototype

allowing performance assessment. MARC Demonstrator will be available at the level of Elegant-Bread Board (as defined by ECSS-E-10-02A).

6. Reference Documents

RF1: ECSS-E-50-12A**,** SpaceWire - Links, nodes, routers and networks, January 2003
RF2: ECSS-E-50-11 Draft version09, Remote Memory Access Protocol (RMAP), June 2008
RF3: CCSDS Spacecraft On-Board Interfaces Services (SOIS) Informational Report, Green Book, 850.0-G-1.1, Jan 2008 and related documents (i.e. SOIS Sub-Network Packet Service, SOIS Sub-Network Memory Access Service, SOIS Sub-Network Time Distribution Service)

A PORTABLE SPACEWIRE/RMAP CLASS LIBRARY FOR SCIENTIFIC DETECTOR READ OUT SYSTEMS

Session: Onboard Equipment and Software

Short Paper

Takayuki Yuasa, Wataru Kokuyama, Kazuo Makishima, Kazuhiro Nakazawa

The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo, Japan 113-0033

Masaharu Nomachi,

Laboratory of Nuclear Studies, Graduate School of Science, Osaka University, 1-1 Machikaneyama, Toyonaka, Osaka 560-0043

Motohide Kokubun, Hirokazu Odaka, Takeshi Takashima, Tadayuki Takahashi

Department of High Energy Astrophysics, Institute of Space and Astronautical Science (ISAS), Japan Aerospace Exploration Agnency (JAXA), 3-1-1 Yoshinodai, Sagamihara, Kanagawa Japan 229-8510

E-mail: yuasa@amalthea.phys.s.u-tokyo.ac.jp, kokuyama@granite.phys.s.u-tokyo.ac.jp, maxima@phys.s.u-tokyo.ac.jp, nakazawa@phys.s.u-tokyo.ac.jp,

nomachi@fn.lns.sci.osaka-u.ac.jp, kokubun@astro.isas.jaxa.jp, odaka@astro.isas.jaxa.jp, ttakeshi@stp.isas.jaxa.jp, takahasi@astro.isas.jaxa.jp

ABSTRACT

We developed a C++ class library which provides modularized scheme to transfer data via RMAP over SpaceWire. The library is designed to be highly portable so that users can execute their products using it on both POSIX (eg. Linux or MacOS X) and TRON environments. TRON is a real-time operating system that is widely adopted in embedded computers, and will also be used on computers, called SpaceCube, to be onboard Japanese scientific satellites. To achieve the portability, we encapsulated hardware- and operating-system-dependent functionalities such as SpaceWire I/F, a multi threading framework, as well as a TCP/IP socket implemented for development phases of scientific payloads.

1 SPACEWIRE/RMAP LIBRARY

1.1 SPACEWIRE-BASED DATA ACQUISITION SYSTEM

We are developing a new data acquisition (DAQ) system based on SpaceWire and RMAP (Remote Memory Access Protocol) to be widely used in future satellite-borne scientific instrument developments [1,2]. Compared to conventional systems, a SpaceWire network-based DAQ system has features of high scalability and compactness. Using SpaceWire I/F from early stages of developments will enable smooth transition from R&D phase to actual fabrication and integration of the flight hardware. To reduce costs of individual developers, a standard framework for SpaceWire-based DAQ has been developed and distributed by Japan SpaceWire Users Group which consists of research institutes (including JAXA and universities), and industrial enterprises.

As shown in Figure 1, present DAQ system consists of the following three components.

- **Detector**: Outputs analog or digital signals. Controlled through specific interfaces.

- **SpaceWire I/F Circuit Board**: Digitizes and stores output signals to make them accessible from SpaceWire network through their SpaceWire interfaces. Equipped with reconfigurable FPGAs to implement user-dependent logics for detector control and signal processing.

- **SpaceCube**: Small-sized computer with SpaceWire interfaces. Operated with a real-time operating system, on which user-dependent read-out programs written in C/C++ runs. "SpaceCube" is a name of the architecture, not a product name.

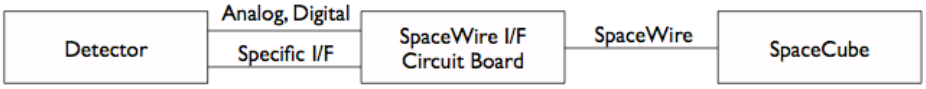


Figure 1. Components used in SpaceWire-based data acquisition

Detectors are designed and developed by individual users. Several types of general-purpose SpaceWire I/F circuit boards have been developed each with different functionality such as analog-to-digital conversion (ADC) and digital input/output. As SpaceCube computer, we usually use SpaceCube1 developed by Shimafuji Electric for ground-based experiments. Currently available space-qualified SpaceCube computers include SpaceCube2 (NEC) and SpaceCard (Mitsubishi Heavy Industrial).

1.2 Concept and Structure of SpaceWire/RMAP Library

Individual instrument developments need their own read-out program to perform different SpaceWire and RMAP accesses. However, basic functions needed therein are essentially the same; examples include send/receive packets and interpret/create RMAP packets. Therefore we developed a common software library scheme to deal with SpaceWire/RMAP-related functions, called SpaceWire/RMAP Library. It is designed to be modularized and portable, to improve re-usability of products and to make validation processes easier. Since ground-based and spacecraft-borne SpaceCube computers could have different operating systems, dependencies on the hardware and operating system are carefully minimized.

The library is written in C++ language, and has no dependency on external library except for STL (Standard Template Library; widely available in many operating systems and compilers). Figure 2 shows the whole structure of the library. SpaceWire I/F class is an abstract wrapper class for SpaceWire I/F hardware and drivers loaded in operating system kernels. It holds virtual methods for device initialization/finalization (open()/close()), packet transfer (send()/receive()), and device configuration (setLinkStatus()). Actual SpaceWire interfaces can be utilized by filling each virtual method with codes specific to those devices. Currently, two types of implementations of SpaceWireIF class are available,

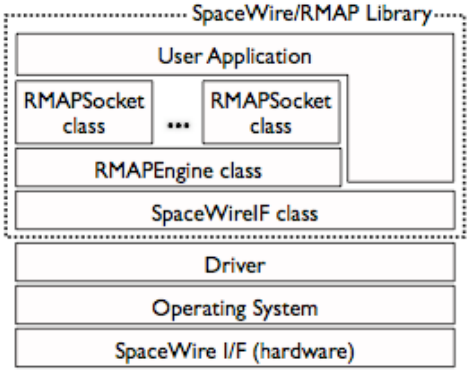


Figure 2. Structure of SpaceWire/ RMAP Library.

each for SpaceWire interfaces (or IP cores) developed by Shimafuji Electric and by NEC Software, respectively.

RMAP transaction is realized by two conjunct classes, RMAPSocket and RMAPEngine. RMAPSocket, which corresponds to socket in a TCP/IP protocol stack, provides read() and write() methods for user application to perform RMAP transactions to a designated RMAP destination node. RMAPEngine handles requests from multiple RMAPSockets and realizes concurrent multiple RMAP transactions (to different RMAP nodes) in multi-threading environments. RMAP packets, errors, and destination information including routing pathways, are also expressed as classes.

A DAQ read-out program often uses multi threading and TCP/IP data transfer. Although those functions have less relevance to SpaceWire or RMAP, we also included them into the library as encapsulating classes, called Thread Library and IP Socket Library, referring to Java Thread and Socket in their naming style. Table 1 lists classes included in these libraries. At present, these libraries provide implementation classes for POSIX and T-Kernel[1] environments, and therefore, ensure a high portability of the programs developed using these libraries. In section 2.2, we present an example of the portability by executing a multi-thread read-out program on both Macintosh (POSIX) and SpaceCube1 (T-Kernel) without changing the source code.

Table 1. Contents of Thread and IP Socket Libraries.

- Thread Library
  - Thread class
  - Condition class
  - Mutex class
  - Message class
- IP Socket Library
  - IPSocket class
  - IPClientSocket class
  - IPServerSocket class

1.3 Current Status of the Library

The total size of SpaceWire/RMAP Library and sub-libraries is about 300,000-line C++ code, including in-source documentation for automatic document generators such as Doxygen or Javadoc. The source code archive is distributed through Japan SpaceWire Users Group, toghether with the Shimafuji Electric's driver software for SpaceWire I/F IP core on SpaceCube1[2]. GNU Compiler Collection's C++ compiler (GCC g++) can be used to build the library and read-out programs. Since an HTML-based reference of the application programming interface (in English) and an introductory tutorial (in Japanese) are also available, users can start developments immediately. The library has been used in several detector development activities, including the X-ray CCD and the X-ray micro-calorimeter experiments onboard the ASTRO-H satellite, the next generation Japanese X-ray astrophysical observatoryto be launched around 2013.

2 Performance and an Example of Application

2.1 Transfer Speed

We measured the data transfer speed using SpaceCube1, implemented with Shimafuji Electric SpaceWire IP core and the SpaceWire/RMAP Library. In hardware (physical) layer, the IP core exploits about 80% of link speed; for example, 80 Mbps can be achieved over a 100 MHz link. Overhead time of the host I/F (PCI I/F) and the

[1] T-Kernel is one kind of implementation of TRON operating systems.
[2] Which will be released as an open source product in the near future at http://www.shimafuji.co.jp/ .

driver software reduce the speed to about 32 Mbps (in SpaceWireIF class layer). We also tested practical end-to-end data transfer speed between a RAM on SpaceWire I/F circuit board and a read-out program on SpaceCube1. The SpaceWire I/F circuit board acts as an RMAP target device, and the read-out program utilizes full features of SpaceWire/RMAP Library, such as the concurrent multiple transaction with RMAPSocket and RMAPEngine. When we transferred data with a packet size of 8 kB, the transfer speed of 8 Mbps was obtained.

2.2 Example Application

We applied the SpaceWire/RMAP Library to a simple ground experiment setup. The detector consists of two photo-multiplier tubes (PMTs) with inorganic scintillators and amplifiers attached to them. The SpaceWire I/F circuit board in this case is a octal 50 MHz ADC board, which is controlled and read-out via its SpaceWire I/F. Using SpaceWire/RMAP Library, we developed a read-out program initially for SpaceCube1. To examine the portability, we then built the same source code with a C++ compiler for MacOS X (g++), and executed the generated binary on Macintosh. Since a SpaceWire I/F for Macintosh was not available at that time, the program used the SpaceCube's SpaceWire I/F via TCP/IP. To realize a SpaceWire-to-TCP/IP converter, we also executed a converter server daemon program on the SpaceCube[3].

The read-out program executed on Macintosh communicates with the SpaceCube via TCP/IP to send/receive SpaceWire packets to/from SpaceWire ADC Box (Figure 3). Such procedures are automatically performed inside the library, and hence, the read-out program need not to be modified. The total DAQ system was also successfully operated in the same way as on SpaceCube1, and correctly transferred 700,000 events with an event size of 10 bytes/event each in 5 minutes in a test measurement wherein we irradiated the detector with gamma rays from radioisotopes.

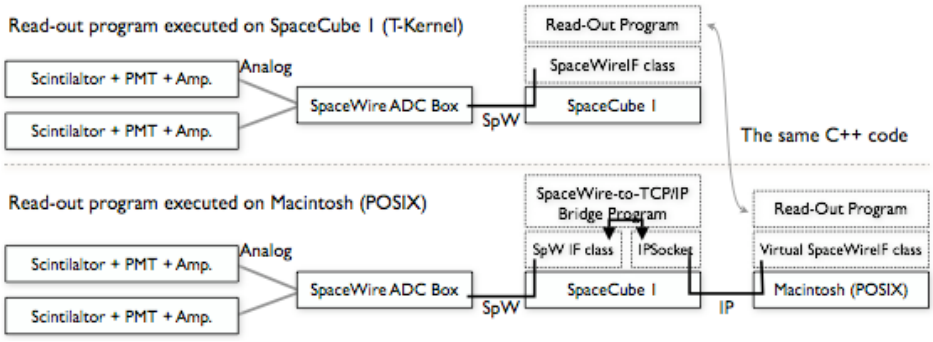


Figure 3. A block diagram of test measurements.

3 References

1. Takayuki Yuasa et al., “Development of a SpW/RMAP-based Data Acquisition Framework for Scientific Detector Applications”, International SpaceWire Conference, Dundee, Scotland, September 2007
2. Hirokazu Odaka et al., “Development of a SpaceWire-based Data Acquisition System for a Semiconductor Compton Camera”, International SpaceWire Conference, Dundee, Scotland, September 2007

[3] The SpaceWire-to-TCP/IP converter program for SpaceCube1 is also included in the library package.

CRUCIAL SPACEWIRE ELEMENTS IN RASTA

Session: SpaceWire Equipment and Software

Short Paper

Sandi Habinc, Jiri Gaisler

Gaisler Research, Första Långgatan 19, SE-413 27 Göteborg, Sweden

Claudio Monteleone, Chris Taylor

European Space Agency, Keplerlaan 1, P.O. Box 299, NL-2200 AG Noordwijk, The Netherlands

E-mail: sandi@gaisler.com, jiri@gaisler.com, claudio.monteleone@esa.int, chris.taylor@esa.int

ABSTRACT

The RASTA (Reference Avionics System Test-bench Activity) was initiated by the European Space Agency (ESA) to provide a single development platform to reduce the number of prototyping environments established in technology developments.

The RASTA objectives are to allow the developed technology items to be validated and demonstrated in a flight representative prototype environment, to support mission and spacecraft design and on-board software validation through the project life-cycle by means of a coherent development platform, to maximize reuse of the existing avionics technologies and to be scalable and flexible.

The main communication interface used in RASTA is SpaceWire network [1]. Some of the crucial elements of the RASTA initiative are based on the GRSPW SpaceWire Link Codec IP core [5] developed by Gaisler Research, Sweden. The IP core has been included in many of the FPGA and ASIC implementations that are used in the RASTA boards.

1 OVERVIEW

The RASTA development platform has been developed for ESA to provide the means to demonstrate functional and performance requirements of satellite on-board systems and subsystems. A crucial part of the RASTA development platform is the SpaceWire network and the elements implementing it. The key elements implementing the SpaceWire network are the various RASTA boards harbouring ASIC or FPGA components featuring SpaceWire links.

A variety of boards have been developed to support the design of complex avionics and payload management systems. These boards carry components with SpaceWire interfaces from a number of different manufactures. For example, boards have been developed to support the Aeroflex UT699 device, the Atmel AT7913E (SpaceWire-RTC) ASSP and the LEON3FT-RTAX Actel FPGA based processors.

Other boards are also compatible with the RASTA concept, such as the SpaceWire router boards from Aeroflex and Star-Dundee. Generic FPGA boards targeting Xilinx and Actel devices are used to implement system-on-a-chip configuration.

To be representative, the RASTA development platform is based on flight compatible hardware and software. This is supplemented by the addition of a workstation hosting the cross-development and debugging environment for software development. The two together provide an "easy to use" facility that allows developers to perform tests and validation, at system level, in a highly representative environment.

2 PROCESSOR BOARDS

The main building blocks of the RASTA development platform are the processor boards. The processor boards contain basic memory and debug interfaces, such as the LEON2/3 Debug Link UART and JTAG. The boards also contain an Ethernet 10/100 Mbit/s MAC and PHY, which can be used with RTEMS or VxWorks Workbench.

Processing can also be provided by means of FPGA devices, with the LEON3 SPARC processor [6] pre-programmed into either an Actel or a Xilinx FPGA. The latter allows for prototyping of processor systems with novel features such as multiple processor cores or a memory management unit.

All processor boards can also be extended with accessory boards providing common interfaces such as RS-232 UART interfaces.

3 INTERFACE BOARDS

For interfaces other than a simple UART, the processor boards can communicate via the cPCI backplane to interface boards implementing dedicated communication protocols. Each interface board implements a PCI Initiator and Target interfaces to allow efficient communication with the processor board.

The initiator capability allows the interface board to implement direct memory access (DMA), off-loading the processor from repetitive task such as periodic data movement. This concept has been proven with the Advanced Data and Power Management System (ADPMS) on the PROBA-2 spacecraft.

The PCI interface has the following capabilities:
- 32-bit bus width, 33 MHz (133 MByte/s)
- 32-bit DMA bus master (133 MByte/s)

In the future, the cPCI back-plane interface will be replaced with an active SpaceWire backplane using SpaceWire routers. For now, the SpaceWire communication is done via front-panel connectors.

The interface boards implement the various communication protocols. The following communication protocols have been implemented so far:
- SpaceWire with Remote Access Memory Protocol (RMAP)
- Redundant Mil-Std-1553B bus
- Controller Area Network (CAN) 2.0B
- 10/100 Mbps Ethernet

- ECSS/CCSDS Telemetry, with PacketWire etc.
- ECSS/CCSDS Telecommand, with PacketWire etc.
- CCSDS Time Management with TimeWire
- General Purpose Input Output (GPIO)

The key SpaceWire interface has the following capabilities:
- 200 Mbps maximum rate, full duplex, DMA capability
- Remote Access Memory Protocol (RMAP) [2]
- MDM connectors with LVDS electrical levels

The CCSDS telemetry and telecommand interface has the following capabilities:
- Two different telecommand decoders
- Two different telemetry encoders, with support up to 8 Virtual Channels
- Dedicated PacketWire and SpaceWire interfaces

Each interface board also implements a memory controller interfacing the following memory. The memory can be used as safeguard memory required for the processor board to communicate with the remaining representative flight data systems in the RASTA architecture.

One or more of the above interfaces can be implemented on the same interface board. The internal RASTA interface board FPGA design is based on the de-facto standard AMBA Advanced High-speed Bus (AHB) [3], to which the high-bandwidth units are connected. Low-bandwidth units are connected to the AMBA Advanced Peripheral Bus (APB), which is accessed through an AHB to APB bridge.

4 FUTURE BOARDS

The RASTA development platform has been developed to take into account future development boards, covering upcoming processor devices and for example SpaceWire router ASICs.

To mention a few examples, the UT699 (Aeroflex) 6U cPCI board features the LEON3-FT processor, the UT200SpW16RTR-EVB (Aeroflex) 6U cPCI evaluation board features a 16-port SpaceWire router, and the SpaceWire-RTC (AT7913E Atmel) ASIC development suite features a 6U cPCI ASIC board.

Other interfaces can also be developed, such as I2C, SPI, USB and Gigabit Ethernet, by using IP cores from the Gaisler Research VHDL IP core library GRLIB [4].

5 INTEGRATION

To allow for simple hardware integration, all boards used in the RASTA development platform are compliant with the cPCI standard. The boards are pre-assembled during system integration and verification using commercial cPCI crates, either 3U or 6U depending on what boards are required. Each RASTA system is shipped ready to use in a cPCI crate together with cabling, documentation and demonstration software.

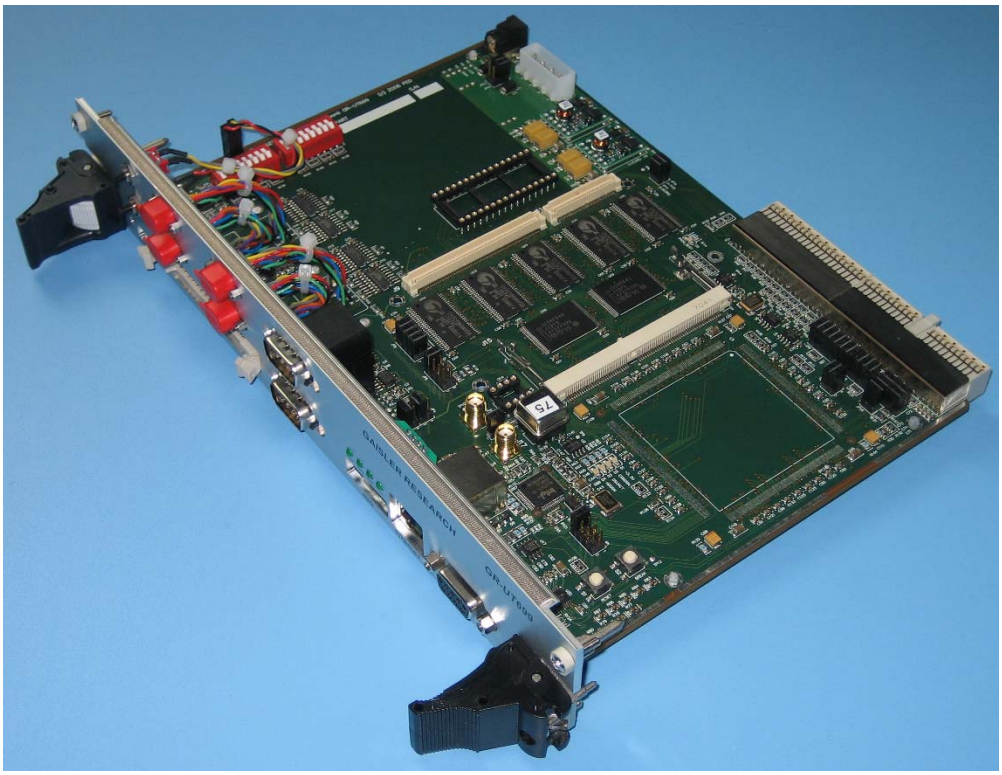


Figure 1 – Example of a 6U high UT699 LEON3FT processor board that fits in a RASTA system

6 CONCLUSION

The critical SpaceWire elements in the RASTA initiative have been largely based on the GRSPW SpaceWire Link Codec IP core. The core has the capability to directly interface to the AMBA AHB and APB on-chip buses by means of DMA capability. The core also implements the complete RMAP communication protocol directly in hardware. The use of RMAP in combination with on-chip or off-chip processors provides the possibility for remote software debugging over SpaceWire.

7 REFERENCES

1. "SpaceWire standard - Links, nodes, routers and networks", ECSS-E-ST-50-12C, 31 July 2008

2. "SpaceWire protocols", ECSS-E-ST-50-11C, July 2008

3. "AMBA Specification, Rev 2.0, ARM IHI 0011A, 13 May 1999, Issue A, first release, ARM Limited

4. "GRLIB IP Library User's Manual", Gaisler Research, www.aeroflex.com/gaisler

5. "GRLIB IP Core User's Manual", Gaisler Research, www.aeroflex.com/gaisler

6. "The SPARC Architecture Manual", Version 8, Revision SAV080SI9308, SPARC International Inc.

A CPU MODULE FOR A SPACECRAFT CONTROLLER WITH HIGH THROUGHPUT SPACEWIRE INTERFACES

Session: Onboard Equipment & Software

Short Paper

Toru Sasaki, Minoru Nakamura, Tadashi Yoshimoto, Minoru Yoshida,

and Shoji Yoshikawa

Advanced Technology R&D Center, Mitsubishi Electric Corporation, 8-1-1, Tsukaguchi-Honmachi, Amagasaki, Hyogo, 661-8661, Japan

E-mail: Sasaki.Toru@eb.MitsubishiElectric.co.jp,

Nakamura.Minoru@ea.MitsubishiElectric.co.jp,

Yoshimoto.Tadashi@ap.MitsubishiElectric.co.jp,

Yoshida.Minoru@ea.MitsubishiElectric.co.jp,

Yoshikawa.Shoji@ap.MitsubishiElectric.co.jp

ABSTRACT

We have developed a CPU module for a spacecraft controller with high throughput SpaceWire interfaces. This CPU module is a main module of the spacecraft controller which executes AOC(Attitude and Orbit Control) and DH(Data Handling) processing. The amount of data from payload subsystems has been increasing in recent spacecrafts, such as earth observing spacecrafts. With that in mind, we selected SpaceWire for its high data transfer rate. To accelerate data transfer throughput without requiring excessive CPU resources, we have developed a SpaceWire controller and a DMA controller. We have made a functional model of the CPU module and evaluated its performance for data transfer. This paper discusses the architecture of the CPU module and the results of its evaluation.

1 INTRODUCTION

To develop a spacecraft controller in a short time and at low cost, we have recognized the importance of establishing a design standard. SpaceWire is becoming a standard for networks of onboard satellites all over the world. In Japan, too, a spacecraft controller equipped with SpaceWire has already been developed [1]. The flexibility and high-throughput of SpaceWire makes it very attractive for the network in our spacecraft controller. The first step in our program for adopting SpaceWire was developing the CPU module for the spacecraft controller, which is designed for the small and medium size satellites. This CPU module uses DMA to accelerate the data transfer through the SpaceWire. Here, we introduce this spacecraft controller and its CPU module, in particular.

2 SPACECRAFT CONTROLLER

Fig. 1 shows a block diagram and picture of the spacecraft controller. The CPU module's main functions are AOC and DH. Sensors, such as thermistors, star sensors and Fiber Optic Gyros (FOGs), are connected to the Sensor I/F module, which uses an FPGA and ADC to gather digital and analog data from the sensors. The I/O module interfaces with actuators such as reaction wheels. The Data Recorder module has an FPGA and SDRAMs for storing data from pay-load subsystems and housekeeping data. A soft-core CPU is embedded in each FPGA for controlling of synchronization and interfaces between modules.

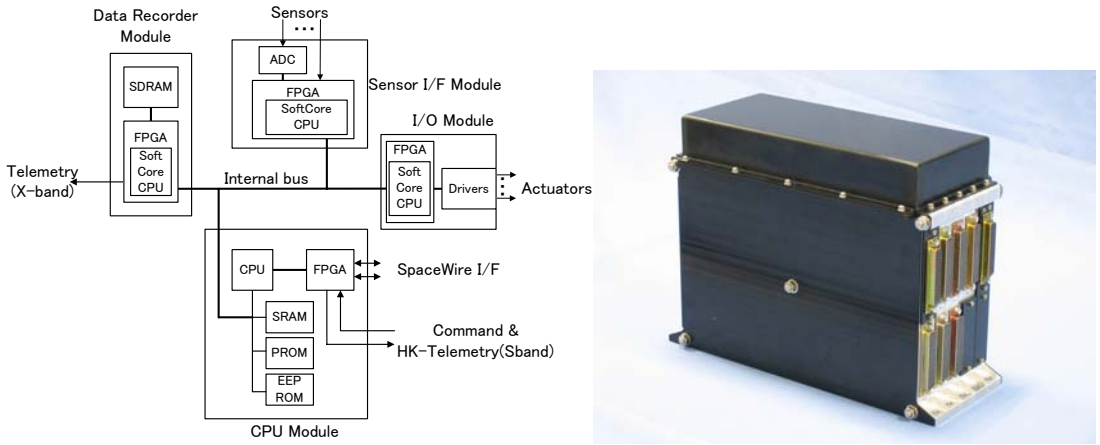


FIG.1. The block diagram and the picture of the spacecraft controller
The size is 310(W)×116(D)×236(H)mm and the weight is 5 kg

3 CPU MODULE

Table 1 gives the specification of the CPU module and Fig. 2 shows the picture of the CPU module. The CPU module is composed of HR5000 CPU, which is developed by JAXA, memories and an Anti-Fuse FPGA. The CPU and the FPGA are connected with PCI bus. Fig. 3 shows a function block diagram of the FPGA. The SpaceWire Controller manages 2-channel SpaceWire interfaces which are supposed to be connected to mission payloads. The DMA Controller has 4 channels for sending and 4 channels for receiving to handle the data transfer of SpaceWire and other peripherals. The FPGA has also a PCI Bus Bridge Controller, a WatchDogTimer, a Timer, a GPIO an Interrupt Controller and a Serial Interface Controller. The PCI Bus Bridge Controller controls the PCI bus between HR5000 and the FPGA.

We describe the specification of the DMA Controller, which has two special functions to enhance the throughput of the SpaceWire. Firstly, this DMA Controller has two sets of control registers, count registers, and destination or source address registers in each channel. This design enables software to configure one set of registers while the DMA controller is transferring data under the other configured set of registers(double register mode). Therefore we can reduce the delay for DMA restart. Secondly, the DMA Controller detects EOP(End of Packet) of SpaceWire packet. The burst-length can be set 1, 2, 4 or 8 in word (4 bytes), but the length of the SpaceWire packet may not be a multiple of the burst-length. This situation might cause the DMA Controller to wait for all the data being filled in the burst length. To

prevent the DMA Controller from waiting, the DMA Controller can detect EOP and can send the residues separately.

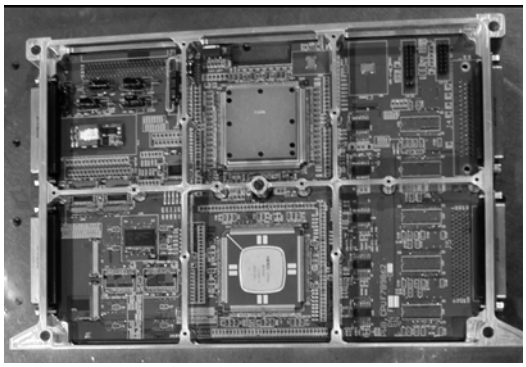


FIG.2. The picture of the CPU module. The two largest ICs in the CPU module are an Anti-Fuse FPGA(upper) and HR5000 CPU(lower).

Table 1. Specifications of the CPU module

CPU	HR5000 (core 100MHz bus 50MHz)
FPGA	RTAX2000(Actel)
SpaceWire Interface	2ch
System Memory	Asynchronous SRAM(2MB) with ECC
Power Consumption	4.2W

To transfer data by using the DMA Controller, software sets a DMA count register and a source or destination address register and selects the burst-length. Then the DMA Controller begins DMA transfer once the software sets a start-bit of a DMA control register. If we use the double register mode, we can set another registers and keep the channel ready for the next DMA transfer while executing the former DMA transfer. DMA transfer is executed until the detection of EOP or the expiration of a DMA count register. Transfer completion is notified by an interrupt or a flag in the DMA status register.

We also developed software which works on the CPU module and manages the transfer of the data by using SpaceWire Controller and DMA Controller. The software uses an original operating system(OS). The device drivers in the OS handle the controllers in the FPGA.

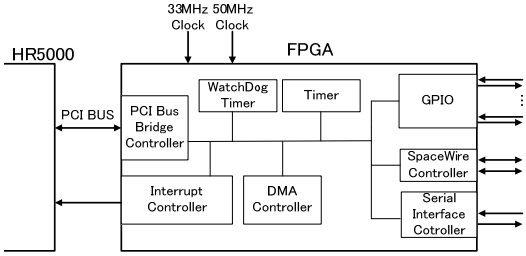


FIG.3. FPGA function block diagram

4 EVALUATION

Our evaluation of the CPU module was especially focused on the effect of the double register mode. We compared the data transfer rate using the double register mode with the rate using a single register(single register mode) and examined the effects of varying the length of SpaceWire packets.

For this evaluation, a SpaceWire interface was connected to a loopback wiring. And we made the CPU module to send data from the system memory and to receive the same data through the loopback. We measured the time for the completion of transferring data. The link speed of SpaceWire was set 10Mbps. The result is shown

in Fig. 4. We found that the transfer rate by the double register mode was enhanced up to about 1.5 times faster. At the length of 384 bytes, the data transfer by the double register mode was 7.01Mbps. The logical data transfer rate of the SpaceWire with 10Mbps link speed is estimated as about 8Mbps. Hence our controller is said to reach about 88% of the maximum performance of SpaceWire with 10Mbps link speed.

5 DISCUSSION

We can see a drop around the short-length packet in Fig. 4. It indicates that the double register mode does not increase the efficiency so much when transferring short-length packets. We suppose that the problem is in software's overhead in re-setting the DMA registers. If the overhead is quite large, the former DMA transfer will complete before software starts up the next DMA transfer. This situation reduces the performance of the double register mode because the double register mode needs a little more complicated process in the completion and start of a DMA than the single. We expect that we can improve the overhead by optimizing the software and get better efficiency of the double register mode around shorter length packets.

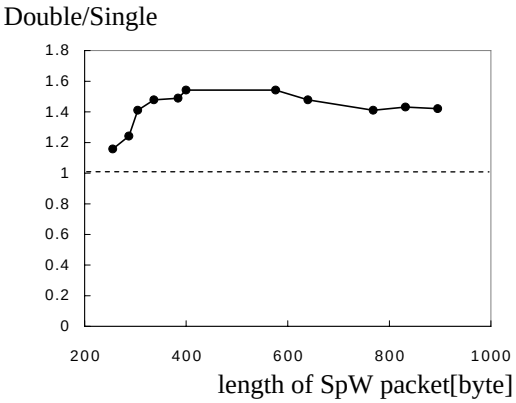


FIG. 4. The ratio of the double to single register mode in transfer rate as a function of length of SpW packet.

6 CONCLUSION

We have developed a CPU module with SpaceWire interfaces for a spacecraft controller. The data through the SpaceWire is managed by a SpaceWire Controller and a DMA controller in an FPGA. The DMA controller has two sets of registers for configuration and detects the arrival of EOP to promote the throughput of the data transfer. The measurement of data transfer showed that using the double register mode enhanced the throughput by 50% at its best, compared to the single register mode. And we found that we need the improvement of the software which handling the double register mode to increase the efficiency of transferring short-length packets. Moreover we will implement the RMAP protocol or the SpW-RT(Spacewire Real Time) protocol in the CPU module. We plan to report the status of these work on the next Conference.

7 REFERENCES

1. Tadayuki Takahashi, Takeshi Takashima, Seisuke Fukuda, Satoshi Kuboyama, Masaharu Nomachi, Yasumasa Kasaba, Takayuki Tohma, Hiroki Hihara, Shuichi Moriyama, Toru Tamura, "Space Cube 2 – An Onboard Computer Based on SpaceCube Architecture", International SpaceWire Conference 2007, Dandee, UK Sep. 17th-19th, 2007.

LEVERAGING SERIAL DIGITAL INTERFACES STANDARDIZATION: THE RASTA REFERENCE ARCHITECTURE FACILITY AT ESA

Session: Spacewire onboard equipment and software

Short Paper

Aitor Viana Sanchez, Gianluca Furano, Massimiliano Ciccone, Farid Guettache, Claudio Monteleone, Chris Taylor

ESA - European Space Technology Centre (ESTEC)

ESA/ESTEC P.O. Box 299 / 2200AG Noordwijk ZH, the Netherlands

Manuel Prieto, Ignacio Garcia Tejedor

University of Alcala

Ctra. Madrid-Barcelona, km. 33.6, 28871 Alcala de Henares, Madrid

E-mail: aitor.viana.sanchez@esa.int, gianluca.furano@esa.int, massimiliano.ciccone@esa.int, farid.guettache@esa.int, claudio.monteleone@esa.int, chris.taylor@esa.int, manuel.prieto@aut.uah.es, ngarcia@aut.uah.es

ABSTRACT

This paper presents an overview of the internal R&D activity project in ESA called Reference Avionics System Test-bed Activity (RASTA). This activity aims to provide a hardware/software reference infrastructure where all the incoming TRP and GSTP activities can be integrated, coming out with a common testbed which may be missionized instead of dedicated environments for each activity.

Within RASTA, most of the space representative buses are included, CAN bus, MIL-STD-1553 and Spacewire. Several units comprise RASTA, such as: LEON2 based on-board computer, Telemetry and Telecommand control unit and a Mass Memory Unit, and the communication technology available so far may be selected between cPCI and Spacewire bus.

One of the latest activities started early this year is the Modular Advanced Mass Memory Architecture (MAMMA). This unit tends to be an *intelligent* unit fully based on Spacewire embedding several services being as autonomous as possible from the on board computer. This architecture will exploit the capabilities of active Spacewire backplanes.

Some RASTA building blocks for the space community are already available: Basic SW and SW drivers (CAN/1553/SpW, TT&C), OS abstraction layer, SW libraries (CFDP). In the future more SW libraries will be available (e.g. PUS library) and more HW units integrated in the environment (e.g. intelligent RTUs)

1 INTRODUCTION AND SCOPE

RASTA is a Data Systems Division's (TEC-ED) infrastructure for hosting, testing and validating hardware and software elements related to flight avionics. One of the main drivers for RASTA has been the need for an end-to-end reference system which can be used to integrate developments resulting from TRP and GSTP activities. RASTA objectives may be summarised as follows:

- To define an overall system that embodies the latest industrial agreements on avionics architectures and includes the necessary elements to cover the end-to-end communications associated with a typical spacecraft
- To provide a system with well defined software and hardware interfaces such that the modification/incorporation of new elements does not disturb the existing infrastructure
- To provide a specification of the reference system so that it may be applied to new developments and used for subsequent integration, test and evaluation
- To publicise the specifications for encouraging harmonisation within industrial infrastructures
- To utilise the system for project support in terms of mission design feasibility and performance
- To utilise the system for standards support including prototyping and validation of new protocols
- To evaluate the feasibility of new flight architectures, in terms of extended connectivity, functional redundancy, use of modern interfaces, use of rapid prototyping methods
- To enable connection with remote elements for example with ESOC ground related test-beds

To achieve its objectives, RASTA must therefore be completed with elements and functions that are in addition to the purely hardware aspects associated with flight avionics such as: an onboard basic software framework and ground developments (e.g. SCOS 2000).

The end-to-end RASTA configuration aims to be representative of a scenario where a Mission Control Centre (MCC) communicates to one or more spacecrafts through a TM/TC space link. Initially, in a configuration where more than one flight segment is present, only one flight segment is directly connected to the Ground System through a TM/TC link. All the additional Flight Segments will be interconnected trough a simulated inter-satellite link (i.e. Proximity-1) but not 'directly' reachable from the Ground Segment. In the near future it is foreseen to have a TM/TC link also to a second Flight Segment.

Figure 1 represents a view of the main HW elements of RASTA and the way they interconnect.

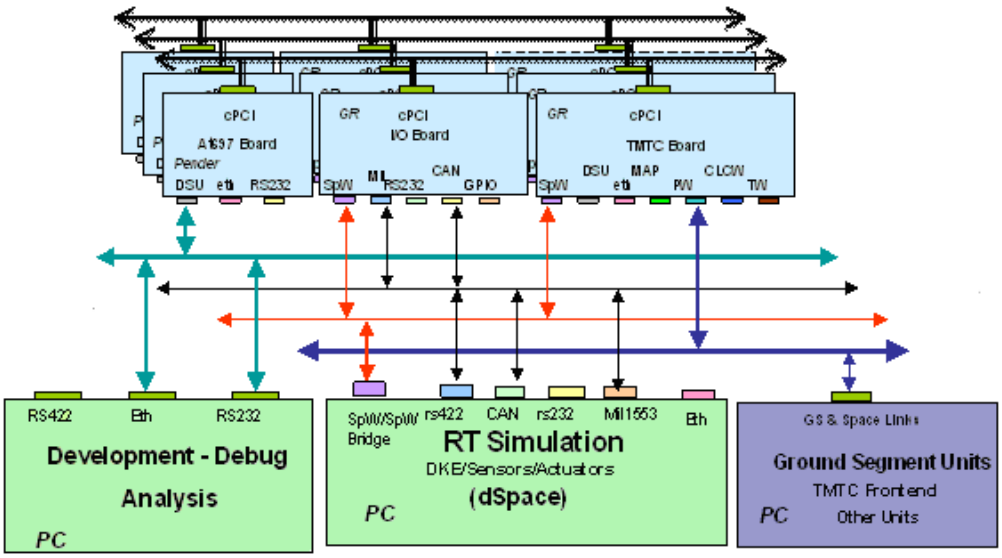


Figure 1 - RASTA Hardware Overview

Several configurations may be tackled with RASTA, so far two configurations are considered. In the baseline configuration a ground segment is connected, through a simulated space link, to a single spacecraft made of two onboard computers (i.e. a CDMU and a Payload computer). While in the extended configuration the same ground segment is connected to two spacecrafts; respectively with two and one onboard computer (i.e. OBC 1 and OBC 2 as part of an orbiter. OBC 3 as part of a Rover). This two configurations will be used to cover various scenarios; from a simple one with a single spacecraft hosting a main OBC and a payload computer, to more complex one with 2 spacecrafts representing the combination of an orbiter and a lander or a lander and a rover.

In section 2.1 the hardware elements comprising RASTA will be detailed. In section 2.2, all the software building blocks available so far and the on-going ones will be briefly explained. The section 2.3, the parallel activity based on RASTA hardware dubbed MAMMA is explained. Finally, section 3 states the conclusions.

2 REFERENCE AVIONICS SYSTEM TED-BED ACTIVITY

2.1 RASTA HARDWARE

The Rasta hardware architecture is designed as an open system which provides physical framework to integrate components and subsystems, from the three domains described below, as they become available:

The On board data System

The on board data management product family is the core element of the facility. It is based on a generic repository of representative building blocks. It is composed of:

- Processor board, mainly LEON2/LEON3 Boards @ 100Mhz
- Interfaces to On board Communication system (Spacewire, CAN, Mil-Bus, RS422, RS232, Ethernet)

- Interfaces to TMTC (Packet-wire, MAP, CLTU)

These elements have been designed to be representative of flight hardware and to enable cross-coupling in FT architecture

The data system environment

The on board data systems environment consists mainly on AOCS/GNC related components and sub-systems (vehicle dynamics, kinetics, environment, sensors, actuators) and payload subsystem. These blocks can be either simulated on workstations or real units.

The ground equipment

Ground segment units representative of the real Ground Segment facility (TMTC Front End, etc...)

2.2 RASTA SOFTWARE

2.2.1 RTEMS REAL-TIME OPERATING SYSTEM

Real-Time Executive for Multiprocessor Systems is a free open source real-time operating system designed specifically for embedded systems. It is wide use for embedded applications and it is becoming a de-facto standard for space applications, mainly because the source code is fully available.

RTEMS has been designed to support different API standards (e.g. POSIX), also providing a native API and up to it RTEMS will be qualified.

RTEMS has been selected for being the RASTA RTOS mainly because it is open source, even though all the software assets for RASTA are mainly RTOS independent to ensure portability.

2.2.2 OPERATING SYSTEM ABSTRACTION LAYER (OSAL)

The goal of this library is to promote the creation of portable and reusable real time embedded software for RASTA. Given the necessary operating system abstraction layer implementations, the same embedded software might compile and run on different platforms and on a different operating system ranging from spacecraft computer systems to desktop PCs, ensuring that all the software deliverables for RASTA will work even when changing the platform, RTOS, etc.

The OSAL library is first coming from a NASA open source project ([1]), and has been adopted in RASTA to be one of the software cores of the system. TEC-EDD section at ESA is maintaining its own OSAL flavour and feed backing to NASA's project when a major change is performed.

2.2.3 CCSDS SPACECRAFT ON-BOARD INSTERFACE SERVICES (SOIS)

For the CCSDS SOIS ([4]), two independent activities are underway by SciSys (UK) and SAAB (S). Both activities take benefit from the sub network protocol prototyping activities for Milbus and Spacewire. The implementation from SciSys will use the OSAL abstraction library to ensure portability across different platforms and Operating Systems.

2.2.4 CCSDS FILE DELIVERY PROTOCOL (CFDP) LIBRARY

Within the RASTA activity, a CFDP ([2]) library together with a CFDP validation environment is being developed under a GSTP contract with SpaceBel. So far, the first deliverable comprising the validation environment and a CFDP library (non-flight version) is already available. A new flight version developed from scratch is on-going to be tested in RASTA.

2.3 RASTA RELATED ACTIVITIES

2.3.1 MODULAR ADVANCED MASS MEMORY ARCHITECTURE

This activity is framed into RASTA and will implement new development concepts for future Solid State Mass Memories (SSMM). MAMMA will be mainly based on space wire backplane connections between memory and MMU controller and, aims to be fully independent from the memory device technology while also focused on FLASH memories maximizing the performance over FLASH. To achieve that the memory access natively block-based, and any other access (e.g. byte based) is 'emulated' on top of it.

The activity is today half implemented using external SDRAM ([3]) memory connected to the controller using cPCI backplane. So far the maximum bandwidth achieved (write burst transfers, DMA, no EDAC) in this configuration is 70 Mbytes/s, which is the upper boundary. This data rate will for sure be lower when integrating the MMU communication protocol and using Spacewire as backplane. Some measurements have been already performed over the space wire interfaces in RASTA, to know exactly what bandwidth might be expected. Best effort tests over v (configured at 200 Mbps) in RASTA shows that only ~6 Mbytes/s data rate can be achieved (see Figure 2), and it is dependent from the packet size. Those numbers are reasonable having in mind the fact that Spacewire is implemented in RASTA using a 33 MHz FPGA.

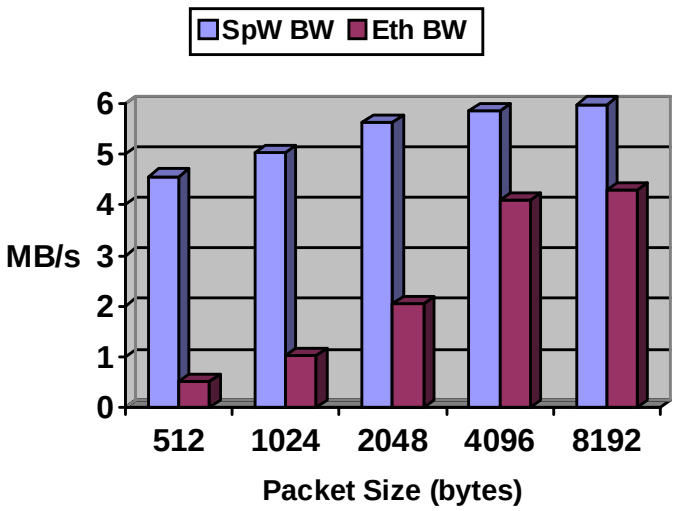


Figure 2 - RASTA SpW/Ethernet Bandwidth (Best Effort)

In the first half of the activity, the low level protocol to access the memory devices (local/remote access) and the error detection and correction mechanisms (RAID) have

been implemented and tested. From now until the end, the file system and the user communication protocol will be implemented and tested.

3 CONCLUSIONS

The paper shows how a « building block » approach in avionic systems prototyping and validation may lead to the development of innovative products and concepts that are of general space industry benefit.

The versatility of serial digital interfaces and the availability, for space systems of performance grades that until now were only possible with Commercial Of The Self based electronics allows also exploring new architectural concepts that increase failure tolerance and overall reliability of systems well beyond the simple cross strapped redundancy schemes used so far. The MAMMA Spacewire backplane powered mass memory here presented could be a good case of study.

4 REFERENCES

1. GSFC Open Source SW - http://opensource.gsfc.nasa.gov/projects/osal/osal.php

2. CSDS File Delivery Protocol (CFDP): Introduction and Overview. Green Book. Issue 3. April 2007

3. MM-6165D 2GB or 4GB 64-bit/66MHz PMC SDRAM Memory Buffer - http://www.vmetro.com/category4194.html

4. Spacecraft Onboard Interface Services. Green Book. Issue 1. June 2007 (http://public.ccsds.org/publications/SOIS.aspx)

Test & Verification 1

Thursday 6 November
11:00 – 12:20

THE SPACEWIRE INTERNET TUNNEL AND THE ADVANTAGES IT PROVIDES FOR SPACECRAFT INTEGRATION

Session: SpaceWire Test and Verification

Long Paper

Stuart Mills, Steve Parkes

University of Dundee, School of Computing, Dundee, DD1 4HN, Scotland, UK

Raffaele Vitulli

European Space Agency, ESTEC, Keplerlaan 1, 2201 AS Noordwijk, The Netherlands

E-mail: smills@computing.dundee.ac.uk, sparkes@computing.dundee.ac.uk, Raffaele.Vitulli@esa.int.

ABSTRACT

The SpaceWire Internet Tunnel is a tool developed by the University of Dundee and STAR-Dundee to allow the components of a spacecraft utilising SpaceWire to be integrated virtually. This concept is known as "virtual spacecraft integration" and involves connecting the components remotely using a network such as the Internet. A pilot study was conducted by ESA to investigate the benefits of such a system.

This paper describes the SpaceWire Internet Tunnel in detail, reporting some of the technical hurdles which were overcome to achieve virtual spacecraft integration. Advantages and limitations of the system, identified by both the University of Dundee and those involved in the pilot study are described, with the reasons for any limitations explained.

1 INTRODUCTION

The concept of virtual spacecraft integration has been described in previous papers by the University of Dundee [1] [2] [3]. It provides a means by which integration testing of spacecraft components can be performed without the need to bring each of the components to one physical location. The SpaceWire standard [4] aims to improve reusability, promote compatibility and reduce system integration costs. Virtual spacecraft integration has the potential to reduce system integration costs still further, by reducing travel and by identifying problems at an earlier stage of spacecraft development than is currently the case.

Virtual integration is achieved through the use of a network such as the Internet. A section of the spacecraft's onboard bus is replaced with a virtual connection over the network, allowing components to communicate with one another despite potentially being great distances apart.

2 THE SPACEWIRE INTERNET TUNNEL

The SpaceWire Internet Tunnel is a tool for performing virtual spacecraft integration in a SpaceWire network. Originally developed by the University of Dundee under ESA contract, it is now a commercial product available from STAR-Dundee [5]. An example SpaceWire network which could benefit from virtual spacecraft integration is shown in Figure 1. This network contains two separate sub-systems, which may be developed by different companies, possibly in different countries. This is quite common in European missions, for example.

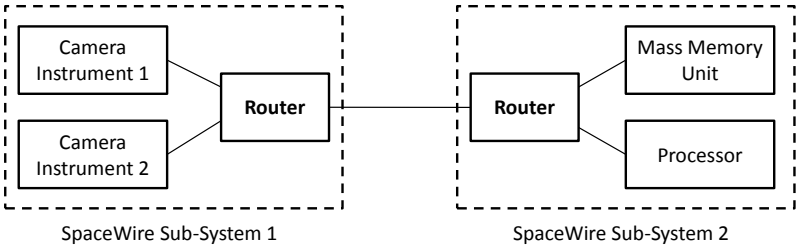


Figure 1: Example SpaceWire network containing two distinct sub-systems

A SpaceWire Internet Tunnel replaces a SpaceWire link in an onboard network, and consists of both software and hardware components. A SpaceWire cable representing one end of the link to be replaced by the SpaceWire Internet Tunnel is connected to a SpaceWire IP-Tunnel device. This device is then connected to a PC by a USB cable. Software running on the PC manages the Tunnel and allows traffic crossing the Tunnel to be monitored and recorded. A similar set-up is used at the other end of the link being replaced, and the software running on the two PCs tunnels traffic received on the SpaceWire links over a network to the other end. This arrangement is shown in Figure 2, where the two sub-systems from the example network in Figure 1 have been connected virtually using a SpaceWire Internet Tunnel. These two sub-systems may be in the same lab, or may be in different continents.

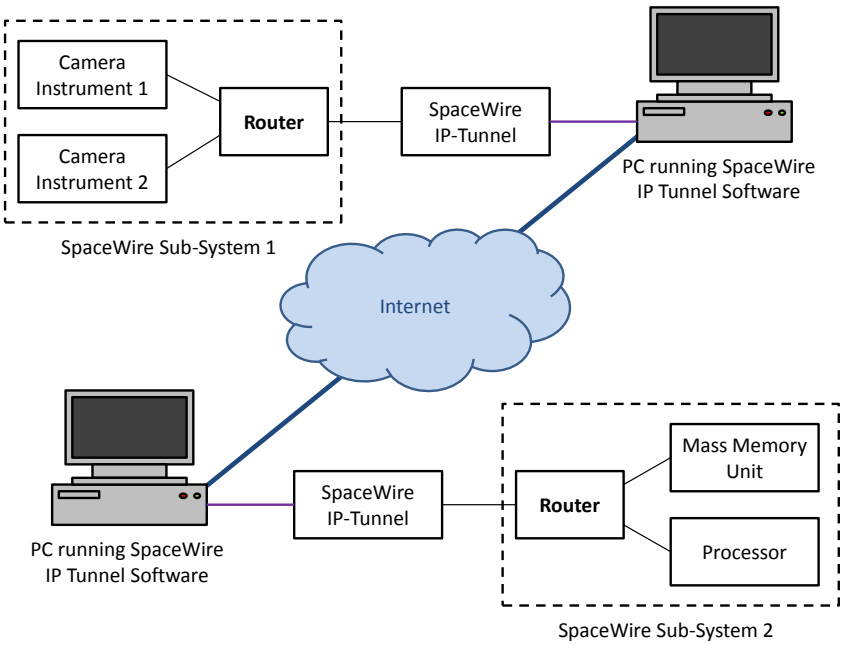


Figure 2: Example SpaceWire network containing two sub-systems integrated virtually

As well as exchanging data packets, the Tunnel also ensures that the link state is reflected at each end of the Tunnel. This means that if a link is disconnected at one end of the Tunnel, the link at the other end will also be disconnected. Other than the increased latency and reduced bandwidth, the Tunnel is almost transparent.

The hardware component of the Tunnel, shown in Figure 3, has two SpaceWire ports, allowing up to two Tunnels to be established. The software component is a Java application which can be run on Windows or Linux, as drivers for the SpaceWire-IP Tunnel have been written for these platforms. Any number of connections can be established using the Tunnel software, up to two for each connected Tunnel device.

Figure 3: The SpaceWire IP-Tunnel hardware

The SpaceWire Protocol Analyser is a module provided with the SpaceWire IP Tunnel Software which allows the traffic crossing a Tunnel to be monitored and recorded. As the SpaceWire Internet Tunnel is intended for use during integration testing, it is important that tools are provided to identify and correct any problems in the system.

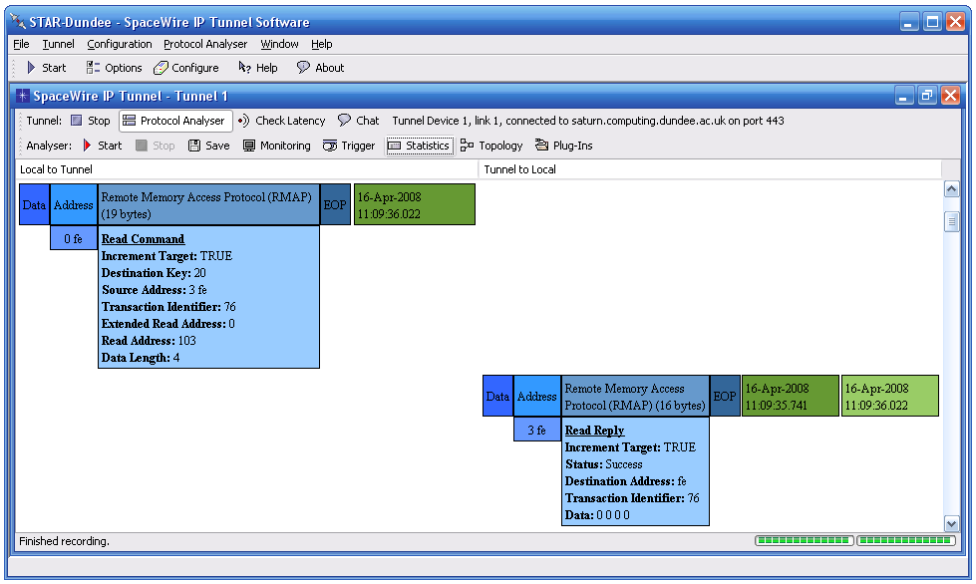


Figure 4: The main window of the SpaceWire IP Tunnel Software, showing recorded RMAP packets

The Protocol Analyser allows plug-ins for specific protocols to be loaded so that packets containing these protocols can be monitored and recorded. Higher-level protocol plug-ins can be written by users and loaded at run time, so potentially any protocols can be supported. A plug-in for the Remote Memory Access Protocol (RMAP) [6] is included with the Protocol Analyser. This allows the individual fields of recorded RMAP packets to be displayed, and also permits triggering on RMAP packets. A screenshot of the Tunnel Software showing some traffic recorded using the Protocol Analyser and formatted using the RMAP plug-in is shown in Figure 4.

3 TECHNICAL DIFFICULTIES AND OVERCOMING THEM

During development of the SpaceWire Internet Tunnel, a number of potential limitations were noted. Each of these limitations was identified as being in one of the following three areas:

- Issues relating to the bandwidth and latency restrictions of the Internet
- Security concerns related to sending data across the Internet
- Problems establishing connections over the Internet

The SpaceWire Internet Tunnel has mechanisms to address each of these areas, reducing or eliminating each limitation.

The first category of limitation is particularly problematic when tunnelling SpaceWire traffic. For example, if the time between routing two bytes of a packet is greater than the timeout period within a router, that router will terminate the packet and indicate there was a timeout error. To avoid such issues when tunnelling, the SpaceWire Internet Tunnel Software contains a number of mechanisms to greatly reduce the chances of timeouts being introduced due to the bandwidth or latency restrictions of the terrestrial network in use. These mechanisms are present at both the entrance and the exit of the Tunnel, as timeouts may also occur when a router is unable to send bytes as a link is already in use.

Despite these mechanisms there are some systems where the SpaceWire Internet Tunnel is not suitable. Such systems include those which expect packets within a bounded time period, or which have high bandwidth requirements.

To cope with the security concerns of tunnelling sensitive information over the Internet, all Tunnel traffic is sent using Transport Layer Security (TLS) [7], the successor to the Secure Sockets Layer (SSL). TLS is the protocol used by secure web pages which transmit sensitive information such as credit card details. All traffic is encrypted, and the protocol ensures the contents cannot be viewed or modified. To ensure that the Tunnel at the other end of a Tunnel connection is who they claim to be, all Tunnels can also be configured to use a password. When sent over the network, this password is encrypted using the same TLS protocol.

A number of problems were identified by users when establishing network connections between two Tunnel ends. These mainly related to firewall restrictions and proxy issues. Some organisations expect their users to connect to the Internet via a proxy server. Support was added to the Tunnel Software for proxy servers and a number of the authentication systems used by proxy servers.

Other organisations only allow Internet traffic on a limited number of TCP (Transmission Control Protocol) ports, such as those used for web pages (normally port 80). The SpaceWire Internet Tunnel allows any port to be used. In addition to avoiding problems with firewalls which only allow certain ports to be used, this also avoids problems with firewalls or proxy servers which check that the format of packets on a particular TCP port is in the correct format for that port. Using port 443, which is assigned for secure HTTP traffic being sent using TLS/SSL, the firewall or proxy server sees the encrypted Tunnel traffic and assumes the traffic to be in the correct format, as it cannot view the encrypted contents.

Another limitation when establishing network connections is the client-server nature of network connections: one end of a Tunnel acts as a server listening for connections, while the other end acts as a client and connects to that server. Most organisations are not happy for their users to run servers behind their firewalls, and block all attempts to connect to those servers through their firewall. The only exceptions to this rule are dedicated servers, such as web servers, which normally run on PCs which have been granted special firewall permissions to accept connections from outside the firewall.

Although some network administrators are happy to allow a Tunnel PC to accept Tunnel connections, some users found this was not permitted. This meant they could only establish Tunnels within their organisation, or with users in other organisations who had been granted the required firewall permissions. To address this limitation, the SpaceWire IP Tunnel Server was developed.

4 THE SPACEWIRE INTERNET TUNNEL SERVER

As mentioned in the above section, a limitation of the SpaceWire IP Tunnel Software is that one end of the Tunnel must act as a server, and therefore can require special firewall privileges. The SpaceWire Tunnel Server was developed to address this issue, allowing both ends of a Tunnel to act as a client.

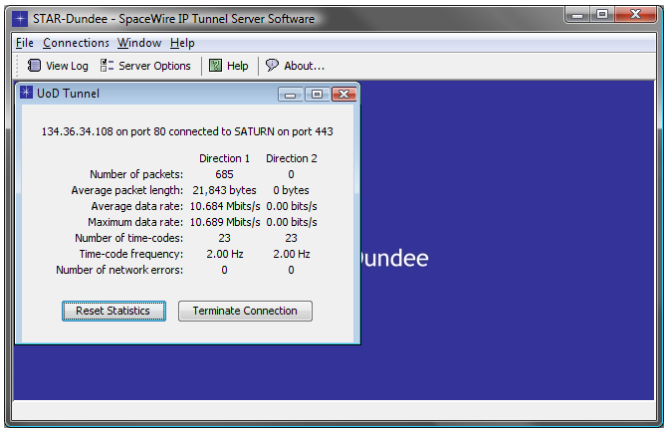


Figure 5: The main window of the SpaceWire IP Tunnel Server Software

The SpaceWire IP Tunnel Server is a software application which establishes connections between Tunnel pairs. A screenshot of this software in operation is shown in Figure 5. The Tunnel Server listens for connections from clients, and so must have the same firewall permissions as other servers. On establishing a connection, the Tunnel Server determines if the other connection in the pair has already been established. If it has, then a virtual connection is established between

the pair, and all traffic received from one client is passed on to the other. There is no difference than if the two clients were connected directly, so this has no effect on the Tunnel's mechanisms to cope with bandwidth and latency restrictions. Latency may be increased, however, as the traffic is now crossing two network connections.

Although a PC acting as a Tunnel Server must still have the same firewall permissions as a Tunnel end acting as a server, a Tunnel Server can manage multiple Tunnel client pairs. This means a single Tunnel Server could manage connections for a number of organisations. Alternatively, an organisation might want to setup their own Tunnel Server, just as they might have web and FTP (File Transfer Protocol) servers.

A further benefit of using a Tunnel Server is that the user does not need to know the address and port of the other end of the Tunnel, only those of the Tunnel Server. Connections are identified by name, and when connecting to a Tunnel Server a user can select a named connection from the list of those presently established. This can be particularly useful when establishing a number of Tunnel connections, or establishing connections with different Tunnel ends for different tests. For example, a user might connect to a connection named "Camera Subsystem" during some tests, and "Camera Subsystem Simulation" during others.

Communication with a Tunnel Server uses TLS to ensure the traffic cannot be viewed or modified, just as with a direct connection between two tunnel ends. In addition to the password used when establishing a connection, each named Tunnel connection can also have an associated password. To avoid connections from unauthorised users the Tunnel Server Software can deny access from specific addresses, or only allow specific addresses. The TCP ports to accept connections on can also be specified, as can the maximum number of connections that can be established.

As with the SpaceWire IP Tunnel Software, the SpaceWire IP Tunnel Server Software is written in Java. This means it can be run on any platform which supports Java and, as no communication with SpaceWire devices is performed, no specific drivers are required on the platform in use.

5 THE TOPNET PILOT STUDY

The theoretical advantages of virtual spacecraft integration and in particular the SpaceWire Internet Tunnel are quite obvious. However, until recently the real-world benefits (and limitations) when using the SpaceWire Internet Tunnel had not been investigated. An ESA funded pilot study, the TopNet Pilot Study, was completed earlier this year to investigate the benefits and limitations when in use within a real project environment. The study involved three consortia, each consisting of partners spread across Europe. Each consortium proposed experiments where SpaceWire devices situated in each of the consortium's partners would be virtually integrated.

The first stage of the study involved each of the consortia establishing Tunnel connections with the University of Dundee and exchanging SpaceWire traffic over that Tunnel. As well as giving the users the opportunity to familiarise themselves with the software and hardware, this also gave the University of Dundee an opportunity to identify any peculiarities in the users' networks. This lead to modifications to the Tunnel Software, with the addition of features such as proxy server support.

After familiarising themselves with the SpaceWire Internet Tunnel, the consortia then went on to conduct their experiments. These experiments were quite varied and covered a number of applications. While some experiments used software and hardware that had previously undergone integration testing, other experiments used software and hardware that was being tested together for the first time. Some experiments involved two forms of integration testing: both virtual integration testing and the traditional method of bringing all components to a single location. Numerous measurements were made during each of the experiments, providing valuable information on the impact a SpaceWire Internet Tunnel has in a SpaceWire network.

On completing their experiments, each consortium presented their findings and compiled a report containing their results and conclusions. The general consensus of those involved in the pilot study is that although there is still the possibility to make improvements, virtual spacecraft integration and the SpaceWire Internet Tunnel is a very useful tool for performing integration testing.

6 ADVANTAGES OF THE SPACEWIRE INTERNET TUNNEL

A number of the advantages of the SpaceWire Internet Tunnel are obvious:

- Integration testing is possible at an earlier stage of development
    - Saves time and money in correcting any problems
- Necessary travel is reduced
    - With associated financial and environmental savings
- Integration testing can be much more flexible
    - Integration testing can be performed at any time
    - Sub-systems can easily be replaced with simulators
- Geographical limitations are reduced
    - Cooperation between organisations involved in a project is improved

There are still limitations, however. There are some systems for which virtual spacecraft integration cannot allow proper integration testing to be performed. This includes those systems which have strict requirements on bandwidth and/or latency. An observation made by some of the contractors involved in the TopNet Pilot Study, however, was that it was often possible to perform limited integration testing for these systems using the SpaceWire Internet Tunnel. For example, by designing software to have configurable latency requirements, or even having a special "Tunnel" mode, initial integration testing could be performed to validate interfaces, etc. Once this testing is completed successfully, full integration testing can be performed.

The only other major limitation identified by the contractors was related to problems establishing connections, which should be addressed by use of the SpaceWire IP Tunnel Server, which was not available during the pilot study. Other issues identified by the contractors related to minor software issues and suggestions for improving the monitoring and recording capabilities of the SpaceWire Protocol Analyser. This emphasised the importance of analysis tools when performing virtual integration testing. Improvements to the software will continue to be made to add new features and address any issues identified.

A number of additional benefits of the SpaceWire Internet Tunnel not previously considered were also reported by the contractors:

- Virtual integration testing can be completed in a fraction of the time required for physical integration testing
- Electronic Ground Support Equipment (EGSE) does not need to be transported to an integration site when correcting problems, it can be virtually integrated
- When a system works using the Tunnel, it gives great confidence in the system
- The SpaceWire Protocol Analyser can be a very useful tool for monitoring and recording traffic, even when not tunnelling

7 CONCLUSIONS

This paper has noted a number of advantages of the SpaceWire Internet Tunnel. The TopNet Pilot Study has shown that these benefits can be realised in real projects. There may be some systems for which virtual spacecraft integration is not an ideal method of performing integration testing, but it has been demonstrated that some testing is still possible in such systems.

In conclusion, although physical integration testing of a system is still essential, virtual integration testing can be a very important stage in future projects.

8 ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of ESA for the work performed on the SpaceWire Internet Tunnel at the University of Dundee. We would also like to thank the contractors involved in the TopNet Pilot Study for their valuable feedback.

9 REFERENCES

1. S. M. Parkes, “Virtual Satellite Integration”, DASIA (Data Systems In Aerospace) 2001, Nice, France, May 2001.

2. S. Mills, S. M. Parkes, R. Vitulli, “SpaceWire Internet Tunnel”, DASIA (Data Systems In Aerospace) 2005, Edinburgh, Scotland, UK, May 2005.

3. S. Mills, S. Parkes, R. Vitulli, “Virtual Satellite Integration and the SpaceWire Internet Tunnel”, International SpaceWire Conference 2007, Dundee, Scotland, UK, September 2007.

4. European Cooperation for Space Standardization, “SpaceWire, Links, Nodes, Routers and Networks”, Standard ECSS-E-50-12A, Issue 1, European Cooperation for Space Data Standardization, February 2003.

5. STAR-Dundee, “STAR-Dundee Website”, STAR-Dundee, http://www.star-dundee.com/.

6. European Cooperation for Space Standardization, “SpaceWire Protocols”, Draft Standard ECSS-E-ST-50-11C, Draft 1.2, European Cooperation for Space Data Standardization, July 2008.

7. T. Dierks, E. Rescorla, “The Transport Layer Security (TLS) Protocol, Version 1.2”, Request for Comments 5246, Internet Engineering Task Force, August 2008.

SPACEWIRE MARGINS TESTER

Session: SpaceWire Test and Verification

Short Paper

Alex Kisin, Glenn Rakow

MEI / NASA GSFC

E-mail: alexander.b.kisin@nasa.gov; glenn.p.rakow@nasa.gov

ABSTRACT

The subject of this presentation is a SpaceWire Margins Tester which is intended to simulate major parameters to give designers an idea about SpaceWire link voltage and timing margin. The operational principal uses a good SpaceWire signal from a known source and degrades it to a point where the SpaceWire link under test will start to experience receive errors. Methods for simulating and testing the margin ranges of SpaceWire physical layer are shown on attached Power Point slides [1].

1. Problem

A common question that occurs during SpaceWire link testing is: "OK, it works now in a lab – but what are the worst physical layer conditions that this link can tolerate for a desired error rate?" It is very important to take this into account as the communication speeds and distances between SpaceWire links are increasing and it is difficult to verify this especially if some of the commonly used SpaceWire hardware components, particularly MDM connectors and some cables, may negatively impact link performance. In addition, temperature and active radiation may also contribute to link degradation.

As a result, the engineer needs to simulate SpaceWire conditions for various communication speeds. The parameters include: Data and Strobe skew to satisfy setup and hold timing, LVDS skew for each individual differential pair, "eye pattern" opening span, and LVDS receiver bias.

SpaceWire margin simulations are hampered by SpaceWire protocol and physical layer difficulties.. There are no explicit provisions in the existing protocol standards to support error tests. The constant current used in LVDS links makes margin testing very cumbersome. Last, the different cable wire gauges and their lengths provide major impact on transmission quality.

2. Suggestion

The proposed SpaceWire Margins Tester or SWMT simulates most of SpaceWire physical layer parameters to allow the designer to determine each particular

SpaceWire link margins and predict possible performance degradations at worst anticipated conditions.

2.1 Methods and Assumptions

The SWMT receives the SpaceWire signal from a known source, degrades the margin in the SpaceWire signal by error injection, and retransmits the signal to the device under test (DUT) until a link error is detected. Link errors are detected when the link goes silent during a restart (clock dropout condition). The link restarts serve as substitutes of conventional bit error rates (BER). By knowing link protocol parameters, it is possible to translate resulting frame error rate (FER) to more commonly used BER.

It is important to note that only the DUT receiver is subjected to simulation. The DUT transmitter site is needed to respond with valid test results. It is intended that the distance between the SWMT and the DUT is minimal so that the DUT's transmitter will have minimal effect on the link's performance. As a result, the SWMT standalone instrument is designed to satisfy the above mentioned methodology. A more precise Complex SWMT is also possible and its specs are discussed in attached slides presentation [1].

3. Technical Characteristics

3.1 Simulated Parameters

Of all the SpaceWire parameters that can be simulated, only the three main parameters of skew, span, and bias are simulated.

3.1.1 Skew

Variable skew simulation between the Data and Strobe SpaceWire signals is needed to define minimal setup and hold time that the DUT can tolerate to have reliable clock extraction. Skew is simulated by using delay elements to slide one signal versus the other in both directions from nominal "0" timing point. As a result, the minimal time interval between Data and Strobe can be determined. The suggested range of signal sliding is ± 30 nS (60 nS of total covered interval) with around 1 nS resolution. This skew corresponds to an equivalent minimum communication speed of 16.6 Mbps. The SWMT will automatically select the skew range based on the operating frequency of the link. The SWMT supports links speeds of up to 310 Mbps (3 nS bit time). The maximum communication speed is set by the operator at the beginning of every test, starting from 10 Mbps and with step resolutions of 10 Mbps. Because of some implementation difficulties, differential pair skew is not tested in a current version of SWMT.

3.1.2 Span

Variable span is needed to simulate minimal voltage at receiver inputs. This is the trickiest parameter to simulate for the LVDS physical layer. The SWMT uses modulated high speed amplifiers with voltage outputs to adjust the span. The span is defined by loading the DUT's termination resistor so the differential voltage drop

across resistor will correspond to LVDS DUT receiver allowable operating range. The maximum drop across the 100 Ω termination resistor is 700 mV p-p at 3.5 mA nominal current. The minimal drop is specified by LVDS specification document [2] at ± 100 mV, or 200 mV p-p. To have some extra margins, the SWMT will set its operational range from 180 to 720 mV p-p with programmable steps of 20 mV. To minimize signal loss across the cable between the DUT and SWMT, the shortest possible cable (0.5-1 m) with largest wire gauges (AWG 22-24) is required. This approach was successfully tested by NASA GSFC to troubleshoot one of its spacecraft electronic units.

3.1.3 Bias

Variable bias, which also can be referred to as common mode voltage (CMV), is rather simple to simulate. The LVDS specification document [2] defines receiver's nominal bias at around +1.2 V. The SWMT will set its simulation range as ± 1.2 V from nominal or 0 V to +2.4 V on absolute scale: all with programmable steps of 200 mV. The SWMT will also be able to detect ground fault condition when its pin 3 is shorted to overall shield or DUT chassis.

3.2 Simulation

The parameter's simulation can be either static, by manually changing one parameter at a time and monitoring result; or dynamic, by defining the working range of change of one or more of desired parameters and allow the SWMT to try all available combinations within range. In dynamic mode, only one parameter is changed at a time on a 1 ms time interval. At the end of an interval DUT generated clock dropout counter is assessed and if new value is detected it is immediately reported.

The longest time that it takes to run through all possible combinations at 10 Mbps communication speed is 23 seconds, but this time cycle may be greatly reduced if higher speeds and/or tighter allowable ranges are selected.

3.3 Connection Modes

Two connection modes are available: Pass mode – from external Source to DUT and Loop mode – from DUT transmitter back to DUT receiver; in this mode the DUT has to establish a self loopback data link. In Pass mode, the DUT transmission is passed to an external Source unaltered; in this case SWMT is connected in to a break between SpaceWire link source and DUT.

4.0 Control

There are two ways to control the SWMT.

4.1 Local

Local control is done through front panel buttons and dials and provides manual access to all of the simulated parameters. In addition, each SWMT contains up to 8 different profiles (combined set of simulated parameters) stored in non-volatile SWMT memory.

4.2 Remote

Remote control is optional and is provided from remote computer through an electrically isolated USB 2.0 port. Isolation is required to provide valid Bias simulations. Remote computer duplicates all local control operations and provides for a more automated option.

5. Telemetry

The main telemetry parameter is dropout counter value. Every second front panel display shows the accumulated number of occurred dropouts and current test elapsed time. The test duration can be as long as 99,999 seconds (in excess of 27 hours) and can show up to 99,999 link dropouts. Once started, a test can be terminated either by the operator, or by the timer or dropout counter overflow. The remote computer is able to track telemetry with 1 ms resolution (vs. 1 second for visual display) together with current profile's setup.

6. Future Developments

In addition to a Simple (or Coarse) tester, a Precision tester can be made. This requires changing the structure of existing SpaceWire IP cores by adding a module for testing. This change would allow performing BER detection inside IP (instead of FER) and report it on demand. Activation of this test mode could be done through a dedicated operational code from outside. Implementing this change allows an external tester not to inject errors in to the pre-existing data from some other source, but to generate its own test patterns and provide a much higher level of test fidelity especially in area of simulating Data and Strobe signals skew and differential pair skew. These new features can be discussed in greater details during the ISC-2008 meeting and the prototype could be ready for ISC-2009.

7. References

1. SpaceWire Physical Margins Tester – ISC-2008 presentation
2. ANSI TIA/EIA-644A [LVDS] Standard

8. Acknowledgments

The authors would like to thank Michael Pagen for his suggestions that helped to develop Simple SWMT concept.

USING SPACEWIRE AS EGSE INTERFACE

Session: Test & Verification

Short Paper

Anders Petersén, Torbjörn Hult

Saab Space, SE-405 15 Göteborg, Sweden

E-mail: anders.petersen@space.se , torbjorn.hult@space.se

ABSTRACT

The demand for an EGSE interface with increased speed can be met with a lot of different interfaces available and among other high speed serial links SpaceWire can be used.

The new generation of Saab Space Data Handling systems has the processor function based on the COLE system-on-chip device. The COLE ASIC is based on the LEON2-FT processor [2], with additional features. Eight SpaceWire links supporting RMAP [1] in hardware are included. The COLE ASIC provides an Enhanced Debug Support Unit (E-DSU) with an increased trace buffer and supports filtering of the trace buffer storage. The trace buffer can also be dumped in real-time on a dedicated high speed SpaceWire link without affecting the timing on the internal AMBA bus.

1 INTRODUCTION

In early data handling system based around the ERC32 processor [3], the only available EGSE interface was the processor UART with a baud rate of 38400 baud. This has been a considerable bottleneck during the verification process, since the test application software used tends to grow in size leading to increased time for downloads. When designing the next generation data handling system, an EGSE interface with increased speed was one of the drivers. This can be achieved with a lot of different interfaces available, Ethernet, high-speed UART, and among other high speed serial links, SpaceWire. Since SpaceWire consists of space qualified hardware, the main advantage of using SpaceWire as an EGSE interface is that the same interface can be used on bread board models as well as flight models.

2 COLE SYSTEM ON-CHIP DEVICE

The new generation of Saab Space Data Handling systems has the processor function based on the COLE system-on-chip device. The COLE ASIC developed by Saab Space is a combination of the COCOS I/O controller ASIC and the LEON2-FT processor [2], with additional features. In the COLE ASIC a high speed , (200 Mbps when receiving, 160 Mbps when transmitting) SpaceWire interface is implemented. Eight SpaceWire links are included; each link can support RMAP [1] in hardware. The LEON2-FT processor is equipped with a Debug Support Unit (DSU) providing breakpoints, watchpoints and trace facilities.

2.1 DEBUG COMMUNICATION LINK (DCL)

The LEON2-FT processor provides a Debug Communication Link (DCL) which is a UART interface with a simple protocol to perform read and write operations on the on-chip AMBA bus [4]. This interface is the normal EGSE interface used to control the DSU.

Since the COLE on-chip SpaceWire links support the SpaceWire RMAP protocol, and the DSU is a memory mapped device connected to the AMBA bus, it is possible to use one of the SpaceWire links as a high-speed software debug interface instead of the DCL UART. Another advantage is that this SpaceWire link can be used as a spare functional interface after launch.

2.2 ENHANCED DEBUG SUPPORT UNIT (E-DSU)

When working with a high-performance cache based architecture such as LEON it is not possible to get a real-time trace over all instruction and data transfers on the AMBA bus. There are different approaches to this problem; one can either provide a high-speed link for dumping data or some techniques to reduce the amount of data.

The COLE Enhanced Debug Support Unit (E-DSU) provides an increased trace buffer (8 times larger than the AT697) and supports filtering of the trace buffer storage, on calls, branches, return instructions, traps and address area. The trace buffer can also be dumped in real-time on a dedicated high speed SpaceWire link without affecting the timing on the internal AMBA bus [4]. The E-DSU also supports the possibility to read the IU Program Counters (PC and NPC) while the processor is executing.

Memory and I/O access counters as well as cache hit counters have been added to the E-DSU for collection of statistics for profiling purposes.

2.3 SPACEWIRE INTERFACE

The COLE SpaceWire interface is based on the ESA SpaceWire CODEC IP block [6] supplemented with a dedicated design that interfaces the AMBA bus. The CODEC is used in double data rate mode and can transmit at twice the clock rate of the COLE chip, up to 160 Mbps. In receive mode the link interface can operate up to 200 Mbps. The interface control logic provides several DMA channels with hardware round-robin arbitration between the channels when accessing the link, allowing multiple software tasks to easily share the same link.

The RMAP read and write protocols are supported in hardware, while read-modify-write must be implemented in software. The interface hardware also supports the CCSDS Packet Transfer Protocol [1], using the User Application field as a virtual channel ID to be able to receive up to seven different packet streams on different DMA channels.

3 SOFTWARE DEVELOPMENT FACILITY

A typical Software Development Facility (SDF) for COLE is shown in the figure below where the target Panther board (a processor board including the COLE ASIC) is connected to an Ethernet network via a Test Equipment (TE) PC and an USB/Ethernet-to-SpaceWire bridge. The developers run the Software Development

Environment on their respective workstations and load and debug the target Panther board through the Ethernet network and the Test Equipment PC.

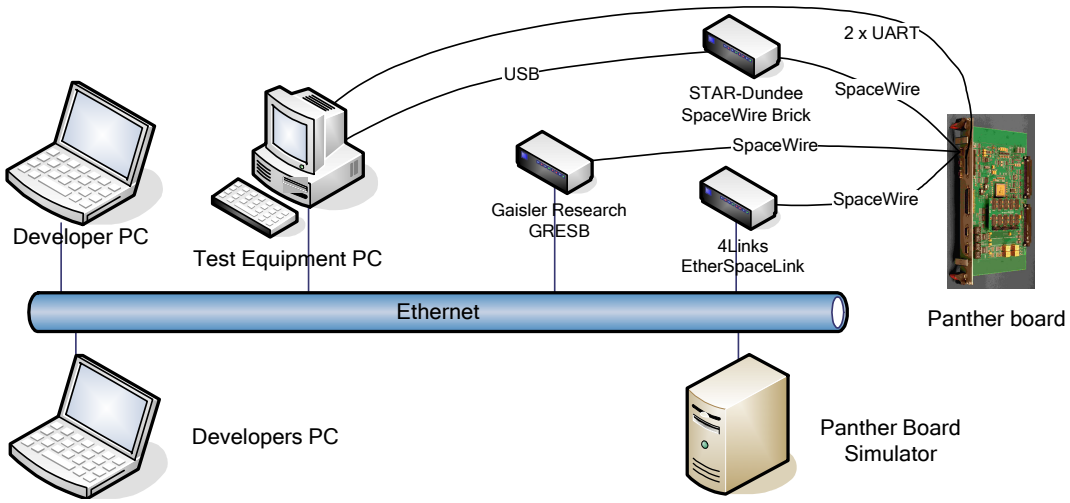


Figure 1 – Software Development Facility (SDF)

Adding a potent high-end PC to the network, running the COLE and Panther board target simulator enables the software to be developed transparently on real target hardware or the simulated target.

4 SOFTWARE DEVELOPMENT ENVIRONMENT

The Software Development Environment (SDE) for COLE is built around the Eclipse Integrated Development Environment (IDE) framework [5] and a COLE Broker.

Eclipse [5] is the well known open source software development framework. COLE Broker is a Saab Space application running on the Test Equipment PC and responsible for managing all debug communication with COLE. Tools dedicated to a certain software development or debug activity interface COLE through the "broker" in order to facilitate simultaneous and efficient use of the available hardware among several different tools and developers.

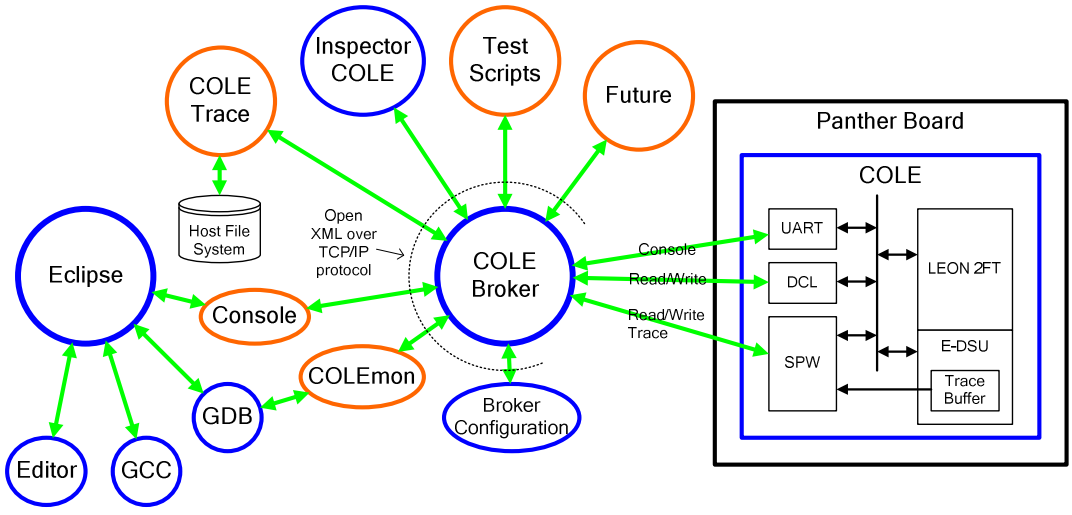


Figure 2 - Software Development Environment

Tools surrounded by a blue (dark) circle in Figure 2 are available. The orange (grey) tools are planned and the "future" tool is just a place holder for future tool ideas. The code editor, compiler (GCC) and debugger (GDB) are almost out-of-the-box tools already integrated with Eclipse. It is possible to use Gaisler Research's GRmon as the bridge between GDB and COLE however it has the drawback that it may conflict with other tools also interfacing the COLE resource. COLEmon is a lightweight solution for this, it is not as powerful as GRmon but it will do the job.

In order to get a fully "native" look, the console (one of the COLE UARTs can be used for this) should appear as when running and debugging a native application.

The trace tool is intended to download the filtered and compressed trace which is dumped by COLE in real-time through the high-speed SpaceWire link, store it, decompress it, and perform offline analysis for offline trace back and forth, timing measurements, profiling, etc.

Inspector Cole comes in handy when you are running your application and you want to see what is going on within the COLE ASIC. All COLE registers can be read and presented in a decoded way where each bit field within a specific register gets a meaning. It is also possible to alter a register for error injection or whatever need you have. Although COLE is a quite large and complex system-on-a-chip it is easy to navigate to the location you are interested in and there find out what is going on.

If the Software Development Facility, or a derivative thereof, is used also as Software Validation Facility then the COLE broker is easily interfaced from the test scripts via the used open XML over TCP/IP protocol.

5 CONCLUSION

To increase test application software download performance in the COLE ASIC, a dedicated high speed SpaceWire link with support for the RMAP protocol has been used. With the SpaceWire link it is also possible to perform a real time trace with the enhanced debug support unit (E-DSU) that is capable of filter and compress the trace.

A software development environment taking advantage of the high speed SpaceWire link and COLE ASIC features is built around the Eclipse Integrated Development (IDE) framework and a COLE Broker.

6 REFERENCES

1. ECSS-E-50-11A, SpaceWire Protocols, Draft 1.3, July 2008

2. The Leon2-FT Processor User´s Manual, Version 1.0.15, June 2005

3. Atmel, TSC695F SPARC 32-bit Space Processor User Man, 4148H-AERO-12/03

4. AMBA™ Specification, ARM IHI 0011A, Rev. 2.0

5. http://www.eclipse.org

6. University of Dundee, SpaceWire CODEC VHDL User Manual, Issue 1.2, 27 October 2005

IMPROVEMENTS IN SPACEWIRE TEST

Session: Test & Verification

Short Paper

Paul Walker, Barry Cook

4Links Limited, Bletchley Park, MK3 6EB, England

E-mail: paul@4links.co.uk , barry@4Links.co.uk

ABSTRACT

During system test and integration, recordings are made of the interactions between the various subsystems. Both the bandwidth per link and the number of links in a typical SpaceWire system present challenges for such recording. Furthermore, the recordings need to be both time tagged and dated. The time tag is used for analysis of functionality and latencies, and the date for archival purposes so that there is no confusion between different recordings. With the move towards SpaceWire-RT, there is an increased need both for precise SpaceWire Time Codes and for measuring the response of the system to those Time Codes.

The new Multi-link SpaceWire Recorder and Absolute Time Interface meet these challenges and work together to provide recordings of many links stored on different discs on different computers (and even in different countries) and yet all coordinated and dated to remarkable precision. In addition, zero-jitter time codes can be generated.

1 RECORDING TRAFFIC ON SPACEWIRE LINKS

MIL 1553 and CAN are both buses and both operate at 1Mbit/s; recording data at this rate is not a great challenge. SpaceWire [1] link speeds are one or two orders of magnitude faster and, without the constraint of a single bus, system speeds can be more than an order of magnitude greater still. Recording SpaceWire traffic is thus vastly more of a challenge. The new Multi-link SpaceWire Recorder [3] meets this challenge.

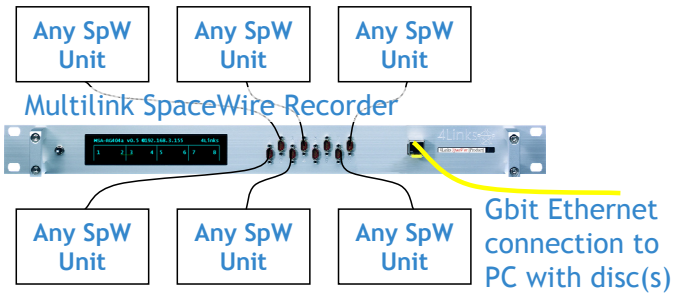


Figure 1: Multi-Link SpaceWire Recorder

The Multi-Link SpaceWire Recorder (MSR) enables passive recording of SpaceWire transfers, on up to four links per MSR unit. It monitors packets in both directions on each link, sending the data to a computer for recording and analysis. Software is provided to help manage and search the recordings. The MSR brings to SpaceWire the comprehensive records that are available from earlier and much slower on-board technologies such as MIL1553.

As well as recording the data flowing over a SpaceWire link, the MSR can optionally be set to trigger a waveform capture of the wire signals on a variety of events, including low-level SpaceWire errors.

When recording both directions of several SpaceWire links, it is useful (or essential) to be able to relate the timing of each packet in each direction on each link to the other packets.

1.1 Time Tagged Recordings

The timing relation between packets is provided by Time Tags on the start and end of each packet. A short example recording is shown in Figure 2. [For the technically minded, notice the level of detail provided by the time tags here. The interval between packets is 840ns, which is 600ns for the three Bytes of each packet at 50Mbits/s plus 80ns for the End of Packet plus a further 160ns for a Null token between packets.]

```
10.079 194 952 0s  1<--2  Data @0000  05 55 30
                   1<--2  EOP at 10.079 195 552 0s (SOP + 0.600us)
10.079 195 792 0s  1<--2  Data @0000  05 55 30
                   1<--2  EOP at 10.079 196 392 0s (SOP + 0.600us)
```

Figure 2: Time tagged record of two short packets

Even on such short packets in one direction of a single link, this time-tagged recording can be extremely useful. Figure 3 shows a slightly later set of packets from the recording of Figure 2. Packets were sent in pairs separated by 100ms, to allow an action to complete before trying to start the next action. The first two packets in Figure 3 are a valid pair, but the third packet immediately follows them rather than waiting for the 100ms, and then the fourth packet is actually 200ms later (the millisecond times of these incorrect arrival times are shown in red in the Figure). The incorrect arrival times resulted in some of the packets being ignored, and explained a system integration failure. Knowing the cause of the failure made it easy to design a workaround to the problem.

```
10.479 772 121 4s  1<--2  Data @0000  05 55 32
                   1<--2  EOP at 10.479 772 721 4s (SOP + 0.600us)
10.479 772 961 4s  1<--2  Data @0000  05 55 37
                   1<--2  EOP at 10.479 773 561 4s (SOP + 0.600us)
10.479 773 800 0s  1<--2  Data @0000  05 55 70
                   1<--2  EOP at 10.479 774 401 4s (SOP + 0.601us)
10.680 351 792 0s  1<--2  Data @0000  05 55 73
                   1<--2  EOP at 10.680 352 392 0s (SOP + 0.600us)
```

Figure 3: Time tagged records showing incorrect intervals between packets

1.2 Absolute Time, Datation, and SpaceWire-RT

Recordings such as are described here are often made during system integration, both to provide information in the event of a failure and to record the actual behaviour when the system is behaving correctly. So it is useful to tag the recordings not only with relative time but with an absolute date and time.

Furthermore, with the move towards SpaceWire Real-Time, there comes a need for test equipment both to generate a variety of Time Codes and to measure their arrival times (as well as packet arrival times) precisely.

2 ABSOLUTE TIME INTERFACE (ATI)

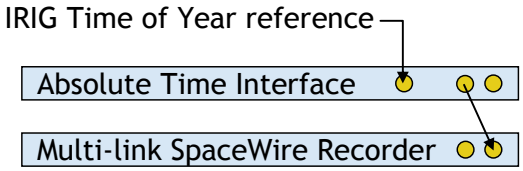


Figure 4: Absolute Time Interface and Multi-Link SpaceWire Recorder

The 4Links Absolute Time Interface (ATI) [4] enables one or more 4Links test units in a test system to be synchronized to an accurate "Time of Year". The Time of Year reference is input to the ATI by a signal conforming to the IRIG B002 standard [2] for signalling Time of Year. Products to this standard are available from many suppliers, and the time signals can be derived from sources such as GPS, Loran, or from an on-board satellite time reference.

The recordings are presented almost identically to the undated recordings, except there are fields for the day of the year as well as hours, minutes and seconds (to ten decimal places). Figure 5 shows a sequence of two packets presented in this format.

```
Day:hh:mm:ss  ms  us  ns     Flow   Activity
288:15:25:10.263 728 870 1s  1<--2  Data @0000  01 02 03 04  05 06 07 08
                             1<--2  Data @0008  09 0A 0B 0C  0D 0E 0F 10
                             1<--2  Data @0010  11 12 13 14  15 16 17 18
                             1<--2  Data @0018  19 1A 1B 1C  1D 1E 1F 20
                             1<--2  Data @0020  21 22 23 24  25 26 27 28
                             1<--2  Data @0028  29 2A 2B 2C  2D 2E 2F 30
                             1<--2  Data @0030  31 32
                             1<--2  EOP at 288:15:25:10.263 731 370 1s (SOP + 2.500us)

288:15:25:15.249 072 811 5s  1<--2  Data @0000  01 02 03 04  05 06 07 08
                             1<--2  Data @0008  09 0A 0B 0C  0D 0E 0F 10
                             1<--2  Data @0010  11 12 13 14  15 16 17 18
                             1<--2  Data @0018  19 1A 1B 1C  1D 1E 1F 20
                             1<--2  Data @0020  21 22 23 24  25 26 27 28
                             1<--2  Data @0028  29 2A 2B 2C  2D 2E 2F 30
                             1<--2  Data @0030  31 32
                             1<--2  EOP at 288:15:25:15.249 075 311 5s (SOP + 2.500us)
```

Figure 5:Recorded packets, time tagged to Time of Year, from synchronizing to ATI

This synchronization can be used with any number of Multi-link SpaceWire Recorders, each connected to a different computer (and discs), to record SpaceWire traffic at a far higher bandwidth than is possible on a single computer or disc. The common date and time reference means that all the recordings can be coordinated.

The units synchronized to the ATI can include bridges such as the EtherSpaceLink, Diagnostic SpaceWire interface or SpaceWire Packet Generator, as shown, for example, in Figure 6. The reconfigurability of the test units means that the same hardware can be used for active test stimuli from the bridge units and passive recording from the MSR, often without needing to unplug cables to the flight equipment.

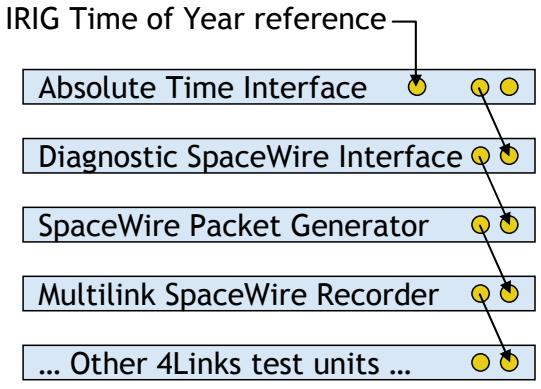


Figure 6: Synchronizing different test units

2.1 ISSUING AND RECORDING PRECISE TIME CODES

The ATI also has a single SpaceWire port which outputs a 1 pulse per second SpaceWire Time Code synchronized to Time of Year. With the Time-code Generator option (TG), the ATI outputs up to four independent and configurable Time Codes, all synchronized to Time of Year. Time codes are generated with zero jitter.

With the evolution of SpaceWire towards SpaceWire-RT, there is increased need for Time Codes and for instrumenting their distribution. By generating precise Time Codes and by synchronizing test units to measure arrival times of packets and Time Codes, the ATI enables precise measurement of Time Code arrival times and Jitter. Figure 7 shows a short example recorded by the MSR of Time Code arrival.

```
Day:hh:mm:ss  ms  us  ns     Flow   Activity
288:15:19:31.000 000 000 5s  1-->2  Time Code 00
288:15:19:32.000 000 000 5s  1-->2  Time Code 01
288:15:19:33.000 000 000 5s  1-->2  Time Code 02
288:15:19:34.000 000 000 5s  1-->2  Time Code 03
288:15:19:35.000 000 000 5s  1-->2  Time Code 04
288:15:19:36.000 000 000 5s  1-->2  Time Code 05
288:15:19:37.000 000 000 5s  1-->2  Time Code 06
288:15:19:38.000 000 000 5s  1-->2  Time Code 07
288:15:19:38.869 933 928 8s  1<--2  Data @0000  01 02 03 04  05 06 07 08
                             1<--2  Data @0008  09 0A 0B 0C  0D 0E 0F 10
                             1<--2  Data @0010  11 12 13 14  15 16 17 18
                             1<--2  Data @0018  19 1A 1B 1C  1D 1E 1F 20
                             1<--2  Data @0020  21 22 23 24  25 26 27 28
                             1<--2  Data @0028  29 2A 2B 2C  2D 2E 2F 30
                             1<--2  Data @0030  31 32
                             1<--2  EOP at 288:15:19:38.869 936 428 8s (SOP + 2.500us)
288:15:19:39.000 000 000 5s  1-->2  Time Code 08
288:15:19:40.000 000 000 5s  1-->2  Time Code 09
```

Figure 7: Recording of Time Code together with a data packet

3 CONCLUSIONS

The Multi-link SpaceWire Recorder can record time tagged traffic over many links, at many hundreds of aggregate Mbits/s, with recordings being made on several different computers. Synchronizing these recordings with the Absolute Time Interface makes it possible to inter-relate the different recordings as if they were a single recording.

This powerful logging and datation, together with precise, zero jitter, issue of Time Codes, creates a powerful tool for accurately recording the temporal behaviour of a complete SpaceWire system or any part of it. Synchronizing with a global time reference such as GPS extends this capability to remote testing and virtual satellite integration between multiple sites.

4 REFERENCES

1. The ECSS-E-50-12 Working Group, “ECSS-E-50-12A 24 January 2003, SpaceWire - Links, nodes, routers and networks”, published by the ECSS Secretariat, ESA-ESTEC, Requirements & Standards Division, Noordwijk, The Netherlands
2. Range Commanders’ Council, Telecommunications and Timing Group, IRIG Standard 200-4 “IRIG Serial Time Code Formats”
3. 4Links Limited, “Multi-link SpaceWire Recorder: product outline”
4. 4Links Limited, “Absolute Time Interface: product outline”

Test & Verification 2

Thursday 6 November
13:20 – 14:20

TOOLSET FOR TEST AND VERIFICATION OF IP-BLOCKS WITH SPACEWIRE INTERFACE

Session: SpaceWire Test and Verification

Short Paper

Elena Suvorova

St. Petersburg State University of Aerospace Instrumentation

67, Bolshaya Morskaya st. 190 000, St. Petersburg RUSSIA

E-mail: suvorova@aanet.ru,

ABSTRACT

Toolset for test and verification of RTL, post-synthesis and post-implementation models is usually based on BFM. The BFM (Base Formal Model) typically is used for testing and verification of interfaces that correspond to different standards. (Other term BFM – Base Functional Model is typically system level model of a particular device or IP-block). A toolset obligatory includes also generators of test sequences (or a prepared set of test sequences) and modules for monitoring and processing of test results. Often the toolset also includes an expanded set of components for simulation and performance estimation of the device in context of a real system. This expanded set could include switches, memory blocks and other devices. We suggest to use this approach for SpaceWire toolset development. We specify the SpaceWire BFM as a multilayer structure, every layer of which corresponds to a layer of the SpaceWire standard. It allows to use the BFM for test and verification of SpaceWire controllers with support of different SpaceWire layers and at different stages of design. For example, if we plan to test IP-block, that includes layers from character to packet we will use only corresponding layers of a BFM in test shell. In many cases the detailed test of physical level is very important problem for SpaceWire product designers. BFM includes physical level also.

The important task of coordination between test tools and tested device or IP-block settings is considered.

1 BFM STRUCTURE

Suggested BFM of SpaceWire is multilevel structure. Levels of BFM correspond to layers of SpaceWire standard. All levels of BFM exclude signal and physical are described on SystemC. These two down layers could be described on SystemC, VHDL or Verilog (dependently on design tools used for simulation, and on Unit Under Test (UUT) description language)

Let's consider the BFM structure. Figure 1 illustrated information flows between levels of model.

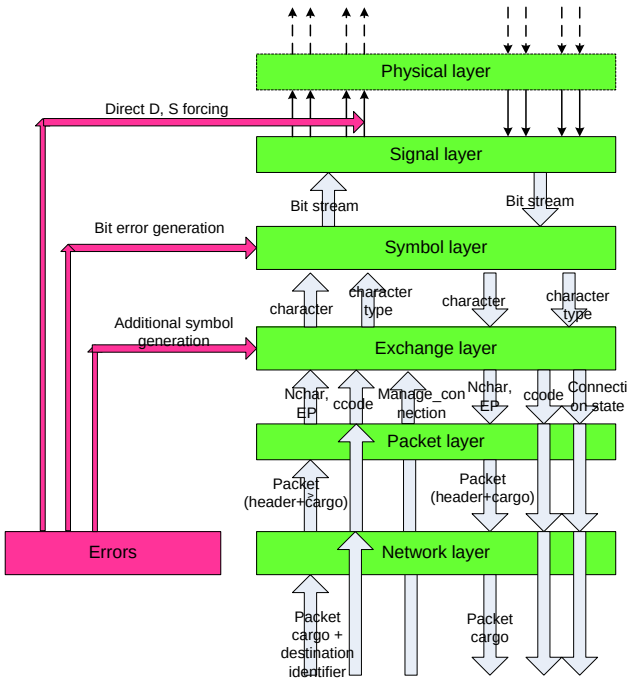


Figure 1. – The information flows in BFM model

Main information flows defined in SpaceWire standard are shown by grey arrows. One function of BFM is verification of UUT behavior when external errors appear. Therefore BFM include error situation generation possibilities. Suggested BFM allow generation errors of exchange, symbol and signal level.

The exchange level of BFM includes features for data flow control errors generation. It could send to channel Nchar symbols that are not credit. The BFM allow sending to channel FCT symbols that credit more than 56 Nchars. Also BFM allow sending to channel different symbols independently from current state of state machine. For example it allows to send Nchar symbols in states ErrorWait, Ready, Started, Connecting.

Possibility of modifying (inversion) of any bits of symbols exists on symbol level BFM (after generation of true bit representation for the symbol).

On signal level BFM allow change values of D and S lines in any time moments. As result we can simulate situations when signals D and S changed together or when time interval between D and S change is very short.

These possibilities allow testing system behavior in case of different types of external faults and noise. On figure 1 the control flows for error generation are marked by red arrows.

Information exchange between neighbor levels of BFM is going via access points that are logical ports. Logical port is a class that includes set of methods for information exchange. For example let's consider interconnection between packet level and exchange level. This interconnection includes three access points for transmission to exchange level: for data and end packet symbols, for control symbols (time, interrupt and acknowledge codes), for interconnection management. The access point (logical port) for control codes transmission includes next methods:
int send_Ccode(t_code Ccode_);
bool ready_to_send_Ccode();
t_code receive_Ccode();
bool received_Ccode();
The test generator and results controller could be connected to different levels of BFM dependently on set of SpaceWire layers includes in UUT. Also different levels of BFM could be connected to corresponding levels of UUT. The special wrappers are used between BFM and UUT interface because the interface of UUT is specified in terms of signals (class sc_signal).

2 THE STRUCTURE OF SUGGESTED TOOLSET

In this article we suggest method and toolset for test and verification of IP-blocks and devices that includes SpaceWire controllers some levels of protocol SpaceWire. One of important tasks for test development is control of UUT regime. For example we need test UUT in link start and auto start regimes, change transmission rate. If UUT includes network layer, then we need manage routing table. Correspondingly suggested test tools include not only BFM, test generators, test controllers and wrappers, but also special components for UUT configuration. Configuration process is typically specific for every concrete type of devices or IP-blocks. Interfaces for IP-block configuration could be differ for different blocks. Also the time between writing of new setting and real change could be varied essentially. We divide all UUT to two groups. First group includes UUT that are fully configurable via external interface. For example into this group included coder-decoder SpaceWire IP-block and SpaceWire controller IP-block for systems with internal processor.

For such UUT configuration our toolset include special component that allows transfer of configuration information via typical parallel synchronous memory interface. This component also includes functions for translation to/from signal based interface. Then user could describe wrapper from this interface to interface of his IP-block on any hardware description language. The example of test shell for such IP-blocks based on suggested toolset is represented on figure 2. On this figure components from suggested toolset are green.

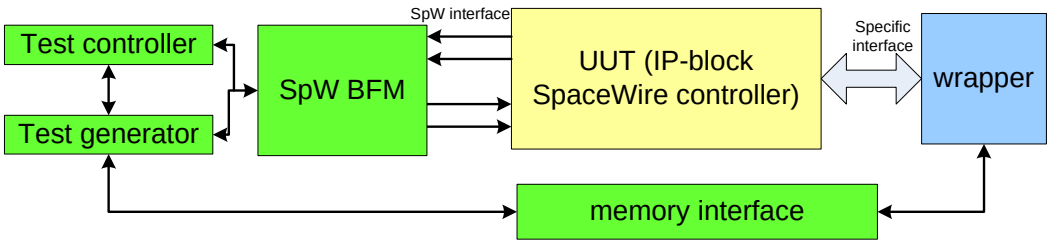


Figure 2. – The example of test shell structure for IP-blocks

Additionally toolset includes two special interface components. First of them intends for connecting to coder/decoder SpaceWire with typical interface represented on figure 19 of SpaceWire standard [1]. Second interface component includes two sub interfaces FIFO for data packets sending and receiving. The data line width is parameterized (number of bit lines s 2^n). Data packets are aligned to words boundaries. Interface includes byte valid lines. This special interface also includes sub interface parallel synchronous memory interface for working with state and regime registers. The control codes also could be sending and receiving via this interface.

Second group includes devices (UUT) with internal processor or automata. For these devices coordination between internal settings and test program in test shell typically is very difficult.

These devices could be divided to two subgroups. First subgroup includes devices configured only via SpaceWire interface. Second subgroup includes devices that could be controlled via other interface, for example interface with external memory.

In this article we consider possible decision of configuration problem for UUT with internal processor. The test program could be loaded to such UUT. But in this case appear the problem of synchronization between test program in UUT and test program in test shell. Therefore we suggest other approach. In frame of this approach the control of testing process is made by test program placed in test shell. This program when need could observe state of UUT and set regime of UUT with using of special commands. These commands are written one by one to fixed addresses of external memory model that accessible by internal processor of UUT. The simple program (written on C) that read and executes these commands is loaded to UUT. The interface component of test shell implement monitoring of reading commands by UUT and define time moments when next command could be written to memory. The program loaded to UUT is universal, for porting it to other device we need only correct base address of external memory. This scheme provides small time interval between start settings and activation new regime. This feature remove problem of synchronization between test program in test shell and test program in UUT.

Usually for connection of external memory is used parallel synchronous or asynchronous memory interface or sequential interfaces such as I2C, SPI or USB. Correspondingly suggested toolset includes components for interconnection external memory via these interfaces.

The example of test shell based on suggested toolset for device with internal processor is represented on figure 3.

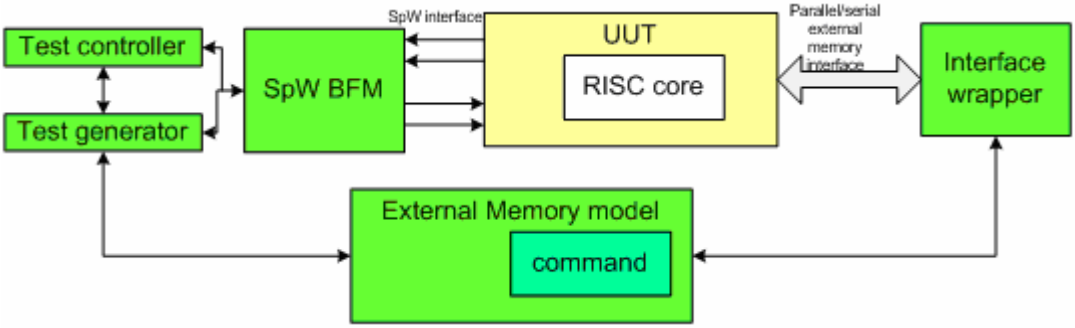


Figure 3. – The example of test shell for device with internal processor.

Thus suggested toolset includes next components: SpaceWire BFM, test generators and test controllers and wrappers for all levels of BFM, model of external memory and interface components for control of UUT regimes.

3 CONCLUSION

Suggested toolset could be user for test and verification of SpaceWire controllers IP-blocks that support different levels of protocol, of devices includes support some levels of protocol, for example, SpaceWire switches, special processors includes SpaceWire controllers.

4 REFERENCES

1. ECSS E50 12A, “SpaceWire. Links, nodes, routers and networks”, 24 January 2003

DESIGNING SPACE CUBE 2 WITH ELEGANT FRAMEWORK

Session: SpaceWire test and verification

Short Paper

Hiroki Hihara

NEC TOSHIBA Space Systems, Ltd., 10, Nisshin-cho 1-chome, Fuchu, Tokyo, Japan

Shuichi Moriyama, and Tatsuya Takezawa

NEC Soft, Ltd., 2-22 Wakaba-cho, Kashiwazaki, Niigata, Japan

Yuji Nishihara

JAXA's Engineering Digital Innovation Center, Japan Aerospace Exploration Agency (JAXA), Tsukuba Space Center, Sengen 2-1-1, Tsukuba, Ibaraki, 305-8505, Japan

Masaharu Nomachi

Laboratory of Nuclear Studies, Graduate School of Science, Osaka University,

1-1 Machikaneyama, Toyonaka, Osaka 560-0043

Tadayuki Takahashi, and Takeshi Takashima

Department of High Energy Astrophysics, Institute of Space and Astronautical Science (ISAS), Japan Aerospace Exploration Agency (JAXA), 3-1-1 Yoshinodai, Sagamihara,Kanagawa 229-8510, Japan

E-mail: hihara.hiroki@ntspace.jp, moriyama@mxp.nes.nec.co.jp, takezawa@mxa.nes.nec.co.jp, nishihara.yuuji@jaxa.jp, nomachi@lns.sci.osaka-u.ac.jp, takahasi@astro.isas.jaxa.jp, ttakeshi@stp.isas.jaxa.jp

ABSTRACT

Faced with growing demands for function and performance of satellite onboard equipments with maintaining reliability, Japan Aerospace Exploration Agency (JAXA) has developed ELEGANT (Electric Design Guidance Tool for Space Use), a complete C-language based environment for electronic system-level (ESL) design of space and satellite electronics. Space Cube 2 is not only the first SpaceWire based satellite onboard system controller in Japan but also the first real application of ELEGANT framework. ELEGANT provides a seamless tool chain for modelling verification and synthesis from top-level specification down to embedded Hardware/Software implementation. Performance trade-off associated with candidate system architectures has been accomplished early in the system design phase.

1 INTRODUCTION

Space Cube 2 (Figure 1) is the first SpaceWire based satellite onboard system controller in Japan [1] as well as the first real application of ELEGANT (Electric Design Guidance Tool for Space Use), which is developed by Japan Aerospace

Exploration Agency (JAXA) [2]. ELEGANT has the capability of top down design using hardware and software collaborating design methodology.

Figure 1(a): Space Cube 2 (Flight Model)

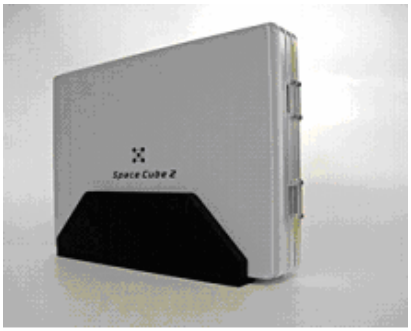


Figure 1(b): Space Cube 2/C (Development Platform)

The system level specification is modelled as behaviours and channels, which is described in SpecC language, as standardized by the SpecC Technology Open Consortium (STOC) [3], at the first step in ELEGANT design flow. The specification model can be verified by simulation subsystem within the specification model simulator. After the specification is verified, designers can evaluate and select among candidate architectures assisted by the architecture exploration tools. Evaluation based on processing speed, software execution steps and hardware scale in accordance with several hardware-software partitioning is possible. In consequence, designers can select the most suitable architecture for their project demands. Once the architecture is fixed, the behaviour description written in SpecC is synthesized into HDL (hardware description language) directly with behaviour synthesis tool.

2 ELEGANT

2.1 ELEGANT ENVIRONMENT

ELEGANT (Figure 2) provides an environment for top-down electronic system-level (ESL) design. Under JAXA's guidance, ELEGANT is a world-wide joint R&D project involving several partners, combining different point tools.

2.2 SPECIFY-EXPLORE-REFINE (SER)

The SER component provides system level design space exploration and model refinement. Starting from an abstraction specification of the desired functionality written in SpecC, SER allows interactive platform definition and specification mapping. SER then automatically implements the specification on the platform and generates various transaction-level models (TLMs) of the design. The SER engine has been developed by the Center for Embedded Computer Systems (CECS) at University of California, Irvine as a derivate of CECS' original SpecC-based System-On-Chip Environment (SCE) [4], where SER has been adapted to JAXA requirements and databases have been filled with necessary components for space and satellite applications.

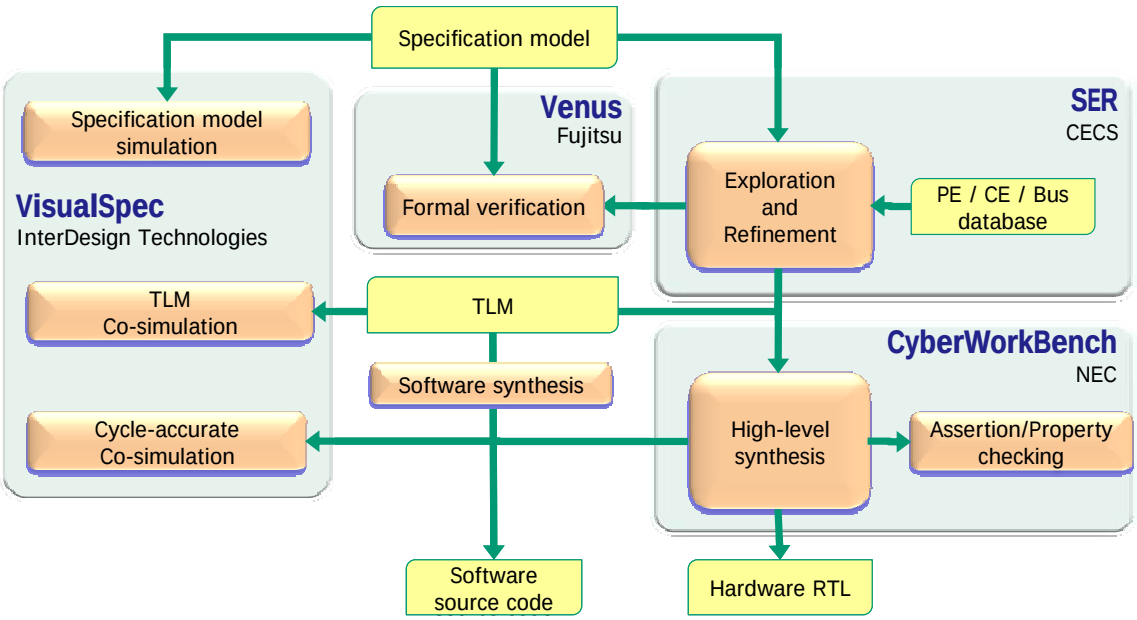


Figure 2: The ELEGANT environment

2.3 VISUALSPEC

In ELEGANT, models are captured and simulated using VisualSpec, a SpecC modeling and simulation environment supplied by InterDesign Technologies [6]. VisualSpec provides visualization and debugging support and optional integration of third-party instruction-set simulators (ISS) into the SpecC simulation backplane. VisualSpec supports profiling and estimation capabilities for model analysis and design quality feedback, including software execution time estimation and timing back-annotation of TLM software models using so-called FastVeri technology.

2.4 CYBERWORKBENCH

For synthesis of hardware components, ELEGANT includes the CyberWorkBench high-level, C-to-RTL synthesis tool originally developed at NEC [5].

As part of the ELEGANT project, Cyber has been adapted to accept SpecC input directly from the SER-generated models. Furthermore, it was extended to generate cycle-accurate (CA) SpecC models of the RTL output for co-simulation of the synthesized hardware with the rest of the system.

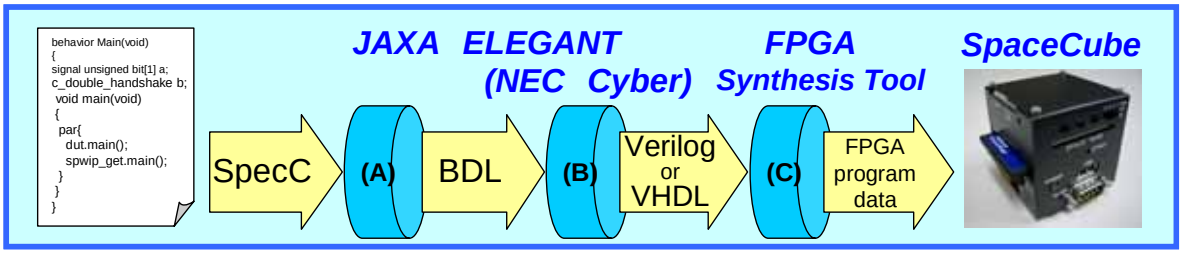
By designing hardware in C, software and hardware designers can communicate in the same language, which makes System LSI design more effective.

2.5 VENUS

ELEGANT contains a formal verification component, Venus, developed by Fujitsu Labs of America (FLA) and Tokyo University. Venus supports equivalence checking between different models in the ELEGANT design flow to guarantee correctness and model equivalence throughout the design process.

3 DESIGNING SPACE CUBE 2 WITH ELEGANT

JAXA initiated a project to evaluate ELEGANT design flow with the development of Space Cube 2. Using SER tools, hardware/software partitioning and exploration of the target architectures was performed. In the process, exploration was supported and driven by quantitative estimation (using VisualSpec estimation and profiling capabilities) to help select an appropriate implementation. Once a partitioning is selected, deriving FPGA data is a straight-forward process using CyberWorkBench as shown in figure 3.



(A) Program Conversion from SpecC to BDL (Behavioral Description Language)
(B) Behavioral Synthesis
(C) Logic Synthesis

Figure 3: FPGA implementation of RMAP from SpecC specification

The capacity of synthesized hardware on FPGAs or ASICs are slightly larger than ones which are logically synthesized from hand-coded HDL, and compact enough for space use.

4 REFERENCES

1. Tadayuki Takahashi, Takeshi Takashima, Satoshi Kuboyama, Masaharu Nomachi, Yasumasa Kasaba, Takayuki Tohma, Hiroki Hihara, Shuichi Moriyama, Toru Tamura, “SpaceCube 2 -- An Onboard Computer Based on SpaceCube Architecture”, International SpaceWire Conference 2007, 17-19 September 2007, p.65-68..

2. A. Gerstlauer, J. Peng, D. Shin, D. Gajski, A. Nakamura, D. Araki, Y. Nishihara, “Specify-Explore-Refine (SER): From Specification To Implementation”, Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE, 8-13 June 2008, p.586-591.

3. R. Dömer, A. Gerstlauer, and D. Gajski. SpecC Language Reference Manual, Version 2.0. SpecC Technology Open Consortium, http://www.specc.org, December 2002.

4. R. Dömer, A. Gerstlauer, J. Peng, D. Shin, L. Cai, H. Yu, S. Abdi, and D. Gajski. System-on-Chip Environment: A SpecC-based Framework for Heterogeneous MPSoC Design. EURASIP Journal on Embedded Systems, 2008.

5. NEC System Technologis, Ltd. CyberWorkBench. http://www.necst.co.jp/product /cwb

6. InterDesign Technologies, Inc. VisualSpec. http://www.interdesigntech.co.jp.

LESSONS LEARNED FROM IMPLEMENTING NON STANDARD SPACEWIRE CABLING FOR TACSAT-4

Session: SpaceWire Test and Verification

Short Paper

Derek Schierlmann, Eric Rossland and Paul Jaffe

Naval Center for Space Technology, Naval Research Laboratory Code 8243, 4555 Overlook Ave SW, Washington, DC 20375, USA

E-mail: derek.schierlmann@nrl.navy.mil, erossland@space.nrl.navy.mil, paul.jaffe@nrl.navy.mil

ABSTRACT

The rapid integration, launch, and deployment of satellites in response to emerging needs has been termed "Operationally Responsive Space" (ORS). One vision of ORS calls for the positioning in a depot of interchangeable satellite payloads and spacecraft buses with a common interface. Upon direction to deploy a particular mission, the appropriate payload would be selected and integrated with a bus, and the space vehicle would be launched. To support such a system, standardized hardware and software interfaces are needed between the payload and bus. For the development of ORS Bus Standards, the SpaceWire standard (ECSS-E-50-12A) has been specified as part of such a payload-bus interface for high rate data. The TacSat-4 satellite, part of the USDOD TacSat experiment series, is intended as a combination of a prototype ORS Standardized Bus for small satellite national security missions and an example payload. This implementation includes an instance of the SpaceWire interface called out in the ORS Payload Developer's Guide. The need for non-standard SpaceWire connectors has been established in previous studies. Such deviations are justified to get more performance or better human factors engineering. When these deviations from a standard are undertaken, extra effort is required to validate the implementation. Often these efforts result in valuable lessons learned. Investigation and testing described in this paper details our recent efforts, at the Naval Center for Space Technology, to design and qualify for flight on TacSat-4 a non-standard SpaceWire connector implementation. This paper will also cover performance, details on qualification, and lessons learned from environmental testing performed during TacSat-4 flight qualification.

INTRODUCTION

The SpaceWire link on TacSat-4 connects the Payload Data Handler (PDH) module in the Command and Data Electronics (CDE) on the bus side with the Universal Interface Electronics (UIE) on the payload side. CCSDS (Consultative Committee for Space Data Systems) space packets are used for the higher level protocol as dictated by the document "ORS Standard Data Interfaces: Bus to Payload, Bus to Ground".

Because of the depot concept imbedded in the ORS standards, the SpaceWire link deviated from the physical layer called out in ECSS-E-50-12A (now ECSS-ST-50-

12C). As mentioned in previous papers [Schierlmann, Jaffe] the ORS standards require that an ORS bus and payload are capable of being mated in a depot facility by minimally trained personnel without specialized tools. Thus the point to point cable called out in Sections 5.3 – 5.4 of the SpaceWire standard is not ideal.

Section 5.4 of 12C calls out the use of a single cable assembly of no more than 10m joined by two identical connectors. TacSat-4 needed to implement the cable assembly as three separate cables with a total of six connectors. This arrangement is shown in Figure 1.

In addition to the three segment flight cable, other cables were fabricated to accommodate integration and test activities. The three segment 10m cable used for I&T testing consisted of two bulkhead breaks: one to provide for passing through the thermal vacuum chamber wall, and another for passing through the turn on panel of the bus. Comm-X payload testing required a longer three segment 18.5m cable for payload I&T activities, especially electromagnetic interference (EMI) testing. The ORS bus and payload teams successfully tested SpaceWire across these cabling configurations.

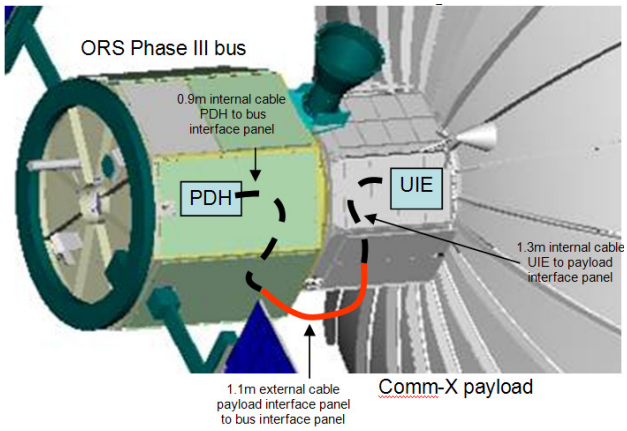


Figure 1: TacSat-4 SpaceWire cabling configuration

Section 5.3 of 12C specified that SpaceWire cables are only to use a 9 position micro-D connector. However, in order to meet depot handling requirements, TacSat-4 chose to use 38999 Series IV connectors (D38999/40FB35SN, D38999/46FB35PN) at the bus/payload interface panels. The choice and validation of these connectors has been documented previously [Schierlmann]. TVAC chamber penetrations were handled with standard hermetic circular connectors from Deutsch (DS07-37S-081, 13084-37S-5020).

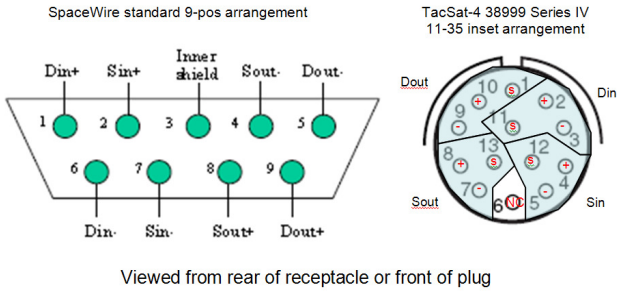


Figure 2: TacSat-4 pin assignments vs. standard Spacewire

The final deviation from ECSS-ST-50-12C was with regard to cable construction as called out in section 5.2. Previous work [Allen, Mueller] has shown that 26 AWG cable outperforms the cable called out in the specification. TacSat-4 chose to use 26 AWG SpaceWire cable manufactured by W.L. Gore & Associates GmbH.

DATA

SpaceWire at the CDE box level and system level was simulated using a PMC spacewire card purchased from Dynamic Engineering. This particular card was able to interface directly onto the VME Power 7E card which was already located in the SES (space environment simulator) chassis. This ability, though at first seemed most beneficial for saving space in the chassis, proved to make interfacing the cables to the PMC card more difficult. The small work area made it difficult to physically connect the cables to the card itself. A couple of mating instances resulted in cable wire to pin connections separating and having to rework the cable. It is important to note that this issue only occurred with the microD solder cup type connectors and did not occur with the potted flying lead connectors.

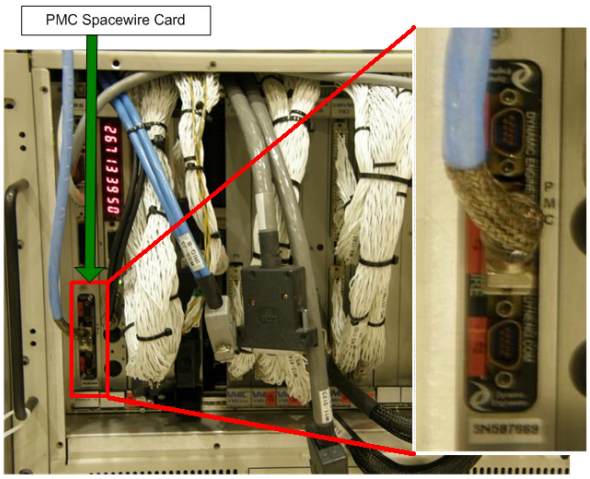


Figure 3: TacSat-4 SES chassis w/ PMC Spacewire Card

Developing GSW (ground software) for utilizing the new PMC card to test the PDH board proved to have a significant learning curve. Knowing the SpaceWire protocol was helpful, but until data was actually flowing across the interface it was difficult to predict what the data would look like and how the PMC card would behave. A breakout box and logic analyzer proved to be critical tools to help understand and troubleshoot the interface. Loopback connectors also proved very useful for stand alone testing of both the PDH and PMC cards.

During ORS bus system level testing, the PMC card simulated the Comm-X payload SpaceWire interface and a SpaceWire test interface to accommodate for testing both channels from the PDH card. The tests performed across the SpaceWire link were run at 25Mbps. Upon completion of the Comm-X payload the ORS bus and Comm-X payload will be integrated to form the TacSat-4 space vehicle. Full space vehicle system level testing will be performed across the SpaceWire link at that point.

For SECCHI, SpaceWire cables of DVI heritage were used. When they were modified for environmental test, we found these cables difficult to work with and prone to breakage. The same was found to be true of the DVI heritage cables used during initial TacSat-4 studies. No such problems were found while using the 26 AWG Gore & Associates SpaceWire cable and the potted flying lead microD connectors used on TacSat-4.

With the addition of the TVAC chamber wall, and to a lesser extent at the bus and payload interfaces, ambiguity arose as to where the out-to-in twisting was to be done. A suggestion from the TacSat-4 bus team was to twist once in each cable, so that an odd number of cables resulted in proper in-to-out assignment. Preliminary designs for the TacSat-4 SpaceWire cable dedicated a pin to carry the outer shield but since the outer shield is chassis ground, this was unnecessary and ill-advised.

Initial qualification of the TacSat-4 implementation of the SpaceWire standard was taken from a previous study [Schierlmann]. This paper extends the work based on feedback received. Eye diagrams were taken using a digital serial analyzer scope (DSA70604). The scope was unavailable for flight cable qualification, but the images were useful as a quick validation of the I&T fabricated cable. The DSA was also helpful in diagnosing a problem with the SpaceWire test board.

The original study used potentially bandwidth limiting 400MHz differential probes when rules of thumb suggest that probes with a bandwidth of at least 1GHz should be used. Further testing was performed with 1GHz probes and no difference was found between results taken with the 400MHz and 1GHz probes.

Table 1: 4-wire (above ground) source - victim cross talk model [Allen]

			Victim						
Connector	Source->Victim	Pins	h (mils)	s (mils)	D (mils)	D/s	D/h	dB	Test group
uD9	Dout->Sout	9/5->8/4	70	50	50	1.00	0.71	-24	2
uD9	Dout->Sout	9/5->8/4	70	50	50	1.00	0.71	-24	2
uD9	Sout->Sin	8/4->2/7	70	50	75	1.50	1.07	-24	2
uD9	Dout->Din	9/5->1/6	70	50	175	3.50	2.49	-40	1
uD9	Dout->Din	9/5->1/6	70	50	175	3.50	2.49	-40	2
11_35P	Dout->Din	10/9->2/3	74	90	223	2.47	3.00		N/A
DB9	Dout->Din	9/5->1/6	98	124	378	3.04	3.86	-30	1
HDD15	Dout->Din	1/2->14/15	76	90	312	3.46	4.10	-47	1

TacSat-4 relied on the crosstalk, jitter and skew analysis performed in previous studies, however, tailored analyses should be used when validating a new interconnect. In the absence of sufficient time or equipment to a complete cross-talk study, one can apply first principles in order to extend to existing data. TacSat-4 chose to look at data in the JWST Connector Choice Study [Allen] and fit the data to a simple cross talk geometry as described in texts[Paul, Johnson 1993]. With some simplification, one finds –as expected- that the cross talk noise shows a strong correlation to the distance (from source to victim) divided by distance to ground (D/h). Given this correlation, one could expect the 11-35 connector chosen by TacSat-4 to perform similarly to the High density D connector investigated. This cross-talk performance is at least as good as the SpaceWire micro-D.

The previous paper incorrectly states that the tests were not run at the full speed of the driver (200Mb/s). However this came from a misunderstanding of the results returned by the SpaceWire driver. A conversation and a quick check of the scope traces confirms that the tests were indeed run at full 200Mb/s speed. Also, the previous paper incorrectly referred to a 38999 Series II with a 10-35 insert arrangement when the connector is really a 38999 Series IV with an 11-35 insert arrangement.

CONCLUSIONS

The 10m and 18.5m cables fabricated for environmental test performed well. The extra length helped to dampen ringing induced by the discontinuities of two inline connectors (bus and chamber wall).

After qualifying SpaceWire cables on two occasions, we still see opportunities for improvement with the test board. Our attempts resulted in noticeable reflection in the signal. One effective option to capture waveforms without an impedance mismatch was to solder to the internals of the SpaceWire brick from Dundee. Given the features and cost of the brick, this approach was risky. In the future we may use a modified DESWBO from Dynamic Engineering for examining waveforms.

The bandwidth of a TDR is 20-30GHz, at which frequencies the padstack, stack-up, and foot prints become critically important. The TDR test board had an excessive discontinuity because the antipad around the SMA connector was too large [Bakel]. This discontinuity was large enough to prevent Iconnect from converging to an impedance solution. When commissioning test boards, it is important to know your frequency of interest. For this study was related to TDR bandwidth (30GHz) and not SpaceWire knee frequency (<1GHz) [Johnson, 1993]. Ensure that your layout engineer is familiar with designing to the frequency of interest. For future testing, we may try the Gore test board described in [Allen].

REFERENCES

Connectors, Electrical, Rectangular, Microminiature, Polarized Shell, General Specification for (w/Amendment 1), MIL-DTL-83513. Revision: F, Dated: 30 June 2008. Available: http://www.dscc.dla.mil/Programs/MilSpec/listdocs.asp?BasicDoc=MIL-DTL-83513

Connectors, Electric, Rectangular, Nonenvironmental, Miniature, Polarized Shell, Rack and Panel, General Specification for, MIL-DTL-24308. Revision: F, Dated: 25 June 2007. Available: http://www.dscc.dla.mil/Programs/MilSpec/ListDocs.asp?BasicDoc=MIL-DTL-24308

C. R. Paul, *Introduction to Electromagnetic Compatibility*, John Wiley & Sons, 1992.

D. A. Powner, C. Cha, N. Doherty, N. Glover, K. Malhotra, C. Phillips, K. Richey, "Geostationary Operational Environmental Satellites: Steps Remain in Incorporating Lesson Learned from Other Satellite Programs," US GAO, Washington, DC, Rep. GAO-06-993, Sep. 2006. Available: http://www.gao.gov/new.items/d06993.pdf

D. Schierlmann, P. Jaffe, "SpaceWire Cabling in an Operationally Responsive Space Environment," Proceedings of the International SpaceWire Conference 2007.

H. Johnson, G. Martin, *High-Speed Signal Propagation: Advanced Black Magic*, New Jersey: Prentice Hall PTR, 2003

H. Johnson, G. Martin, *High-Speed Digital Design: A Handbook of Black Magic*, New Jersey: Prentice Hall PTR, 1993

Insert Arrangements for MIL-DTL-38999, MIL-DTL-27599 and MIL-C-29600 Series A Electrical Circular Connectors Revision: B, Dated: 20 July 2007Available: http://www.dscc.dla.mil/Programs/MilSpec/ListDocs.asp?BasicDoc=MIL-STD-1560

J. Bakel, "Diff Pair Routing," Unpublished, April 2004.

J. Mueller, "Design Challenges Of An Advanced Spacewire Assembly For High Speed Inter-Unit Data Link," 2006 MAPLD International Conference, Washington, DC, September 2006.

Operationally Responsive Space (ORS) General Bus Standard (GBS), ORSBS-002/NCST-S-SB001 Revision 3, Jan. 2008. Available: http://projects.nrl.navy.mil/busstandards/

Operationally Responsive Space (ORS) Payload Developer's Guide (PDG), ORSBS-003/NCST-IDS-SB001 Revision 3, Jan. 2008. Available: http://projects.nrl.navy.mil/busstandards/

ORS Standard Data Interfaces: Bus to Payload, Bus to Ground, ORSBS-004/NCST-ICD-SB008 Revision 2, Dec. 2007. Available: http://projects.nrl.navy.mil/busstandards/

P. Jaffe, G. Clifford, J. Summers, "SpaceWire for Operationally Responsive Space as part of TacSat-4," Proceedings of the International SpaceWire Conference 2007.

S. Allen, "SpaceWire Physical Layer Issues," 2006 MAPLD International Conference, Washington, DC, September 2006.

S. Allen, "SpaceWire/Synchronization Connector Choice," Unpublished, October 2004.

IMPLEMENTATION OF A VERY LOW COST PORTABLE SPACEWIRE MONITOR AND DEBUGGER

Session: SpaceWire test and verification

Short Paper

Paul Jaffe

Naval Research Laboratory Code 8243, 4555 Overlook Ave SW, Washington, DC 20375, USA

E-mail: paul.jaffe@nrl.navy.mil

Keith Leisses

Dynamic Engineering 150 DuBois St. #C Santa Cruz, CA 95060, USA

ABSTRACT

The advent of a wide range of USB peripherals for personal computers has altered the test and measurement landscape. In particular, easily portable and inexpensive USB logic analyzers, such as the DigiView™ DV1-100, have enabled a host of industry-specific test applications. Such a logic analyzer used in conjunction with the Dynamic Engineering SpaceWire BreakOut (DESWBO, http://www.dyneng.com/deswbo.html) and an inexpensive notebook computer allows users to implement a portable and capable SpaceWire troubleshooting tool. Support software included with USB logic analyzers typically provides a myriad of display options, large data capture buffers, specialized trigger functions, and a variety of data export alternatives, including waveform images and ASCII files. A logic analyzer with a suitable sampling rate can be selected depending on the rate of the SpaceWire link of interest. Configuration templates for specific test scenarios are easily stored. The DESWBO fits between two nodes of a SpaceWire link using standard 9-pin MDM connectors. Data, strobe, and recovered clock signals are made available as pin outputs that may be monitored by the logic analyzer. Additionally, pins corresponding to SpaceWire events and conditions allow monitoring and triggering by the logic analyzer. These include the reception of nulls, flow control tokens, normal and error end-of-packets, and timecodes. Signals for monitoring credit, disconnect, escape, and parity errors are provided. Eight data bits and six credit count bits may also be monitored. All components of the system may be stored in a laptop computer bag and deployed at a moment's notice to debug a SpaceWire problem.

INTRODUCTION

One of the common objections raised to the use of SpaceWire is that it is not as widespread in its terrestrial use as other interfaces that have been proposed for space. In particular, MIL-STD 1553, RS-422, IEEE-1394 variants (including FireWire), Ethernet, I2C, and USB have been posed as more palatable alternatives.[1] Though there are many factors that go into interface selection, not the least of which include data transfer capacity, prior spaceflight heritage, power consumption, and development complexity, often price and availability of test equipment are cited as a salient variables to be weighed.

Historically, the costs of interface test equipment are generally dwarfed by other space program cost elements, such as flight hardware procurement and staffing. In some respects this is changing, as the prevalence of high school and university student satellite projects join ranks with small aerospace companies who may be developing only one component or a small experiment. These entities may be severely cost-constrained, and perhaps do not have the millions of dollars in resources often associated with spacecraft development. They frequently depend on uncompensated student labor and donated

launch opportunities, and have very limited if any development budgets. For some, even an additional burden of several thousand dollars may prove insurmountable.

Students will often tend towards tools and technologies with which they are familiar once they enter the workforce. They typically will have spent time developing expertise and experience in specific areas with specialized tools, and will have a penchant towards availing themselves of this knowledge where possible. Students in aerospace will be more likely to employ SpaceWire if they have had an earlier exposure to it, which can be promoted with very low cost debugging tools.

Approximate cost ranges of ground test equipment for various interfaces - prices $USD				
Interface	**Hardware needed**	**Software needed**	**2008 ~cost range**	**Suppliers**
MIL-STD-1553	Interface card	Incl. with hardware	$500-$2,000	SBS, Paravant, others
RS-422	USB or RS232 to 422 conv	Free d/l: Terminal software	$100-$300	Black Box, B&B, others
I2C	USB-I2C converter	Incl. with hardware	$250-$750	MCC, Total Phase, others
USB	(included with PC)	Free d/l: USBET, USBTester	$0	usb.org, Jungo, others
IEEE-1394	(included with many PCs)	Free d/l: libdc1394, VHPD1394	$0-$1500	sourceforge.net, Thesycon
Ethernet	(Included with PC)	Free d/l: tcpdump, Wireshark	$0	tcpdump.org, wireshark.org
SpaceWire	Interface card or converter	Incl. with hardware	$3,500-$12,000+	Star-Dundee, 4Links, others
	DESWBO and LA	Incl. with USB logic analyzer	$900+LA*	Dynamic Engineering

*appropriate USB logic analyzers (LA) range from $149 to $1500+

- Cost of time to become familiar with hardware and software tools, which is considerable in some case, is not included.

- The level of capability provided by the options in the table is highly variable

Figure 1. Cost comparison of interface test equipment

USB LOGIC ANALYZERS (LA)

Logic analyzers have gone from being cumbersome, complex, expensive pieces of equipment that cost tens of thousands of dollars to small, easy-to-use devices available at near commodity pricing. This has been enabled in large part by two advances: the ubiquity of the personal computer and the comparatively recent rise of the USB interface. By 2008, several billion USB interface devices had entered the market[2], and USB had become the dominant interface in the personal computer realm. At the time of this writing, over a dozen different types of USB logic analyzers were available for less than $1,000 US, offering amenities such as integrated oscilloscopes, pattern generators, as many as 34 channels per device, and up to oscillator frequencies of 1 GHz.[3]

For the system described in this paper, a DigiView™ DV1-100 USB logic analyzer was selected, largely because it was on hand, rather than for any specific performance criteria. The logic analyzer features 18 Channels @100 MHz and uses the USB interface for both power and data. Built-in hardware compression reduces resolution vs. collection depth tradeoffs with a compression ratio of up to 256,000:1 depending on signal activity. It can be configured to trigger on rising edges, falling edges, or transition of any channel. Software configuration through a graphical user interface allows for sophisticated pattern triggers employing multiple channels and "don't care" qualifiers. A wealth of display and export software features allow for storage and analysis of acquired data.[4] The DV1-100 is available for about $400 US, and has countless applications beyond use with the DESWBO for SpaceWire debugging.

THE DYNAMIC ENGINEERING SPACEWIRE BREAKOUT (DESWBO)[5]

The DESWBO monitors the signals of a single SpaceWire link and is packaged as a printed circuit board mounted on a compact enclosure with many mounting rail options. The board has two 9-pin MDM connectors to facilitate in-line monitoring of both sides of a SpaceWire link. The DESWBO is designed to detect and decode bit sequences on the SpaceWire link. Signals are issued indicating what types of characters are passing between nodes as well as the contents of data and timecode characters. A running count of flow control credits for each node is calculated by the DESWBO by monitoring FCTs and N-character occurrences.

The low voltage differential signalling (LVDS) signals from each node are buffered to the opposite node and monitored by a field programmable gate array (FPGA). The FPGA contains the equivalent of two SpaceWire receiver modules that decode the SpaceWire signals, and extract the various data and control characters as shown in Figure 2. The DESWBO has been tested with a hardware that has the ability to insert errors under software control. Normal and error conditions are tested on each unit as part of the acceptance test procedure.

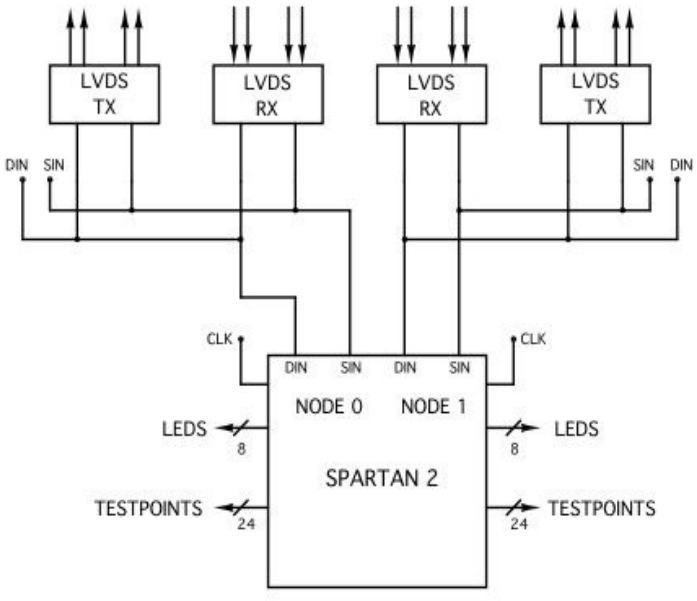


Figure 2. DESWBO Block Diagram

The end-of-packet, got-data and got-FCT signals from each node are connected to pulse capture and duration extension circuits to drive three green LEDs. The error-end-of-packet, parity error, escape error, credit error and disconnect error signals for each node drive five red LEDs. These sixteen LEDs give a quick status of link activity and health.

In addition, there are 24 test points for each node that are driven by the real-time signals from the SpaceWire character receivers, two test points for each node that are connected to the DIN and SIN signals from the LVDS receivers, and one test point for each node that is driven by the recovered clock from the SpaceWire bit receivers. The 24 test point signals consist of eight data bits, six credit count bits, Parallel data strobe, NULL received, FCT received, Timecode received, End-of-Packet, Error End-of-Packet, Credit error, Disconnect error, Escape error and Parity error. There are also three ground test points per node to facilitate probe grounding.

The DESWBO is powered by an included external 5-volt supply with on-board regulators for the various operating voltages required.

USE OF THE SYSTEM

The logic analyzer software configuration allows the user to tailor the interface and signal layout to the task at hand. Specific configurations for SpaceWire troubleshooting can be created and saved for

Figure 3. Large time scale capture

ongoing use. An array of cursors and zoom levels allows for fast discernment of what is happening on a given SpaceWire link. At larger timescales, flag signals like NULL-received and FCT received give a good overall picture of link activity as in Figure 3. For inspecting activity on the actual data and strobe signals, smaller timescales can be used as in Figure 4.

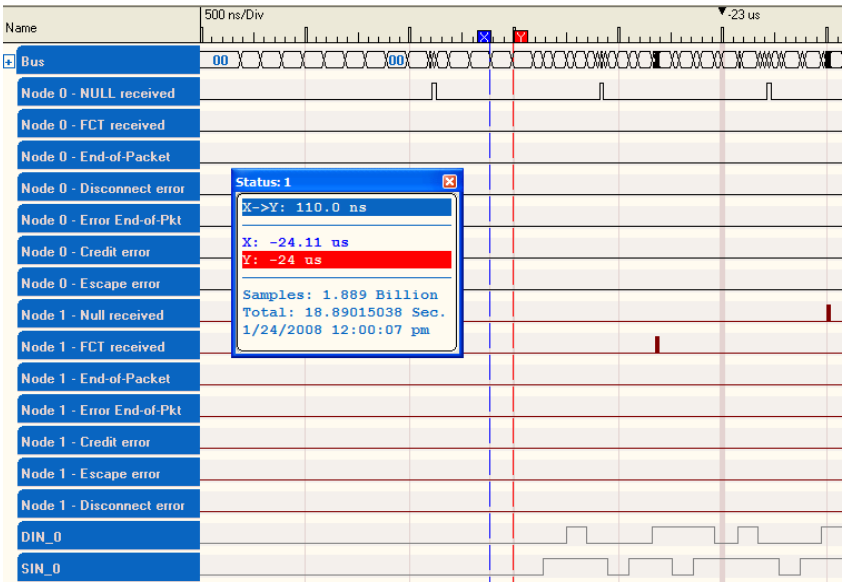


Figure 4. Small time scale capture

This combination was used successfully to debug and gain insight into SpaceWire links on TacSat-4, PnPSat, Herschel II, and other satellite programs. Its portability proved a boon and provoked positive feedback from many colleagues.

The computer, DESWBO, logic analyzer, and necessary cables easily fit in a laptop computer bag. This allows quick deployment of a useful test capability on the other side of the laboratory or the other side of the world.

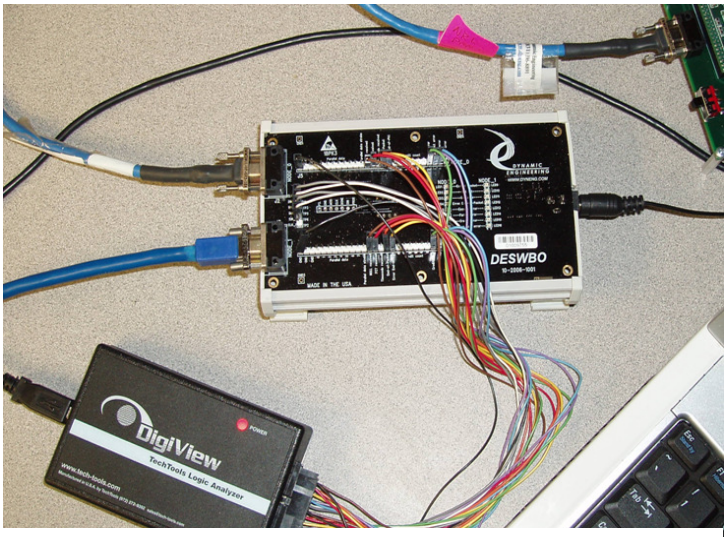


Figure 5. Logic analyzer and DESWBO

CONCLUSION

There is a clear need for very inexpensive SpaceWire test and debugging equipment. Though not as fully featured as other alternatives, the combination of a notebook computer, USB logic analyzer, and Dynamic Engineering's DESWBO provides an easily portable and highly effective troubleshooting capability.

---

[1] Jaffe, P., Clifford, G., Summers, J., "SpaceWire for Operationally Responsive Space as part of TacSat-4", International SpaceWire Conference, Dundee, Scotland, September, 2007.

[2] http://www.jungo.com/st/usb_market.html, retrieved 2008-09-27

[3] http://www.techtravels.org/tech/logicanalyzer.html, retrieved 2008-09-27

[4] "DigiView User's Guide", TechTools 2007.

[5] http://www.dyneng.com/deswbo.html, retrieved 2008-09-27

Missions & Applications 1

Thursday 6 November
14:20 – 15:00

DATA ACQUISITION SYSTEM OF THE POGOLITE BALLOON EXPERIMENT

Session: Missions and Applications

Long Paper

H. Takahashi, M. Matsuoka, Y. Umeki, H. Yoshida, T. Tanaka, T. Mizuno, Y. Fukazawa

Hiroshima University, Higashi-Hiroshima 739–8526, Japan

T. Kamae, G. Madejski, H. Tajima

SLAC and KIPAC, Menlo Park, California 94025, USA

M. Kiss, W. Klamra, S. Larsson, C. Marini Bettolo, M. Pearce, F. Ryde, S. Rydström

Royal Institute of Technology, SE–106 91 Stockholm, Sweden

K. Kurita, Y. Kanai, M. Arimoto, M. Ueno, J. Kataoka, N. Kawai

Tokyo Institute of Technology, Meguro-ku, Tokyo 152–8550, Japan

M. Axelsson, L. Hjalmarsdotter

Stockholm University, SE–106 91 Stockholm, Sweden

G. Bogaert

Ecole Polytechnique, 91128 Palaiseau Cedex, France

S. Gunji

Yamagata University, Yamagata 990–8560, Japan

T. Takahashi

JAXA/ISAS, Sagamihara 229–8510, Japan

G. Varner

University of Hawaii, Honolulu, Hawaii 96822, USA

T. Yuasa

University of Tokyo, Bunkyo-ku, Tokyo 113–0033, Japan

E-mail: hirotaka@hep01.hepl.hiroshima-u.ac.jp, matsuoka@hep01.hepl.hiroshima-u.ac.jp, umeki@hep01.hepl.hiroshima-u.ac.jp, hyoshida@hep01.hepl.hiroshima-u.ac.jp, tanaka@hep01.hepl.hiroshima-u.ac.jp, mizuno@hep01.hepl.hiroshima-u.ac.jp, fukazawa@hep01.hepl.hiroshima-u.ac.jp, kamae@slac.stanford.edu, madejski@slac.stanford.edu, htajima@slac.stanford.edu, mozsi@kth.se, klamra@particle.kth.se, stefan@astro.su.se, cecilia@particle.kth.se, pearce@particle.kth.se, Stefan@particle.kth.se, kurita.k.aa@m.titech.ac.jp, kanai@hp.phys.titech.ac.jp, arimoto@hp.phys.titech.ac.jp, masaru@hp.phys.titech.ac.jp, kataoka@hp.phys.titech.ac.jp, nkawai@hp.phys.titech.ac.jp, magnusa@astro.su.se, nea@astro.su.se, felix@astro.su.se, bogaert@poly.in2p3.fr, gunji@sci.kj.yamagata-u.ac.jp, takahasi@astro.isas.jaxa.jp, varner@phys.hawaii.edu, yuasa@amalthea.phys.s.u-tokyo.ac.jp

ABSTRACT

The Polarized Gamma-ray Observer, PoGOLite, is a balloon experiment with the capability of detecting 10% polarization from a 200 mCrab celestial object in the energy-range 25–80 keV. During a beam test at KEK-PF in February 2008, 20 detector units were assembled, and a 50 keV X-ray beam with a polarization degree of ~90% was irradiated at the center unit. Signals from all 20 units were fed into flight-version electronics consisting of six circuit boards (four waveform digitizer boards, one digital I/O board and one router board) and one microprocessor (SpaceCube), which communicate using a SpaceWire interface. One digitizer board, which can associate up to 8 PDCs, outputs a trigger signal. The digital I/O board handles the trigger and returns a data acquisition request if there is no veto signal (upper or pulse-shape discriminators) from any detector unit. This data acquisition system worked well, and the modulation factor was successfully measured to be ~33%. These results confirmed the capabilities of the data-acquisition system for a "pathfinder" flight planned in 2010.

1. INTRODUCTION

The Polarized Gamma-ray Observer (PoGOLite) is a balloon-borne astronomical soft gamma-ray polarimeter optimized for point-like sources [1]. It measures polarization in the energy range 25–80 keV from sources with flux levels as low as 200 mCrab by using the azimuthal angle anisotropy of Compton-scattered photons. As shown in Figure 1 and 2, it measures coincident Compton scattering and photoabsorption for gamma-ray events in an array of 217 well-type phoswich detector cells (PDCs), each comprising a fast plastic scintillator (detection part), a slow plastic scintillator collimator and a BGO crystal (shield), which are viewed by a single photo-multiplier tube (PMT). The 217 PDCs are surrounded by a side anti-coincidence shield (SAS) made of 54 segments of BGO crystals. The instrument is currently under construction and an engineering flight of a 61-unit ‘‘pathfinder" instrument is planned for 2010 from the Esrange facility in the North of Sweden.

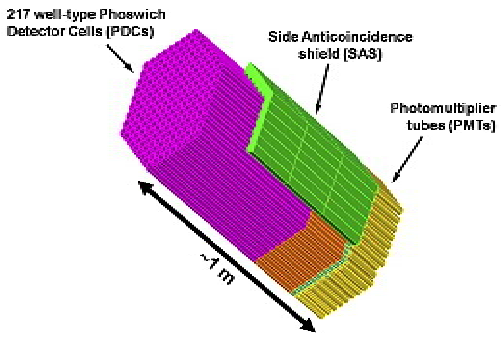


Figure 1. Schematic view of the PoGOLite detector array, consisting of 217 PDCs and 54 SAS units, giving 271 PMT output signals in total.

The in-flight background due to cosmic-rays, charged particles, atmospheric gamma-rays and neutrons is extremely high, typically a few hundred Hz in each unit. The data acquisition (DAQ) system of PoGOLite is required to handle weak signals from astrophysical objects (200mCrab, 2–4 counts per second in the energy range of the instrument) under such a severe environment.

In section 2, we describe the PoGOLite DAQ system. Section 3 details the beam test results obtained at KEK-PF in February 2008, which demonstrate the performance of the detector units and the flight-version DAQ system with the SpaceWire interface.

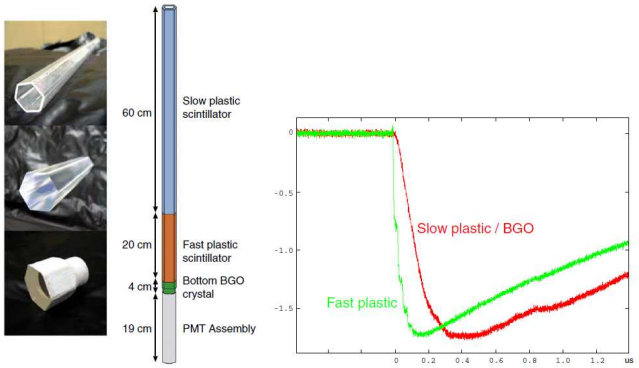


Figure 2. (Left) Structure of the PDC. Fast, slow and BGO scintillators are viewed by one PMT. (Right) An example of waveforms of the PoGOLite CSA output. The decay time of the fast scintillator (τ~2 ns) is distinguishable from that of the slow scintillator/BGO (τ~300 ns) (Pulse- Shape Discrimination).

2. Data Acquisition System of PoGOLite

2.1. Gamma-ray Measurement by PDC Units

The PoGOLite DAQ system consists of six parts: front-end electronics, waveform digitizer, trigger logic, global event logic, microprocessor and storage system. These parts are undertaken by four electronic components: waveform digitizer board, digital I/O board, SpaceCube [2] and router board. The first three functions are implemented on the waveform digitizer board, the fourth logic on the digital I/O board, the last two in the SpaceCube, and all the electronics are inter-connected through the router board. Figures 3 and 4 show pictures of the SpaceWire boards and a block diagram of the DAQ system, respectively.

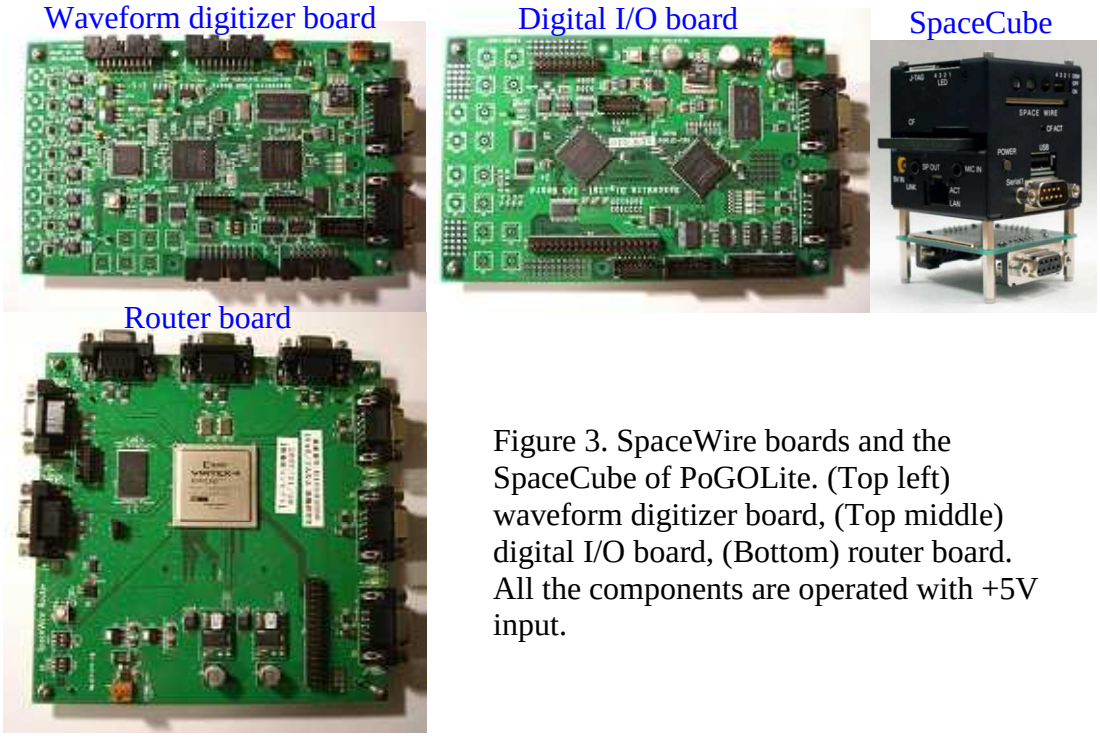


Figure 3. SpaceWire boards and the SpaceCube of PoGOLite. (Top left) waveform digitizer board, (Top middle) digital I/O board, (Bottom) router board. All the components are operated with +5V input.

Signals from the last dynodes of all PDC and SAS PMTs are fed to individual flash ADCs (FADC) and digitized to 12 bit accuracy at a 36 MHz sampling rate in waveform digitizer boards, where one digitizer board can associate up to 8 PDCs. FPGAs on the digitizer boards continuously monitor the waveforms and issue a Level-0 (L0) trigger, when a transient signal compatible with a clean hit in the fast plastic scintillator is detected above the lower discrimination level ~15 keV. A veto signal is issued when the FPGA detects a transient signal above the upper discrimination level (UD veto) or one compatible with a slow rise-time corresponding to that expected from the slow plastic scintillator or BGO (Pulse-Shape Discrimination veto, PSD veto; also see Figure 2 right).

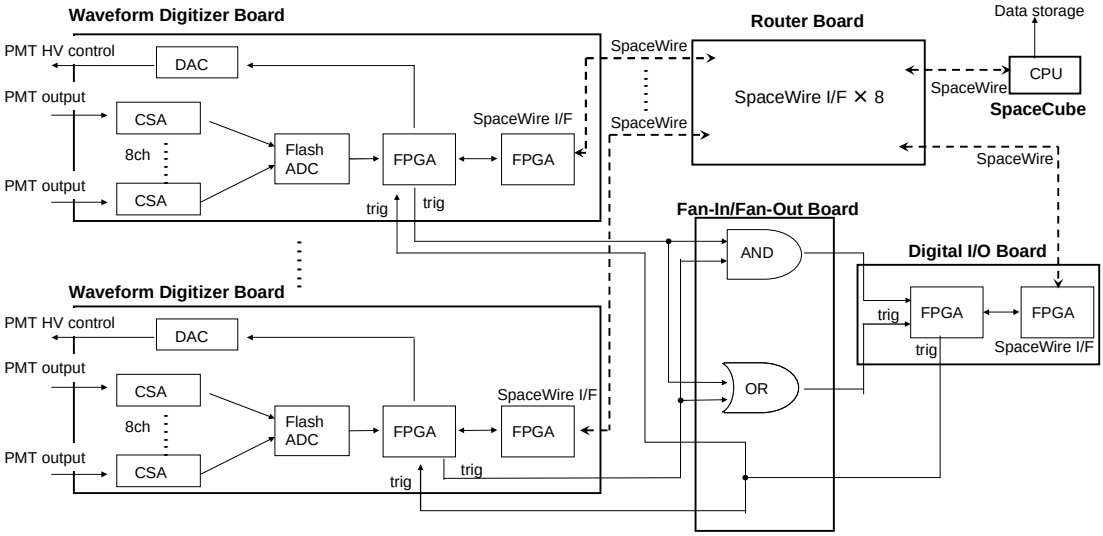


Figure 4. Block diagram of the PoGOLite DAQ system. All waveform digitizer boards and digital I/O boards are controlled and read by the SpaceCube through the router board. The reference voltage to operate each PMT (around +5V) is set by accessing the digitizer board through the SpaceWire connection.

The global event logic on-board the digital I/O board collects all L0 triggers and vetoes, and issues an acquisition request (L1 trigger) if there is no UD nor PSD veto from any PDC. When the waveform digitizer receives an L1 signal from the global event logic, it moves 15 pre-trigger and 35 post-trigger samples from each FADC to a FIFO after zero-suppression. The pre-trigger samples are used to correct for possible baseline shift due to preceding signals. Signals from SAS BGO modules are also stored when an L1 trigger is received. The only difference between the PDC and SAS digitizer boards is the programming of the FPGA. In the final step, the microprocessor in the SpaceCube records all waveforms in the storage system. Figure 5 shows waveforms for a Compton scattering and a photoabsorption, caused by a gamma-ray.

For the 217-PDC PoGOLite, the L0 trigger, the UD rate and the PSD veto rates are estimated to be about 13 kHz, 6 kHz and 12 kHz, respectively. We expect the L1 rate to be ~0.5 kHz, with a dead time fraction of ~0.6%. The validity of the UD and PSD vetoes were tested in a proton beam test at the Research Center for Nuclear Physics (RCNP) in Osaka University, which confirmed the performance of the current system for proton rates of up to ~5 kHz, which is more than an order of magnitude higher than that expected for one PDC unit in flight.

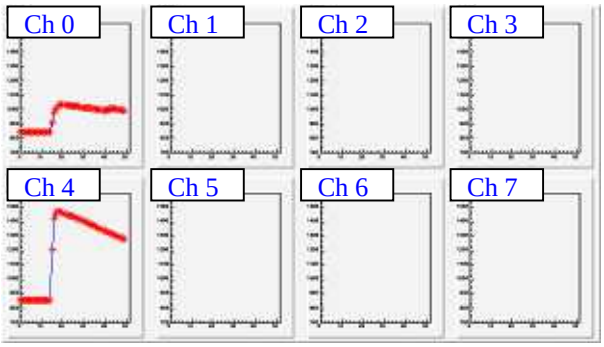


Figure 5. An example of waveforms from a gamma-ray. One photon has Compton-scattered in channel 0 (top left) and subsequently been photo-absorbed in channel 4 (bottom left) of the same waveform digitizer board. Signals from the remaining six channels were not stored due to the zero-suppression setting.

The data size of each waveform is currently 110 bytes, which includes a 10-byte header as well as the 50-clock waveform (15 pre-samples and 35 post-samples). Each one-clock waveform consists of a 12-bit pulse height and a 4-bit dummy column for a total of 16 bits (2 bytes). The header includes a sequential number, the logical address of the waveform digitizer board, an 8-channel hit-pattern for the board, presences of UD and PSD vetoes, and a 32-bit time counter (one count is 0.427 μs). During data acquisition, measured waveforms are first stored in the FPGA of each digitizer board. They are then read out by the SpaceCube through the SpaceWire connection when more than 32 waveforms are stored in one board. This reduces the required number of the read-out accesses by the SpaceCube and increases the data-acquisition rate, since the current rate is limited by the relatively long overhead of the SpaceWire access from the SpaceCube. With this configuration, we have at present obtained a maximum data-acquisition rate of about 400 waveforms per second, corresponding to ~340 Kbps. This rate is sufficient for the 61-unit pathfinder flight planned in 2010.

To support waveform sampling technique described above, the digital I/O board additionally outputs a pseudo-trigger every second if it is not occupied with data acquisition. Once the pseudo-trigger is issued, every waveform digitizer board receives this signal and simultaneously stores a pseudo event, from which we can synchronize the on-board time counters among the SpaceWire boards and estimate the dead time (from the number of the discarded events) in off-line analyses.

2.2. Background Monitoring by SAS Units

As well as storing hit-pattern information for background events coincident with the L1 trigger (see section 2.1), the waveform digitizer board for the SAS continuously records a Pulse Height Analysis (PHA) histogram with a 12-bit resolution, where the exposure of the histogram is changeable. This histogram can be used to study the in-flight background environment, which strongly relates the background contribution. To obtain an accurate energy spectrum from the PHA, we have implemented a base-line subtraction logic in the FPGA of the SAS digitizer board.

As shown in Figure 6 (left), the FPGA subtracts a “delayed” waveform from the original one and issues a trigger when the differential pulse-height exceeds a given threshold. It then subtracts the base-line (pre-trigger pulse height) and records the “corrected” pulse height in a PHA histogram. This logic allows us to measure the pulse-height (photon energy) correctly, even if a signal occurs during the undershoot

of a large pulse caused e.g. by charged particles. To verify the performance of this logic, we obtained spectra of low-intensity gamma-rays from ^{241}Am (59.5 keV) in a high-intensity background from 662 keV gamma-rays from ^{137}Cs, which simulates gamma-ray background. The result shown in Figure 6 (right) confirms the validity of the logic and demonstrates that the SAS unit can reject gamma-rays down to ~50 keV.

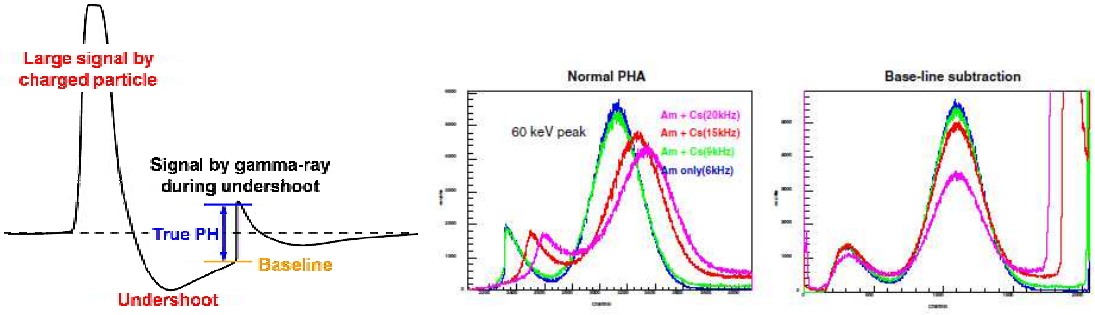


Figure 6. (Left) Base-line subtraction in the SAS PHA monitoring. (Right) Spectra of low-intensity gamma-rays from ^{241}Am (59.5 keV), recorded under irradiation of high-intensity gamma-rays from ^{137}Cs (662 keV). The peak in the left spectrum (without base-line subtraction) is shifted due to the gamma-rays from ^{137}Cs, but in the right spectrum, the peak position remains constant thanks to the base-line subtraction.

We also studied changes in the SAS PHA histogram caused by large signals from charged particles at RCNP. In these tests, SAS PHA spectra of 662 keV gamma-rays from ^{137}Cs were obtained in a background from 392MeV protons. The spectra of ^{137}Cs are shown in Figure 7, where the 662 keV peak was unaffected even with a proton intensity of up to 6.5 kHz. This rate is significantly higher than that expected for one SAS unit in flight (~1 kHz), confirming the performance of the SAS PHA at float altitude (~40 km).

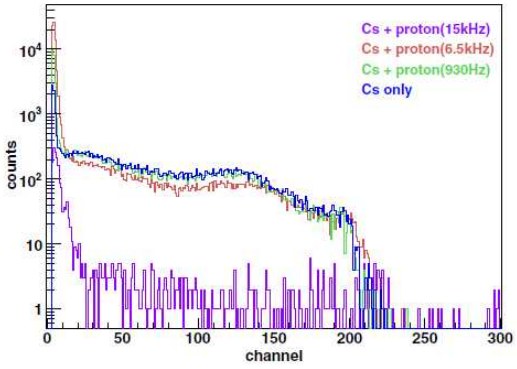


Figure 7. SAS PHA spectra of ^{137}Cs (662keV) measured under irradiation with 392 MeV proton beams with various intensities.

3. Results of KEK Beam Test with 19 PDCs and One SAS Unit

The performance of the detector units and the flight-version DAQ system was verified in a beam test, carried out at KEK-PF in February 2008. In this test, a prototype instrument, comprising 19 PDCs and one SAS unit, was irradiated by a 50 keV X-ray beam (polarization degree ~90%). As shown in Figure 8, signals from all 20 units were fed to 4 waveform digitizer boards (three for the 19 PDCs and one for the SAS).

The DAQ was constructed as described in section 2.1 with a digital I/O board, a SpaceCube and a router board. This setup is identical to that of the 61-unit pathfinder instrument, apart from the number of the waveform digitizer boards. Once the procedure is established, it is straight-forward to expand the DAQ system.

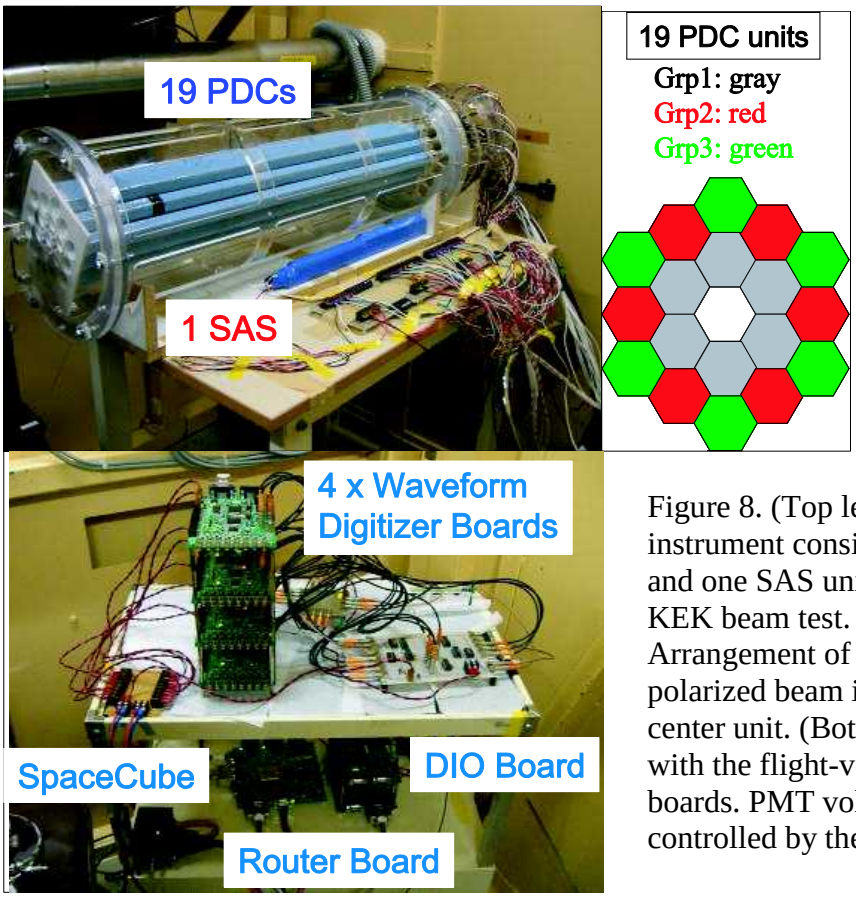


Figure 8. (Top left) Prototype instrument consisting of 19 PDCs and one SAS unit, used in the KEK beam test. (Top right) Arrangement of the 19 PDCs. A polarized beam is irradiated at the center unit. (Bottom) DAQ Setup with the flight-version SpaceWire boards. PMT voltages are also controlled by the digitizer boards.

Candidates of X-ray events with both a Compton scattering and a photoabsorption were selected from the recorded PDC waveforms in off-line analyses. The pulse-height thresholds were achieved as 0.3 photo-electron (~0.5 keV) and 15 keV for the Compton and photoabsorption events, respectively. Between successive measurements, the instrument was rotated in 30-degree steps until a full revolution had been completed. We then determined the azimuthal angle of scattering for each event, corrected it for the rotation of the detector, and derived the modulation curve. Figure 9 shows the measured modulation curve, where distributions of valid Compton events along the azimuth angle are drawn separately for three PDC groups, Grp 1, Grp 2 and Grp3 (see Figure 8 top right). The modulation depends on the distance between the Compton-scattering site and the photoabsorption site. Obtained modulation factors for the 90% polarized beam are (31.3 ± 0.4)%, (37.9 ± 0.7)% and (40.2 ± 0.8)% for Grp1, 2 and 3, respectively. These results are consistent with those predicted by our GEANT4-based simulation within ~5%.

We have evaluated the performance of the PoGOLite DAQ system and confirmed the capability to operate the 61-unit pathfinder launched in 2010. In the near future, we will test the new SpaceWire I/F developed by Shimafuji Electric which increases the read-out speed, and the new waveform digitizer board which samples waveforms with an almost doubled-clock rate (70 MHz). These improvements will allow us to increase

the data-acquisition rate, and more clearly distinguish waveforms of background neutrons from those of caused by gamma-ray events, respectively.

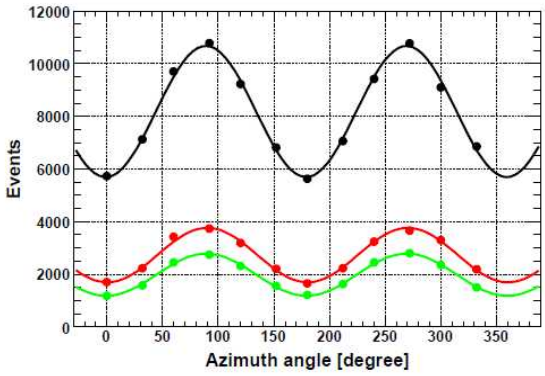


Figure 9. Measured modulations of a ~90% polarized beam with an energy of 50 keV. The curves are obtained from the three PDC sets, Grp1 (black), Grp2 (red), and Grp3 (green), shown in the top right side of Figure 8.

ACKNOWLEDGMENTS

The SpaceWire-based I/O and boards were developed in JAXA's program "Research and Development for Future Innovative Satellite."

REFERENCES

1. Tuneyoshi Kamae et al., "PoGOLite – A high sensitivity balloon-borne soft gamma-ray polarimeter" 2008, Astroparticle Physics, 30, 72-84

2. Tadayuki Takahashi et al., "Space Cube 2 - an Onboard Computer based on Space Cube Architecture", 2007, 1st international SpaceWire conference

SPACEWIRE IN THE SIMBOL-X HARD X-RAY MISSION

Session: SpaceWire missions and application

Short Paper

Cara Christophe, Pinsard Frederic.

CEA Saclay DSM/IRFU/Service d'Astrophysique,

bât. 709 L'Orme des Merisiers, 91191 Gif-sur-Yvette, France.

E-mail: christophe.cara@cea.fr, frederic.pinsard@cea.fr

ABSTRACT

SIMBOL–X is a hard X–ray mission, operating in the 0.5–70 keV range [1], which is proposed by a consortium of European laboratories for a launch around 2013. Relying on two spacecrafts in a formation flying configuration, SIMBOL–X will allow to elucidate fundamental questions in high energy astrophysics, such as the physics of accretion onto Black Holes, of acceleration in quasar jets and in supernovae remnants, or the nature of the hard X–ray diffuse emission.
The instrument combines three type of detectors: a silicon low energy detector on top of a cadmium telluride high energy detector and a scintillator which surrounds them except for the solid angle corresponding to the focused beam from the mirror. Instrument performance is expressed in particularly in term of dead time, which defines in turn the time tagging resolution and relative accuracy of the events from the three detectors. Therefore the SIMBOL-X instrument requires an accuracy of 100 ns. In the presentation we will focus on the SpaceWire [2] standard Time-Code [3] use limitation and provide a way to improve it with a minor upgrade of the standard to reach the expected performances.

1 INTRODUCTION

The SIMBOL-X instrument is an X-ray imager relying on two focal planes to fulfil the energy range requirement: the 0.5 keV to 12 keV sub-range is covered by the silicon detector while the 8 keV to 70 keV sub-range is covered by a 2 mm thick cadmium telluride detector. Each focal plane is a 128 by 128 pixel matrix. A possible cause of performance limitation of such type of instrument is due to cosmic background noise, which decrease its overall efficiency. A basic way to reduce this background is to surround the detector, except in front, with a multilayer shield made of materials having various atomic masses. Depending on its atomic mass and thickness each material traps incoming particles in complementary energy sub-ranges. However the efficiency of this shield is not 100%. Better results may be even

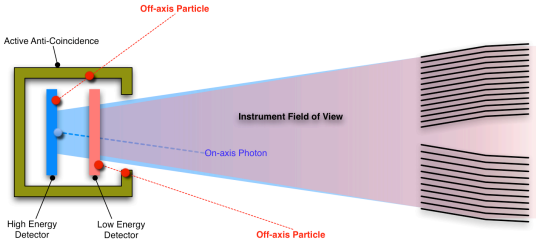


Figure 1 – Instrument layout

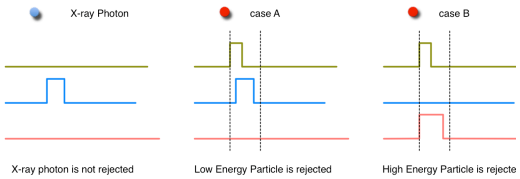


Figure 2 – Active anticoincidence

obtained by adding an active shield to this passive shield. This active shield is made of a scintillator material (CsI, crystal, …) which generates photons when crossed by noise particles. These photons are then detected by means of either photo-multiplier tubes or photo-diodes. Almost immediately after this first interaction the particle hits the focal plane detector and in turn generates an event. The resulting instrument optical layout is illustrated in figure 1.

Therefore rejection of background events is simply achieved by eliminating time-correlated events between the active shield detector and the focal plane detector. The so-called anticoincidence mechanism is illustrated in figure 2 in the case of two focal plane detectors as in SIMBOL-X. The major contribution is the reduction of the telemetry volume: when observing faint sources the X-ray photon rate could be as low as a few counts per second while the background-generated events reaches a rate of several hundreds.

In order to take into account various uncertainties (electronic noise, propagation delay jitter, …) an anticoincidence window is defined: all events occurring within this window shall be rejected. However the width of this window shall be carefully chosen since it determines the efficiency in term of unavailability of the instrument also called the *dead time*. The Figure 3 shows the impact on the *dead time* for window width varying between 10 ns and 100 µs for 10000 events per second. The required 1% of *dead time* for the SIMBOL-X instrument limits the window width to 1 µs.

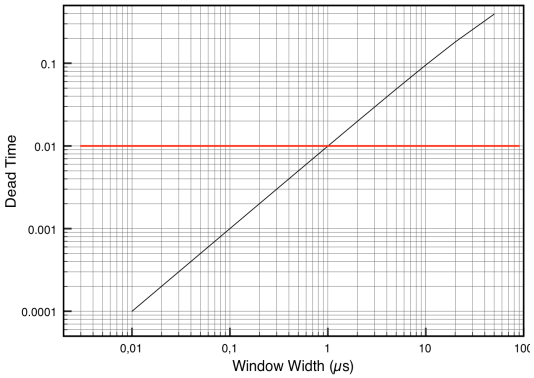


Figure 3 – *Dead time* vs Window width

As shown in figure 4 the instrument comprises 3 detection sub-systems: high energy, low energy and active shielding. In turn each sub-system comprises the detector located in the instrument focal plane and the associated control electronics. A last sub-system (the Data Processing Assembly) is in charge of the whole instrument control and the processing of both scientific and engineering data. In order to optimise interface definition the SpaceWire standard was adopted to handle this bidirectional data flow. Detection sub-systems act has destination nodes while DPA acts as transmission node. Among the exchanged data the events defined by an amplitude, a position and a time tag are received by the DPA. Then the DPA shall check time correlation between events to reject unwanted ones by comparing the time tags of the incoming events within the coincidence window. The time tag accuracy is assumed to be one 10th of the window width. Therefore the SIMBOL-X performance requirement implies a relative time tag accuracy of 100 ns between detection sub-systems. The following table summarizes the various data flows exchanged with the DPA. As shown maximum peak data is limited to 20 Mbps: it determines the SpaceWire operating signalling rate. Each data flow is using a dedicated virtual channel identified by mean of specific *protocol id* and *packet type ids*.

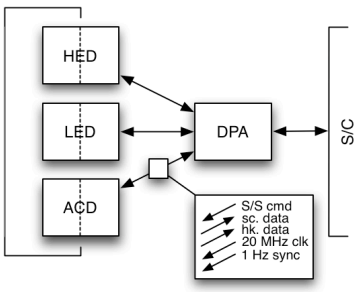


Figure 4 – Instrument Overview

Direction	Data flow	S/S		
		HEDEA	LEDEA	ACDEA
Downstream	Scientific Data	0.35 Mbps	0.80 Mbps	0.01 Mbps
	(peak)	< 20 Mbps	< 20 Mbps	
	Engineering Data	0.024 Mbps	0.024 Mbps	0.002 Mbps
Upstream	S/S Commands	< 0.1 Mbps	< 0.1 Mbps	< 0.1 Mbps

2 SPACEWIRE STANDARD & EXTENSION

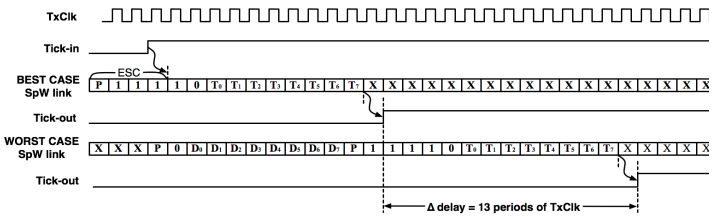


Figure 5 - Best and Worst TIME-CODE transmission delay

The SpaceWire standard specifies the TIME-CODE character to propagate the time across the network [4]. Currently TIME-CODE transmission request occurs asynchronously with respect to the transmitted character stream. Therefore the delay between the TIME-CODE request and the effective character transmission is equal to the time left for the transmission of the current character. The delay difference between the best and the worst cases is then 13 transmission clock periods: best is when an ESC transmission is about to end, worst is when a Data Character has just started. Best and worst case timings are represented in Figure 5. The best achievable time synchronization accuracy through SpaceWire links will be then 1.3 µs for a 20 Mbps transmission rate or 100 ns for a 260 Mbps transmission rate. The increase of the interface frequency well above the need for instrument data transmission - 20 Mbps- is not acceptable since it adds constraints to the design and to the power budget. Alternative solution could be to add a dedicated interface devoted to synchronization, but again with an impact on system budget.
Finally the decision was taken to work around the existing TIME-CODE to increase its accuracy and especially by taking into account the highest priority of this TIME-CODE defined by the standard. First of all the idea is to measure the delay between TIME-CODE request and its effective transmission and then to find a way to send this delay to the destination node in order to compensate this delay. Thanks to priority scheme the only solution is to send a second TIME-CODE immediately after the first one, which carries the measured delay. It allows creating a constant delay between the TIME-CODE transmission request in the transmission node and a synchronisation signal in the destination nodes.

3 IMPLEMENTATION

The block diagram of the SpaceWire and the proposed extension is given in Figure 6. Added functions are the "Time_TX" and "Time_RX" functions. No modification of the SpaceWire standard core is required except the Ack_Time signal, which is added to the "Tx" function and used by the new "Time_TX" function. To be noticed: to enable the implement of this extension, access to TIME-CODE recovery clock and acknowledgement signal is needed. The extension implementation is low resource consuming: it requires only 62 of 4024 (1.5%) combinational cells and 42 of 2012 (2%) sequential cells of an RTSX-SU72 ACTEL FPGA.

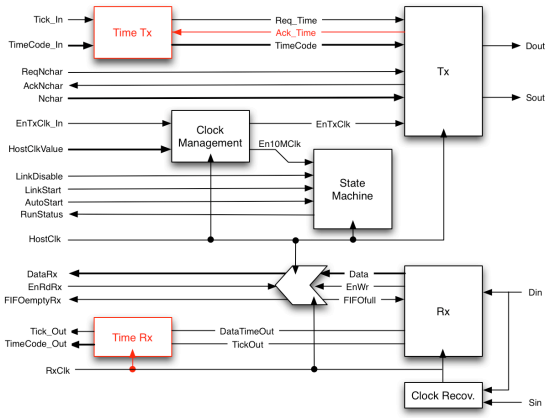


Figure 6 – SpaceWire with extension

4 RESULTS AND DISCUSSIONS

To validate the performance of the extension a prototype of the SpaceWire network was realized: a home made PCI acquisition board implementing four SpaceWire interfaces simulates the DPA while two detector acquisition boards simulate the HED and ACD electronics respectively. The block diagram of this test configuration is depicted in Figure 7.

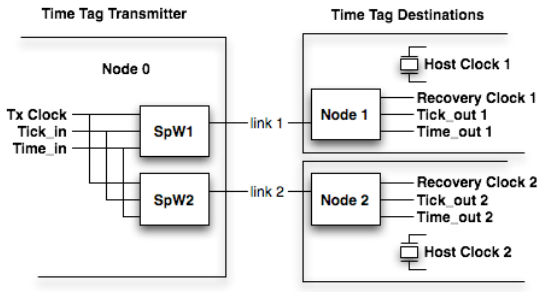


Figure 7 – Block diagram of the tested configuration

First timing measurement on this prototype is shown in Figure 8. The upper trace is the TICK_IN signal of the transmitter and the two next traces (Red and Green) are TICK_OUT signals of the two destination nodes. The rising edge of the TICK_OUT signal is not re-synchronized and then a large jitter of almost 160 ns is measured. This jitter corresponds to what could be obtained with a standard SpaceWire. The falling edge of the same signal is re-synchronized with the extension. The resulting jitter is as low as about 4 ns and is due to cables length and propagation delay mismatches. Practically in the SIMBOL-X instrument the TICK_IN TICK_OUT interface will be used to propagate a 1 Hz synchronisation signal generated by the DPA and synchronous to the satellite on-board time, which will reset time tag counters in each detector electronic. These counters will be incremented by mean of the SpaceWire recovery clock in order to avoid any time drift between detectors.

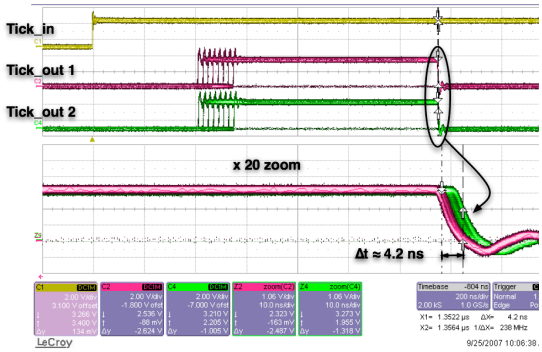


Figure 8 – Timing Diagram

5 CONCLUSION

With the proposed extension a single interface standard fulfil all the needs in terms of data transmission: both scientific, engineering and command and time synchronisation for the SIMBOL-X instrument. It optimizes overall instrument architecture, simplifies integration tasks and test equipment design. The extension fits perfectly within limited available hardware especially in the detector electronics were only small FPGA are foressen. Further improvements such as calibration and compensation of propagation delay mismatch could be done.

6 REFERENCES

[1] Ph Ferrando and al. “SIMBOL-X: mission overview”, *proc. SPIE 6266, p.62660 (2006).*

[2] S.M. Parkes et al, “SpaceWire: Links, Nodes, Routers and Networks” European Cooperation for Space Standardization, Standard No. ECSS-E50-12A, Issue 1, January 2003.

[3] Steve Parkes “The Operation and Uses of the SpaceWire Time-Code”, International SpaceWire Seminar, ESTEC Noordwijk, The Netherlands, November 2003.

[4] F. Pinsard and C. Cara “High resolution time synchronization over SpaceWire links”, Aerospace Conference 2008, IEEEAC paper#1158, 10.1109/AERO.2008.4526462

Missions & Applications 2

Thursday 6 November
15:20 – 16:50

RMAP OVER SPACEWIRE ON THE EXOMARS ROVER FOR DIRECT MEMORY ACCESS BY INSTRUMENTS TO MASS MEMORY

Session: SpaceWire missions and applications

Short Paper

Bryan Dean, Rosalind Warren, Ben Boyes

EADS Astrium, Gunnels Wood Road, Stevenage, SG12AS, UK

E-mail: bryan.dean@astrium.eads.net, rosalind.warren@astrium.eads.net, ben.boyes@astrium.eads.net

ABSTRACT

The ExoMars Rover incorporates a number of high bandwidth instruments including cameras and microscopes which connect to the On Board Computer using SpaceWire technology. We intend to use the Remote Memory Access Protocol (RMAP) to allow the instruments to write large amounts of data to the mass memory within the computer without application software involvement. This will free up much needed processing time for the autonomous navigation algorithms.

Since the file system management tasks will be performed by the application software (ASW) on the processor, the mass memory controller will not have knowledge of the file or data structure in the logical address space. As a result, all write accesses to the mass memory must be controlled by the application software. We are developing a mechanism using RMAP to allow the software to initiate a transfer of data from a specified memory address in an instrument to a specified memory address in the mass memory. Once the data transfer has been initiated, the software can perform other tasks whilst waiting for acknowledgement of a successful transfer.

In order to achieve the high level of processing required for the Rover to navigate and drive autonomously on the surface of Mars, the initial processing of images used for navigation will be performed by dedicated hardware in the On Board Computer. The same mechanism for data transfer from camera to mass memory will be used to send image data directly to this hardware. In this paper we outline the details of this mechanism.

1 DATA HANDLING SYSTEM

The Rover Vehicle Data Handling System is designed with a centralised unit, the On Board Computer (OBC) connected to the instruments and other subsystems. The OBC provides the processing power, memory facilities and overall control. It interfaces to the science equipments and to other subsystems using either SpaceWire or CAN bus links. The use of CAN bus reduces harness size and power by connecting several nodes onto a single bus, at the expense of reduced bandwidth. The CAN bus is used for equipment with low bandwidth requirements but for the optical instruments and for the cameras which generate large data packets that are required promptly for navigation, high speed SpaceWire links are essential.

The processing power is provided through a combination of dedicated hardware in a co-processor and application software running on the processor. The dedicated hardware is used for certain repetitive image processing functions, in order to speed up navigation algorithm execution. The processor provides a platform for software which is more flexible and can be used according to the current tasks being performed by the Rover, for example to execute path-planning algorithms or to compress science data. CPU load is minimised by sending science data directly to the mass memory where possible, without application software involvement.

Within the OBC there are SpaceWire nodes on the mass memory, on the processor and on the hardware image processing within the co-processor. Each instrument with a SpaceWire interface is connected directly to the OBC by a point-to-point link (in some cases cold redundant). SpaceWire routing between the nodes is implemented within the OBC. The SpaceWire links to the cameras are cold redundant; the OBC has the responsibility to select the operational link. The links to the CLUPI and MicrOmega science instruments are single links, with cross-strapping within the OBC.

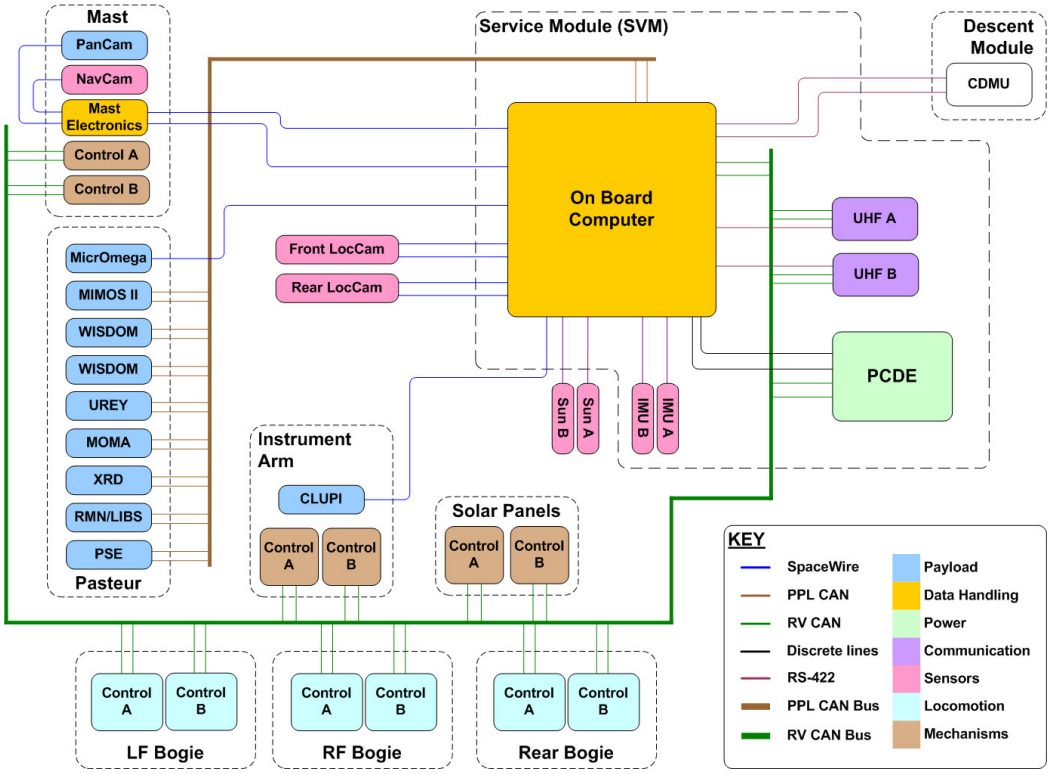


Figure 1: Rover Vehicle Data Handling System

2 DATA TO CO-PROCESSOR

The navigation cameras acquire a pair of stereo images which are used to generate a digital elevation model (DEM) of the surrounding terrain. This DEM is then used to generate a navigation map and plan a path to the specified destination. The Rover will stop every couple of meters to acquire more navigation images and update the navigation map. It is essential that the image processing time during these stops is minimised to enable the Rover to travel a reasonable distance in a Martian day or Sol.

In addition the localisation cameras acquire images whilst driving and the OBC identifies features which are used to track the progress along the planned path and avoid collisions. The speed at which the Rover can drive is limited by the time taken to process the images.

The dedicated hardware in the co-processor reduces this processing time considerably but requires that the images can be sent directly from the cameras to the dedicated hardware (not via the processor). SpaceWire routing allows images from the navigation cameras, the front localisation cameras or the rear localisation cameras to be sent directly to the dedicated image processing hardware in the co-processor. RMAP is used on the Rover to enable the processor to initiate a download of the images directly to the co-processor.

3 DATA TO MASS MEMORY

The memory for science and housekeeping data storage is provided by a mass memory function with memory controller. The majority of the memory is non-volatile but a small volatile area is provided for temporary storage, for example of science data prior to compression. Science data can be written directly to the mass memory, using RMAP for data arriving over the SpaceWire link. The memory controller includes logical to physical address mapping as well as error correction.

The mass memory is essential to provide immediate storage of science data for later processing by software and to provide storage of processed scientific data between the limited downlink windows. The memory is accessed by the application software using logical addressing.

Data received on the SpaceWire interface can be written directly to the volatile part of the mass memory without software intervention. RMAP is used to provide the memory addressing. The ASW enables and disables the access to the mass memory for each individual instrument, identified by the source address in the header of the SpaceWire packet.

Following storage in the volatile memory, the data can be processed by the ASW when there is available CPU capacity. Once this data is in the format required for transmission to Ground, the ASW writes the data to the file system in the non-volatile memory to wait for availability of return-link bandwidth.

4 RMAP OVER SPACEWIRE

Since RMAP is used to write data from the ASW to the mass memory, RMAP is also used to write commands to the instruments and to receive data back. All commands initiated by the OBC are initiated by the processor but responses may be required to be sent to the processor, the hardware image processing in the co-processor or the mass memory. When the processor requests data to be written to the mass memory, it supplies the RMAP header in a write command sent to the instrument, to enable the instrument to reply directly to the mass memory. The instrument then initiates a RMAP write command to send the data to the mass memory or co-processor using the RMAP header provided by the processor.

Each acquisition command to an instrument results in a write command from the instrument back to the OBC. For simplicity, the first 12 bytes of the RMAP header for the write command back from the instrument are provided in the cargo of the write command to the instrument. The instrument does not need to check all the contents of these 12 bytes, nor use the contents of all of these bytes instead of values known locally (e.g. Response Protocol Identifier). This is provided simply to allow for flexibility of implementation in the instrument. The transaction identifiers used will differ depending on the choice made.

OBC Command byte	Response Destination Logical Address	Response Protocol Identifier	Response Packet Type / Command / Source Path Addr Len
Response Destination Key	Response Source Logical Address	Response Transaction Identifier (MS)	Response Transaction Identifier (LS)
Response Extended Write Address	Response Write Address (MS)	Response Write Address	Response Write Address
Response Write Address (LS)	Data CRC	EOP	
	Last byte transmitted		

Figure 2: Cargo - Instrument Acquisition Write Command

If the data is required to be returned directly to the processor, a simple RMAP read command to the instrument will suffice, although additional write commands are required to trigger the data acquisition.

The exchange is illustrated Figure 3 from the point of view of a single instrument, showing a RMAP write to the instrument requesting a further RMAP write to mass memory.

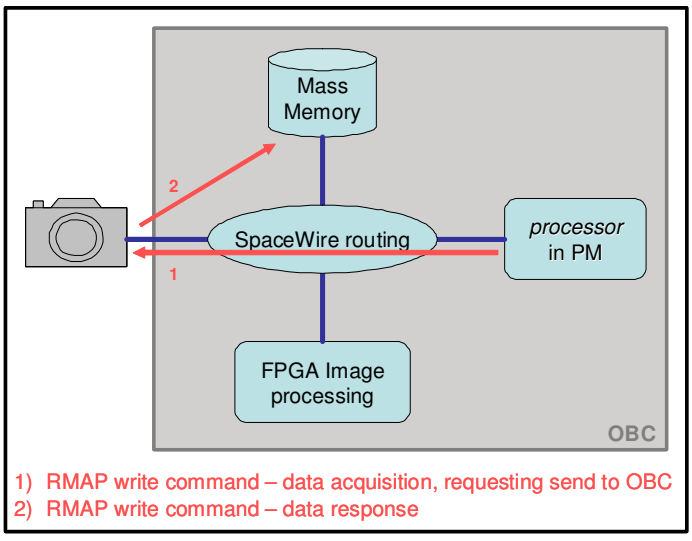


Figure 3: Data acquisition across SpW from an instrument to the mass memory

STANDARD ONBOARD DATA HANDLING ARCHITECTURE BASED ON SPACEWIRE

Session: Missions and Applications

Short Paper

Takahiro Yamada, Tadayuki Takahashi

JAXA/ISAS, 3-3-1 Yoshinodai, Sagamihara, 229-8510, JAPAN

E-mail: tyamada@pub.isas.jaxa.jp, takahasi@astro.isas.jaxa.jp

ABSTRACT

This paper presents the standard onboard data handling architecture that JAXA is developing presently. This architecture specifies an architectural framework on how to develop onboard data handling systems and will be used for all the science spacecraft that JAXA will develop. This architecture consists of three sub-architectures: physical architecture, functional architecture and protocol architecture. JAXA is also developing standard components based on this architecture, which include standard physical components and standard functional components. By using this architecture, the basic portion of onboard data handling systems will be developed by selecting appropriate standard components and connecting them with standard protocols. This will facilitate the design, integration and testing of onboard data handling systems greatly.

1 INTRODUCTION

Presently, the onboard data handling systems of most spacecraft are designed individually for each spacecraft without using any framework. This prevents reuse of components from spacecraft to spacecraft. In order to solve this problem, the Institute of Space and Astronautics Science (ISAS) of the Japan Aerospace Exploration Agency (JAXA) is developing a standard onboard data handling architecture, which specifies an architectural framework on how to develop onboard data handling systems. This architecture will be used for all the future science spacecraft that JAXA will develop. This architecture consists of three sub-architectures: physical architecture, functional architecture and protocol architecture.

The physical architecture specifies how to configure onboard data handling systems physically and defines basic physical elements. Any onboard data handling system will be constructed physically by connecting basic physical elements according to the characteristics and the complexity of the spacecraft. The functional architecture specifies how to configure onboard data handling systems functionally and defines basic functional elements. These functional elements are implemented in physical elements. The protocol architecture specifies how to connect physical and functional elements with communications protocols and defines a set of standard protocols to be used.

JAXA is also developing standard components based on this architecture, which include standard physical components and standard functional components. By using this architecture, the basic portion of onboard data handling systems will be developed by selecting appropriate standard components and connecting them with standard protocols. The difference in the size of different spacecraft will be reflected in the number of components used in each spacecraft. The difference in the characteristics of different spacecraft will be reflected in the way of combining different components in each spacecraft. This method of developing onboard data handling systems will facilitate the design, integration and testing of spacecraft greatly.

2 PHYSICAL ARCHITECTURE

In this architecture, a physical onboard element that handles data in any way (for example, generates, uses, processes, relays, and/or stores data) is called a Node. Furthermore, this architecture defines two types of Nodes: Intelligent Nodes and Non-intelligent Nodes.

An Intelligent Node is defined to be a Node that has one or more processors and can generate and consume Space Packets [1]. A Non-intelligent Node is defined to be a Node that does not have a processor and cannot generate or consume Space Packets. However, the distinction between Intelligent and Non-intelligent Nodes is not always clear and there may be physical elements with features of both Intelligent and Non-intelligent Nodes. Therefore, this architecture should be regarded as guidelines, rather than strict rules. Examples of Intelligent Nodes are command and data handling units, mission processors, attitude control processors, etc. Examples of Non-intelligent Nodes are sensors of various kinds, actuators, etc.

The onboard data handling system of any spacecraft should be physically constructed as a tree. The leaves of the tree are Non-intelligent Nodes, and the other nodes of the tree are Intelligent Nodes. All the Nodes are connected with standard communications protocols that will be described later. Some examples of physical configurations of onboard data handling systems are shown in Figure 1.

Based on this architecture, we are developing two kinds of standard physical components that can be used on any spacecraft. The first kind is a series of standard onboard computers that can be used as Intelligent Nodes of any spacecraft. These onboard computers are developed based on the SpaceCube architecture [2] and each computer has a processor, a real-time OS, SpaceWire interfaces, etc. For Non-intelligent Nodes, we are developing a standard interface card with a SpaceWire interface that is used for communications with an Intelligent Node.

3 FUNCTIONAL ARCHITECTURE

Each node (physical element) has a set of functions. What is important in this architecture is that the standard physical components explained above can be used for any Node regardless of the functions of the Node. The functions of a standard onboard computer (SpaceCube) is determined by the applications programs that run on the computer and what applications programs are to be run on each of the onboard computers on a particular spacecraft is determined for each spacecraft. Therefore, the same SpaceCube can be used for attitude control, mission data processing, or any

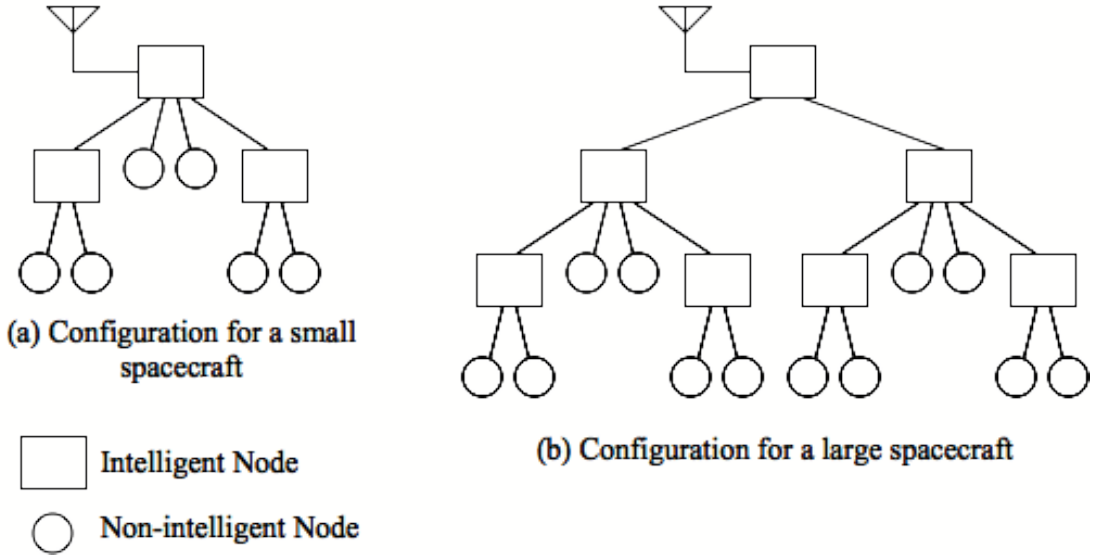


Figure 1 – Examples of physical configurations of onboard data handling systems

purposes. Likewise, the standard interface card can be used for any Non-intelligent Node regardless of the functions of the Node.

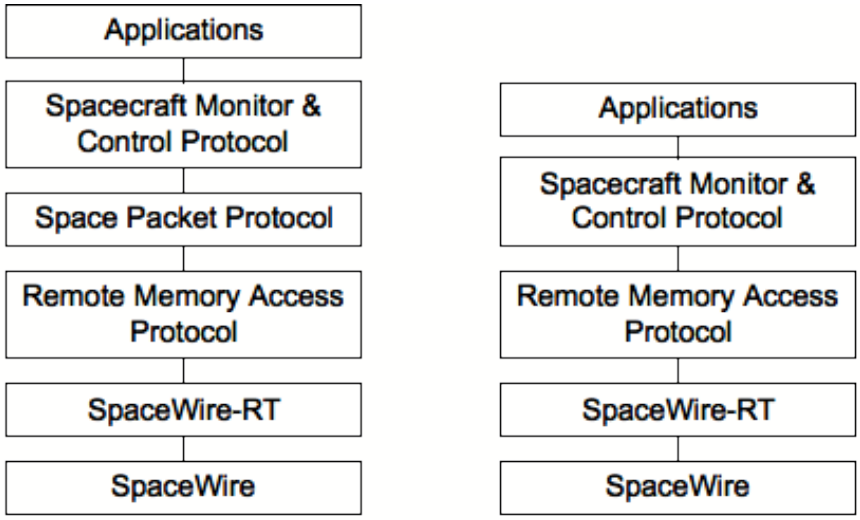
Generally, a Non-intelligent Node is monitored and controlled by the parent Node, which is the Intelligent Node to which it is directly connected. Therefore, the tree of the Nodes (physical elements) on a spacecraft corresponds to the controlling hierarchy of the spacecraft.

We are planning on developing standard functional components that can be used on any Intelligent Nodes. These components are pieces of software that use the capabilities of the Space Packet Protocol and the Spacecraft Monitor and Control Protocol (SMCP) described in the next section and support the operations of the applications programs running on the processor of the Nodes.

4 PROTOCOL ARCHITECTURE

Nodes (physical elements) are connected with standard communications protocols, which are shown in Figure 2. There are two protocol stacks in Figure 2. The first stack is used between two Intelligent-Nodes and the second stack is used between an Intelligent Node and a Non-intelligent Node. The difference between these two stacks is that the Space Packet Protocol [1] is used only between Intelligent-Nodes.

As the lower layer protocols, SpaceWire, SpaceWire RT [3] and SpaceWire Remote Memory Access Protocol (RMAP) [4] are used for all interfaces between Nodes. The Space Packet Protocol is used between Intelligent Nodes on top of the SpaceWire protocols. On top of all these protocols, the Spacecraft Monitor and Control Protocol (SMCP) [5], which is a JAXA standard protocol, is used to monitor and control the functions of the Nodes. SMCP is used by an onboard Intelligent Node to control other Intelligent and/or Non-intelligent Nodes and by a spacecraft control system on the ground to control onboard (Intelligent and/or Non-intelligent) Nodes.



(a) Between Intelligent Nodes

(b) Between Intelligent and Non-intelligent Nodes

Figure 2 – Standard protocols

5 CONCLUSION

This paper presented the standard onboard data handling architecture that JAXA is developing. JAXA is also developing standard hardware and software components that can be used on any spacecraft based on this architecture. These components will be used on all future science spacecraft of JAXA, which include TOPS (EUV telescope mission), ASTRO-H (X-ray telescope mission), SPICA (infrared telescope mission), etc.

6 REFERENCES

1. Consultative Committee for Space Data Systems (CCSCS), “Space Packet Protocol”, CCSDS 133.0-B-1, September 2003.

2. H. Hihara, “Designing SpaceCube 2 with Elegant Framework”, International SpaceWire Conference 2008, Nara, Japan, October 2008.

3. S. Parkes, “SpaceWire RT”, International SpaceWire Conference, Nara, Japan, October 2008.

4. European Cooperation for Space Standardization (ECSS), “SpaceWire Protocols,” ECSS-E-ST-50-11C, Draft 1.3, July 2008.

5. T. Yamada, “Standardization of Spacecraft and Ground Systems Based on a Spacecraft Functional Model”, SpaceOps 2008, Heidelberg, Germany, May 2008.

SPACEWIRE TEST AND DEMONSTRATION UTILISING THE INTEGRATED PAYLOAD PROCESSING MODULE

Session: Missions and Applications

Short Paper

Jørgen Ilstad, David Jameux

European Space Technology Centre, Noordwijk, Netherlands

E-mail: jorgen.ilstad@esa.int, david.jameux@esa.int,

ABSTRACT

A demonstration platform called SpaceWire Test & Demonstration Platform (SpW TDP) is being set up in the Avionics/Data laboratory of the On-Board Data Systems Division at ESTEC. The TDP will be used to perform Onboard Data Handling related research and demonstrations with focus on SpaceWire network and protocols. The first version, TDP 1.0, will be tailored to resemble a simplified onboard data handling system which communicates with a simulated ground station for telemetry (both House Keeping and science TM) and telecommand. Its implementation will be based on the Integrated Payload Processing Modules (IPPM) and complementary demonstration equipment.

1 INTRODUCTION

A demonstration platform called SpaceWire Test & Demonstration Platform (SpW TDP) is being set up in the Avionics/Data laboratory of the On-Board Data Systems Division at ESTEC. The TDP will be used to perform Onboard Data Handling related research and demonstrations with focus on SpaceWire network and protocols. The first version TDP 1.0 will be tailored to resemble a simplified onboard data handling system which communicates with a simulated ground station for telemetry (both House Keeping and science TM) and telecommand.

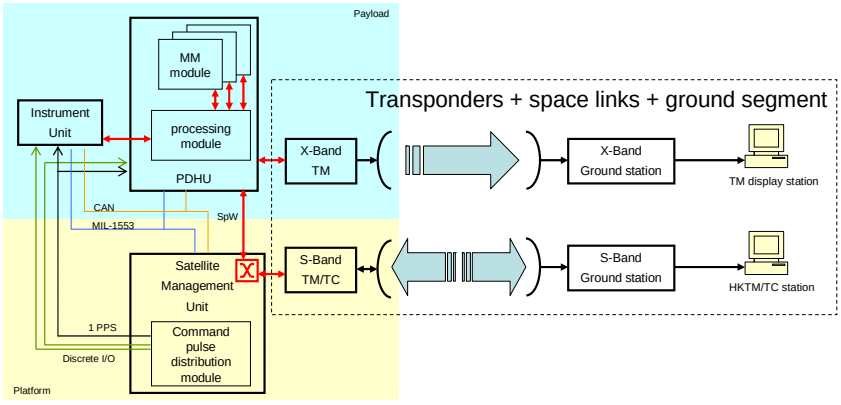


Figure 1: Functional overview of the OBDH demonstration (TDS 1.0).

2 ELEMENTS OF THE TEST & DEMONSTRATION PLATFORM 1.0

2.1 THE INTEGRATED PAYLOAD PROCESSING MODULE

A versatile onboard computer and storage system called IPPM [1], based on the Leon2-FT processor (AT697 ASIC), was developed in the frame of the ESA contract 18780/04/NL/LvH. The IPPM embeds a UoD-SpW-10X routing switch, supporting 8 high speed external SpW links while the two parallel links are connected to a Control and Communication FPGA. The support for the Remote Memory Access Protocol (RMAP) has been embedded in the Control and Communication FPGA, which allows for remote access of the IPPM I/O space over SpaceWire. For low medium rate applications such as for command and control, it also features redundant CAN and MIL-1553B links. The IPPM features, in addition, 262MB of on-board memory which is accessible via the Remote Memory Access Protocol (RMAP).

The IPPM provides the processing capability as well as all the standard interfaces to investigate the use of SpaceWire-based protocols with respect to current OBDH systems based on “classical” buses and protocols and on discrete lines.

2.2 RTEMS

RTEMS is an Open Source RTOS providing a powerful development and run-time environment that promotes the production of efficient real-time embedded applications. The current release 4.9 has been selected as the operating system to be used for each of the IPPM boards. Applications on each of the IPPM units are developed to conform to services defined in ECSS-E-70-41A. The selection of RTEMS is rooted firstly by its support by the European Space Agency related to the RTEMS Centre [10] and secondly by its use in several space projects. SMU’s used for the ESA missions such as Herschel, Plank and GAIA, to name a few, make use of RTEMS on SPARC architecture onboard computers.

Coupling RTEMS with SpaceWire related development activities on the TDP intends to identify requirements and constraints at both SW and HW level, as well as best practices for implementation.

2.3 PACKET UTILISATION STANDARD

The Packet Utilisation Standard (PUS), ECSS-E-70-41A, defines standard services for onboard components, which address the utilisation of telecommand and telemetry source packets. The ECSS-E-70-41A is a comprehensive standard which, in the frame of the development of the TDP 1.0, has been limited to include a sub-set of the services defined in Telecommand Verification Service, Device Command Distribution Service, Memory Management Service and Function Management Service.

PUS is an element of the TDP not only because it provides a standard application process layer and its extensive use in ESA missions, but also because it allows experiments to be performed when introducing newer SpW protocols on an RTEMS based onboard system.

3 ARCHITECTURE OF THE TEST & DEMONSTRATION PLATFORM 1.0

The demonstration platform is based on three IPPM boards. It includes complementary demonstration equipment to obtain Mass Memory functionality, and a TM/TC and Display unit that serves as a Ground Control Unit (GCU). The overall demonstration set-up aims at resembling a simplified onboard data handling system which communicates with a simulated ground station for telecommand uplink and HK and science telemetry downlink. Each of the three IPPM boards will have a dedicated role as an onboard data handling unit. One IPPM board will act as an Instrument Control Unit (ICU), while the other two boards act as Processing Module for the Payload Data Handling Unit (PDHU) and as Satellite Management Unit (SMU). Each IPPM board will be running RTEMS as Operating System (OS) as well as applications that support a subset of services in accordance with the Packet Utilisation Standard [6].

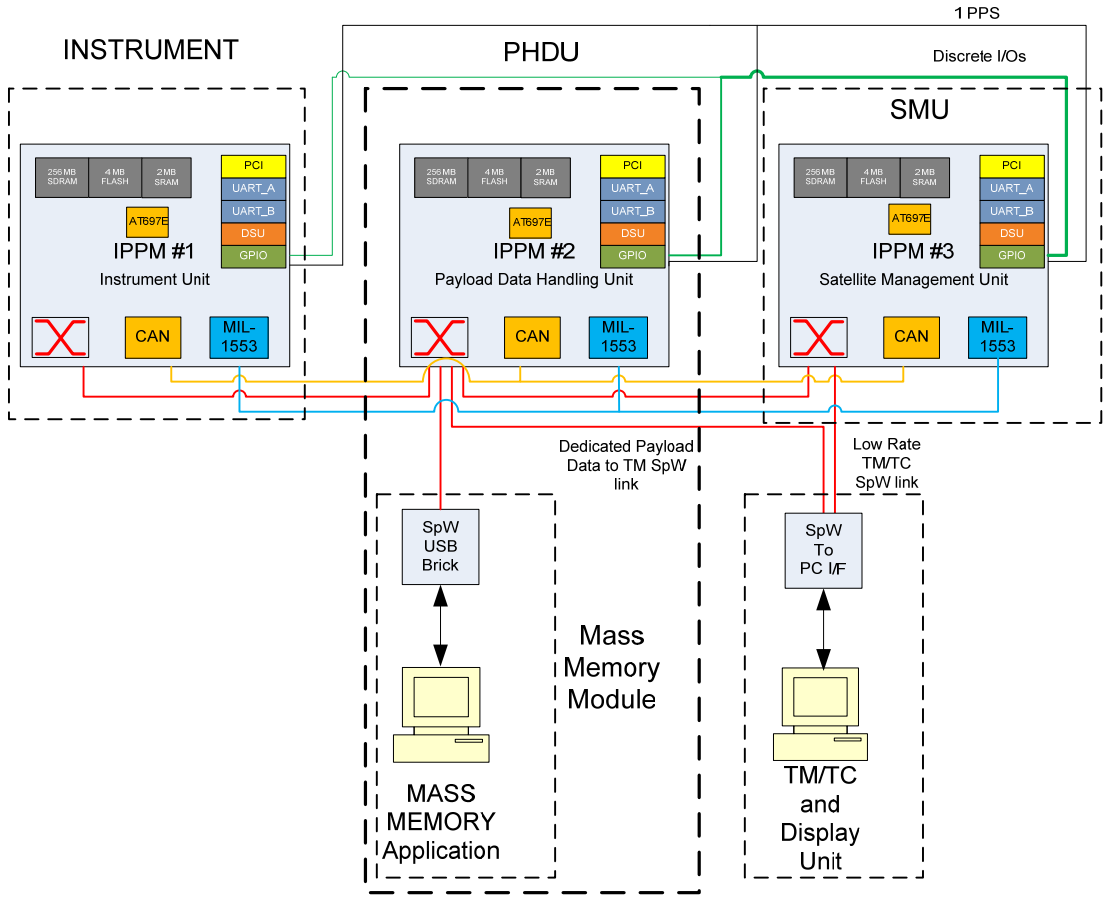


Figure 2: SpaceWire Demonstration Set-up baseline.

4 SPACEWIRE DEMONSTRATION SCENARIOS

4.1 TEST AND VALIDATE SPW NETWORK TOPOLOGIES AND SPW-ENABLED DEVICES

The features of the IPPM board will first and foremost be exploited in the demonstration baseline to identify best practices for SpaceWire network topologies e.g. redundancy schemes and use of current SpW networking protocols RMAP[9] and CCSDS Packet Encapsulation, defined in the soon to be released ECSS-E-50-11 standard [5]. Further, by inclusion of other units to the demonstration baseline, it will be used to establish and demonstrate best practices for current and near-coming use of SpW-enabled devices. To highlight some candidates one could mention e.g. general purpose processing SPARC/Leon2 nodes (SpW-RTC, ERC32, LEON-2 board), dedicated processing modules (compression, FFT, etc), and Non-intelligent nodes with no processing capabilities (sensors and instruments, I/O boards etc.). In essence, the demonstration baseline may be utilised to validate future SpW-enabled devices towards current SpW-based data handling systems.

4.2 INVESTIGATE SPW-BASED TIME DISTRIBUTION

The Pulse Per Second (PPS) signal is a common time distribution solution for onboard systems. As the IPPM features GPIO ports, the capability to distribute PPS is incorporated in the TDP 1.0 baseline. One demonstration scenario is to perform time distribution (for clock synchronisation), a pre-requisite for Command & Control protocols, using SpW time-codes, and to compare this “SpW time” with the “PPS time”. This demonstration scenario will take into account different network topologies and link failure scenarios. If the “SpW time” proves to satisfy Cmd&Ctrl application, SpW time codes could replace the PPS signal in OBDH systems.

4.3 PROTOTYPING AND VALIDATING SPW-BASED PROTOCOLS

At present, there are several parallel initiatives ongoing to develop standardised protocols, e.g. SpW PnP [2,3] and SpW-RT [7] to be used on top of SpaceWire. Utilising the IPPM for this purpose is beneficial as it embeds several of the most recent technology advances, such as the SpaceWire Router 10x, AT697E Leon2 processor and SpW RMAP support. Firstly, it will facilitate answering questions regarding not only how these protocols must be defined, but also if they are compatible with each other. Secondly, exercising these protocols from the IPPM will allow identifying requirements from a software point-of-view when utilising Real Time Operating Systems such as RTEMS.

Similarly, the SpaceWire, CAN and MIL-STD1553 bus interfaces in conjunction with its processing capabilities provided by the IPPM, make the IPPM good candidate for experimentation related to CCSDS Spacecraft Onboard Interface Services initiative (CCSDS SOIS) [8]. The primary objective of the CCSDS SOIS standard development activities is to radically improve the spacecraft flight segment data systems design and development process by defining generic services that will simplify the way flight software interacts with flight hardware and permitting interoperability and reusability both for the benefit of Agencies and Industrial contractors.

5 CONCLUSION

In this paper, the SpaceWire Test & Demonstration Platform 1.0 baseline has been discussed with view towards current and future SpaceWire related development activities. The features of the IPPM make it an ideal candidate for experimentation and validation of SpaceWire-enabled equipment, SpaceWire network topology and protocols as well as software applications for RTEMS aimed at conforming to the CCSDS SOIS recommendations and to the ECSS Packet Utilization Standard.

Future expansion of the basic set-up is foreseen to accommodate SpaceWire equipment based on the next generation of application specific standard products (ASSP) such as the SpW-RTC, which is an ideal candidate component for advanced instrument interfaces due to its Leon2-based architecture.

6 REFERENCES

1. AURELIA MICROELETTRONICA SRL, IPPM USER MANUAL, ESA CONTRACT 18780/04/NL/JA, 2008.

2. ALBERT FERRER FLORIT, MARTIN SUESS, "SPACEWIRE PLUG-AND-PLAY: FAULT-TOLERANT NETWORK MANAGEMENT FOR ARBITRARY NETWORK TOPOLOGIES", SPACEWIRE CONFERENCE, DUNDEE (UK), 2007

3. BARRY M COOK AND C PAUL H WALKER, **"**SPACEWIRE PLUG-AND-PLAY:AN EARLY IMPLEMENTATION AND LESSONS LEARNED", AIAA, CALIFORNIA, 2007

4. CHRISTOPHE HORNVAULT AND OLIVIER NOTEBAERT, "SPACEWIRE NETWORKS FOR PAYLOAD APPLICATIONS", SPACEWIRE CONFERENCE, DUNDEE, 2007

5. ECSS, "SPACEWIRE PROTOCOLS", ECSS-E-50-11 (DRAFT 0.8), 26 MAY 2008.

6. ECSS, "GROUND SYSTEMS AND OPERATIONS – TELEMETRY AND TELECOMMAND PACKET UTILISATION", ECSS-E-70-41A, 30 JANUARY 2003

7. UOD/S.PARKES, "SPW-RT INITIAL PROTOCOL DEFINITION-V1.1", SPACEWIRE WORKING GROUP MEETING, JUNE 2008.

8. CCSDS, "SPACE CRAFT ONBOARD INTERFACE SERVICES – INFORMATIONAL REPORT", CCSDS 850.0-G-1, JUNE 2007

9. P. MENDHAM S.MILLS AND S.PARKES, "NETWORK MANAGEMENT AND CONFIGURATION USING RMAP", SPACEWIRE CONFERENCE, DUNDEE, 2007

10. ESA/EDI SOFT, RTEMS CENTRE, **HTTP://RTEMSCENTRE.EDISOFT.PT**

SPACEWIRE APPLICATION FOR THE X-RAY MICROCALORIMETER INSTRUMENT ONBOARD THE ASTRO-H MISSION

Session : SpaceWire missions and applications

Short Paper

Yuichiro Ezoe[1], Yoshitak Ishisaki[1], Takaya Ohashi[1], Keisuke Shinozaki[2], Yoh Takei[2], Noriko Y. Yamasaki[2], Kazuhisa Mitsuda[2], Kosuke Sato[3], Ryuichi Fujimoto[3], Yukikatsu Terada[4], Makoto Tashiro[4], Richard L. Kelley[5], and Jan-Willem den Herder[6]

[1] *Tokyo Metropolitan University, 1-1 Minami-Osawa, Hachioji, Tokyo, Japan*

[2] *ISAS/JAXA, 3-1-1 Yoshinodai, Sagamihara, Kanagawa, Japan*

[3] *Kanazawa University, Kakuma-machi, Kanazawa, Ishikawa, Japan*

[4] *Saitama University, 255 Shimookubo, Sakura, Saitama, Japan*

[5] *NASA/GSFC, Greenbelt, MD 20771, USA*

[6] *SRON, Sorbonnelaan 2, 3584 CA, Utrecht., Netherland*

E-mail : ezoe@phys.metro-u.ac.jp, ishisaki@phys.metro-u.ac.jp, ohashi@phys.metrou.ac.jp, shinozaki@astro.isas.jaxa.jp, takei@astro.isas.jaxa.jp, yamasaki@astro.isas.jaxa.jp, mitsuda@astro.isas.jaxa.jp, ksato@astro.s.kanazawau.ac.jp, fujimoto@astro.s.kanazawa-u.ac.jp, terada@phy.saitama-u.ac.jp, tashiro@phy.saitama-u.ac.jp, Richard.L.Kelley@nasa.gov, J.W.A.den.Herder@sron.nl

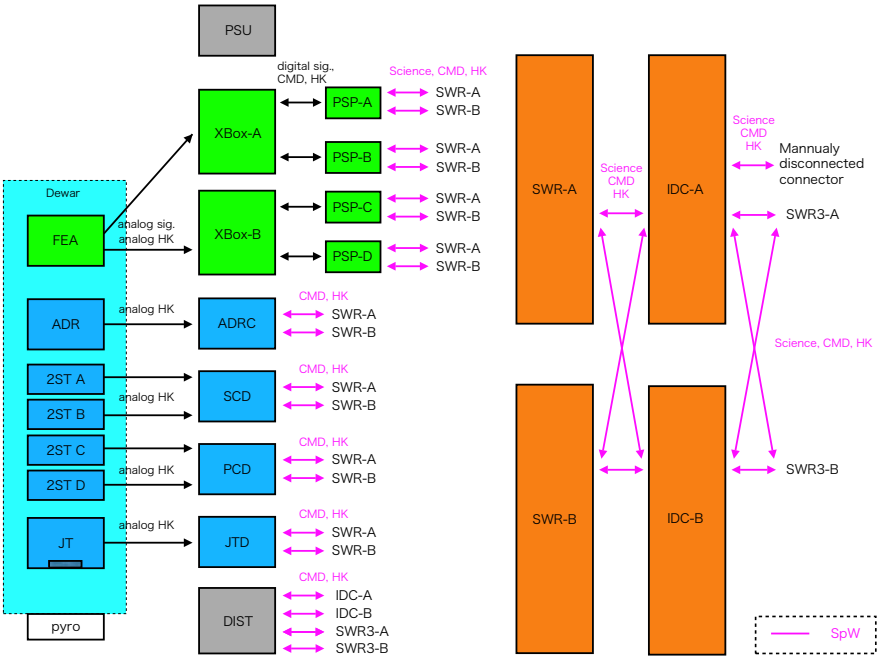
Abstract

SpaceWire application for the X-ray microcalorimeter instrument onboard the Japanese Astro-H mission (2013) is presented. The instrument, Astro-H SXS (Soft X-ray Spectrometer), will be the first satellite-borne microcalorimeter experiment, and will investigate the hot-gas dynamics in galaxies and in clusters of galaxies with >20 times better energy resolution of <7 eV (FWHM) at 5.9 keV than the previous experiments. The Astro-H SXS will make use of SpaceWire as high speed data and command interfaces. In this paper, we present the data network architecture and the development status.

1 ASTRO-H SXS

The X-ray microcalorimeter detects heat pulses by X-ray photons in X-ray absorbing pixels. By precisely measuring the temperature increase, >20 times better resolution spectroscopy at 5.9 keV than X-ray CCDs with high quantum efficiency becomes possible.

Astro-H is a Japanese medium size satellite (∼2600 kg) which will be launched into the low earth orbit in 2013 [1]. Its objective is a wide-band spectroscopy provided by multi-layer coating X-ray telescopes, semiconductor detectors, scintillation crystals and a microcalorimeter named SXS (Soft X-ray Spectrometer) [2]. The SXS is composed of the SXS-XCS (X-ray Calorimeter Spectrometer) and SXS-SXT (Soft X-ray Telescope).



IDC : UART used/total 0/2 ch, SpW used/total 6/6 ch (SWR x2, DIST, RUTR3 x2, MD)
SWR : SpW used/total 10/14 ch (IDC x2, PSP x4, ADRC, SCD, PCD, JTD)

Figure 1: The data block diagram of the Astro-H SXS-XCS.

2 DATA BLOCK DIAGRAM

In Astro-H, we will employ SpaceWire as a high-speed standard data link. The data block diagram of the Astro-H SXS-XCS is shown in figure 1. This diagram is designed so as to avoid a single point failure at the SpaceWire network such as cable connections, data handling units, and so on, as possible as we can. It also enables flexible ground operations without turning on the spacecraft and considers the power-on sequence. The diagram can be divided into four subsystems.

- Detector subsystem (green boxes in figure 1). X-ray pulse signals detected in the microcalorimeter array, which is within FEA (Front End Assembly) in the dewar, are sent to XBoxes (X-ray Box : Analog electronics and ADC converters) and then to PSPs (Pulse Shape Processor : Digital pulse analyzer). Each XBox (-A and -B) handles pulse signals from 32 of 64 pixels, while each PSP (-A, -B, -C, and -D) analyzes 16 pixel data.

- Cryogenic subsystem (blue). To cool the microcalorimeter down to 50 mK, we employ 2-stage Stirling coolers (2ST), a ^{3}He Joule-Thomson cooler (JT), superfluid He, and a 2-stage adiabatic demagnetization refrigerator (ADR). SCD (Shield Cooler Drive : two 2STs for dewar internal radiation shields), PCD (Pre Cooler Drive : two 2STs for JT precoolers), JTD (JT Cooler Drive), and ADRC (ADRC Controller) control the 2STs, JT, and ADR, respectively.

- Data handling subsystem (orange). IDC (Instrument digital control) is a command and telemetry interface with the spacecraft system via two Space Wire routers (SWR3-A and

-B). IDC-B and SWR-B are cold redundant units. IDC has functions to check cooler temperatures, to control the JT start up, and to communicate with the ground support equipment (GSE) during the ground operations. SWR (SpaceWire Router) is a router to distribute commands and receive data from units. IDC generates CCSDS packets and sends the data to the spacecraft system. The expected data rate between IDC and the spacecraft is < 20 kbps for the science data and $\sim$1.5 kbps for the HK data. The GSE can be directly connected to IDC-A via SpaceWire for ground operations.

- Power subsystem (grey). DIST (Distributor) distributes the spacecraft bus power of 32–50 V to all the SXS-XCS units. It can be controlled by either the spacecraft system or IDC. Since DIST can be powered from the GSE and controlled by IDC, we can operate the SXC-XCS without turning on the spacecraft bus on ground. PSU (Power Supply Unit) recieves the primary bus power from DIST and provides regulated power, isolated from the primary power, to XBoxes. It has no data interface.

3 DEVELOPMENT OF SPACEWIRE UNITS

3.1 IDC and SWR

Current designs of the two data handling units, IDC and SWR, are shown in table 1. They will be manufactured by NEC Corporation. The design of IDC is based on Space Cube 2 which will be onboard the JAXA SDS-1 small satellite to be launched in 2009, while SWR is newly developed. Both of them will be universal SpaceWire units for future satellites. In fact, the hardware of IDC will be the same as that of the spacecraft system unit onboard Astro-H.

IDC will have a high-speed radiation-hard CPU named HR5000 (64 bit, 200 MHz at maximum), moderate-size system memories (2 MB EEPROM, 4 MB Burst SRAM, and 4 MB Asynchronous SRAM), and large data recorder memories (1 GB Asynchronous SDRAM and 1 GB Flash Memory). The TRON realtime OS, T-Kernel, will be employed as an operating system. IDC will be equipped with only two modes (ON or OFF). When it is powered on, the IDC onboard software automatically starts up, and begins command and telemetry handling.

Table 1: Current designs of IDC and SWR.

	IDC	SWR
Size	W71×D221×H180 mm	W150×D110×H60 mm
Mass	2 kg	1.2 kg
Power	OFF 0 W, ON 14±1 W	OFF 0 W, ON 5±1 W
SpaceWire port	6 port/Router	14 port/Router
- link rate	50 Mbps	50 Mbps
RS422/UART port	2 port	–
- link rate	19200 bps	–

While SWR is a simple router using the cross bar switching technology, IDC has various functions as described in section 2. To fulfill the requirements, the mission specific software must be developed. To distinguish general tasks such as SpaceWire RMAP I/O and command/telemetry handler from the mission specific tasks such as CCSDS packet forming and ADR control, the JAXA standard middleware will be defined and prepared by the Japanese SpaceWire team. Development of the middleware, especially the SpaceWire API, is ongoing.

IDC will make use of the SpaceWire time code which is defined in the SpaceWire protocol. Its delay and/or jittering within the Astro-H system will be less than a few μs. Referring to GPS signals or an internal clock circuit if the former is unavailable, a spacecraft system unit will distribute the time code (6 bit, 1/64-sec time resolution) to IDC.

3.2 PSP

PSP will be composed of four Universal SpaceWire Boards and Space Cards, both of which are developed by Mitsubishi Heavy Industries Ltd.. One pair of them is used to handle the data from 16 of 64 pixel data. The Universal SpaceWire Board will have two FPGAs with 32 MB SDRAM. In this board, X-ray pulses are detected and pulse shapes are recorded. The Space Card will have an SH4 compatible CPU with 64 MB SDRAM and FPGA with 2 MB EEPROM, 4 MB SRAM, and 32 MB SDRAM. Intelligent data analysis such as second pulse detection, pulse shape filtering, and packet generation is conducted in this board. Development of the pulse analysis software for ground microcalorimeter application and the SXS-XCS is described in [3]. Similar to IDC and SWR, these two types of boards are universal modules and will be used in other instruments onboard Astro-H.

3.3 The other units

SpaceWire interfaces of the other electrical units in figure 1 will be implemented by the SpaceWire IP Core for FPGA (Field Programmable Gate Arrays). One FPGA will support two SpaceWire ports. The type of FPGA is to be determined. A candidate for the data and command transfer between IDC and each unit is to RMAP write from IDC and then to RMAP read by IDC. Thus IDC controls timings of command and telemetry.

4 CONCLUSION

Development of SpaceWire hardware and software for the Astro-H SXS-XCS is being carried out. Based on heritages of coming SpaceWire missions before Astro-H, e.g., SDS-1 and the JAXA small satellite project, we would like to establish the SpaceWire system for the SXS-XCS.

Reference

[1] T. Takahashi et al., "The NeXT mission", 2008, SPIE, 7011, 70110O

[2] K. Mitsuda et al., "The x-ray microcalorimeter on the NeXT mission", 2008, SPIE, 7011, 70112K

[3] T. Hagihara et al., "Digital signal processing systems of an X-ray microcalorimeter array for ground and space applications", this conference

DIGITAL SIGNAL PROCESSING SYSTEMS OF AN X-RAY MICROCALORIMETER ARRAY FOR GROUND AND SPACE APPLICATIONS

Session : missions and applications

Short Paper

T.Hagihara, K.Mitsuda, N.Y.Yamasaki, Y.Takei, H.Odaka

ISAS/JAXA

M.Nomachi

Osaka University

T.Yuasa

University of Tokyo

E-mail: hagihara@astro.isas.jaxa.jp, mitsuda@astro.isas.jaxa.jp, yamasaki@astro.isas.jaxa.jp, takei@astro.isas.jaxa.jp, odaka@astro.isas.jaxa.jp, nomachi@lns.sci.osaka-u.ac.jp, yuasa@amalthea.phys.s.u-tokyo.ac.jp

ABSTRACT

We are developing a high resolution EDS (Energy Dispersive Spectrometer) for a Transmission Electron Microscope (TEM). The EDS utilizes an array of TES (Transition Edge Sensor) micro-calorimeter, which operates at 100 mK and achieves an extremely high energy resolution of FWHM < 10eV in 0.1 to 10keV energy range. The analog signals from the calorimeter array are continuously digitized. Then X-ray events are detected by a digital logic in an FPGA and the blocks of data are buffered in it. Then the data blocks are transferred to a CPU via SpaceWire. The CPU performs optimum digital filtering to determine the pulse height.

We assume a high count-rate condition, such as 200 counts/s/pixel, and data transfer rate between the FPGA and the CPU is estimated to be 25 Mbps, corresponding to a 800 counts/s (4 pixels per FPGA). SpaceWire can accommodate this high-speed transfer rate, and this is why we chose standardized SpaceWire devices for this system.

All components were connected and system performances are being evaluated. At present, data transfer speed via SpaceWire is ~ 1Mbps and energy resolution using ideal waveforms is ~300 (E/ΔE). We have been improving software and logics. We will combine this data acquisition system with a TES micro-calorimeter array this year.

1 INTRODUCTION

Micro-calorimeter is an X-ray EDS which measures the small temperature increase induced by X-ray with high-sensitive thermometer and achieves very high energy resolution (E /Δ E > 1000). TES micro-calorimeter use TES as a thermometer. To reduce thermal fluctuation, micro-calorimeter is operated at ~100mK. In order to realize such potential energy resolution, a digital waveform processing system in

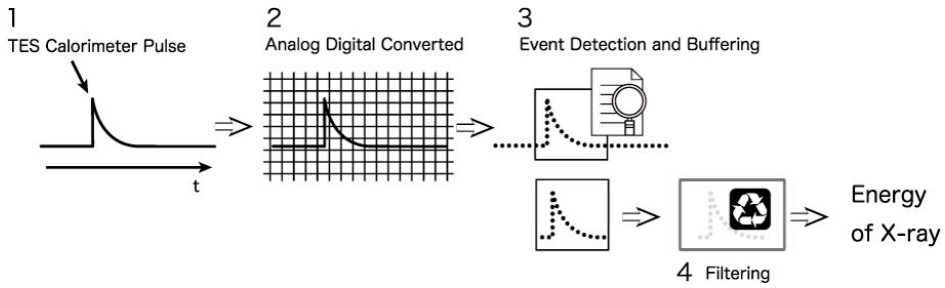
addition to low noise read-out circuits is indispensable. So-called “optimal filtering” to maximize signal-to-noise ratio of a thermal equilibrium calorimeter is realized only by usage of the total wave-form, or wide-band pulse spectrum in the frequency domain[1,3]. It requests a very fast waveform digitizing system with an enough accuracy and multi-input channels (typically ADC vertical resolution >10bit, horizontal resolution >10kHz).

And now, high count-rate adapted calorimeter system is required both in space missions and ground applications. Future X-ray space missions like IXO, which will have collecting area as large as 5 m [2], and X-ray spectroscopic detectors on the ground will request high count rates. As the optimal filtering method assume that each pulse has similar shape to the ideal one, a pulse with a secondary one (pileup events) should be treated with special attention. Discard of pile-up events as dead-time events will reduce the appeals of calorimeters.

In TEM, accelerated electrons (~10keV) interact with target materials and fluorescence X- rays are emitted. Using TES calorimeter as an X-ray detector, surface structure can be scrutinized and perhaps chemical shift (energy shift by chemical bound: ~1eV) can be observed.

2 SYSTEM DESIGN

2.1 EVENT PROCESSING FLOW



Event processing flow can be summarized in following steps;

Step 1: Irradiated energy into a TES calorimeter is converted to the heat and an analog read-out electronics produces a pulse.
Step 2: The output signal is continuously sampled by a fast ADC (analog-to-digital converter).
Step 3: Sampled waveform is scanned to find the pulse in FPGA. When a pulse is detected, a trigger is sent to store the data with enough length to cover the whole waveform.
Step 4: The stored waveform is transmitted one by one via SpaceWire, and a CPU calculated the pulse height by the optimal filtering method.

In step 2, record length, sampling accuracy and speed should match with the requirements from the energy resolution. Clever trigger logic to detect double pulses and buffer handling in step 3 is the key issue to achieve the high count-rate performance. See reference[5] to know more details about these logics.

2.2 SYSTEM REQUIREMENTS

We assumed that 10 pixels of a TES array with a count rate of 200 counts /s/pixel as a spectrometer in a TEM. Energy band is from 0.3keV to 10keV and required energy resolution is 5eV in all range. To achieve required energy resolution, a total waveform (event) becomes 4k Byte (16bit vertical resolution for data handling and length of 2048 samples) and total count rate reaches 2000 counts/s or 8M Byte/s. In order to process such large data in real time, we divided processing flow as shown in 2.1.

2.3 DEVICES

A project to develop common hardware accommodate SpaceWire and data acquisition system has been promoted in Japan for NeXT and other missions by JAXA, Osaka Univ., Univ. of Tokyo and other groups[6]. Digital I/O (DIO) board, one of the outcome of this project, is used in this TES system. It has various data port and a FPGA (Xilinx Spartan-3 XC3S400FTG256) which users can install their logic to handle their data. We programmed the trigger and buffer handling logic, and the control of an ADC board in this FPGA. ADC boards of 4 channels, 14-bit, and 1 MS/s are fabricated for this application. Four channels of waveform data are continuously sent from the ADC board to the DI/O, and the DI/O detects the events and buffered data are sent via SpaceWire to a CPU called “SpaceCube”. SpaceCube is a small TRON-based computer, developed as a platform of SpaceWire verification by ISAS/JAXA and Shimafuji Electronics. Optimal filtering program including the production of the template waveform by FFT, and the calculation of the pulse height by the cross correlation is transported already.

3 EVALUATION OF THE SYSTEM AND FUTURE APPLICATIONS

3.1 SYSTEM ASSEMBLING

The evaluation tests have been started with simulated waveform of TES pulses from a 14-bit function generator. The 14-bit ADC runs in a sampling rate of 1.2 MS/s. The handshaking between ADC and DI/O boards, DI/O and SpaceCube work correctly, and the triggered waveforms are collected by the SpaceCube. We confirmed trigger and buffer handling logic, producing several patterns of waveforms which included pile-up events, and input to the system.

3.2 EVALUATION OF THE PROCESSING SPEED

We evaluated data acquisition speed of event buffering logics in FPGA inputting ideal waveform at various frequencies using function generator. As a result, the event buffering logics in FPGA work well and can collect all input pulses at 1kHz. This is enough for system requirement. We also evaluated event transfer speed between I/O board and SpC via SpaceWire. In this process events were stored in the memory onboard SpC. Event transfer takes 28ms per event (34 events/s, ~1Mbps). If we upgrade the SpaceWire FPGA IP-core this speed will be drastically improved.

3.3 OPTIMAL FILTERING APPLIED TO THE IDEAL PULSES

To assume energy resolution, we input ideal waveform and applied total process to these pulses. In SpC, averaged pulse (figure 3-1) and noise spectrum (figure 3-2) were

calculated using collected events and filtering template (figure 3-3) was created using these averaged pulse and noise spectrum. The pha spectrum (figure 3-4) was obtained adapting optimal filtering method to ideal waveforms using this template. We fitted this spectrum with gaussian and get the result that E/ΔE is 300(we assumed gaussian FWHM as ΔE and gaussian center as E). This degradation of the energy resolution will be induced from noise generated in function generator and algorithm used in our optimal filtering method. The cause of the low energy tail seen in figure 1 still be unknown. We have been refining software and logics.

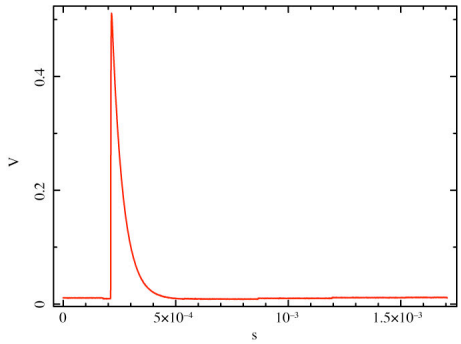


figure 3-1. Averaged pulse.

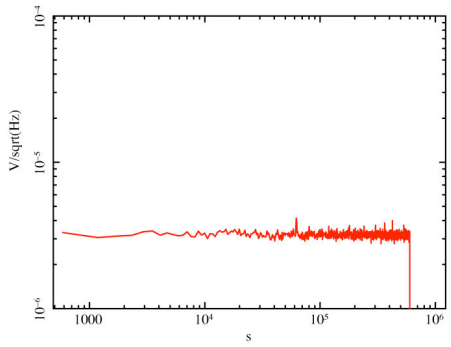


figure 3-2. Noise spectrum.

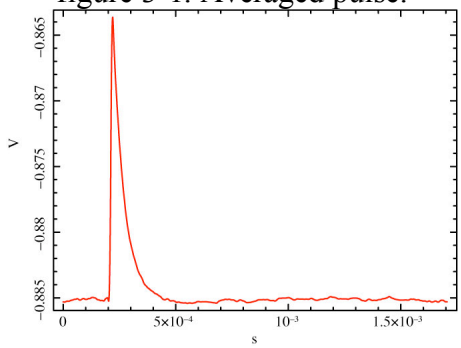


figure 3-3. Filtering template.

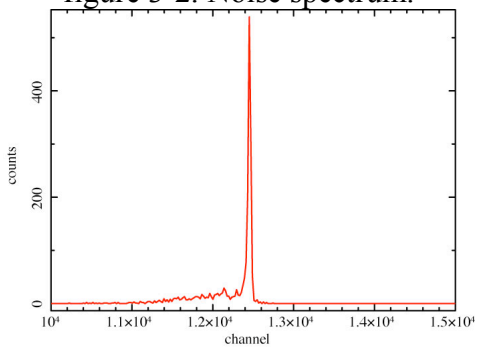


figure 3-4. PHA spectrum.

4 REFERENCE

[1] D. J. Fixen, S. H., Moseley, B. Cabrerra, E. Figueroa-Feliciano, AIPC, PRoc., 605, 339F, (2002).

[2] R. L. Kelly et al., Publ. Astson. Soc. Jap., 59, 77 (2007).

[3] D. MaCammon, Topics in Applies Phys., 99, 1(2005).

[4] K. Mitsuda et al., Journal of Low Temperature Physics, Volume 151 3/4, pp 703(2008)

[5] T. Hagihara et al., Journal of Low Temperature Physics, Volume 151 3/4, pp 997(2008)

[6] T. Yuasa et al., A Portable SpaceWire/RMAP Class Library for Scientific Detector Read Out Systems, International SpaceWire Conference 2008

SYSTEM ASPECTS OF SPACEWIRE NETWORKS

Session: SpaceWire missions and applications

Short Paper

Paul Rastetter, Dr. Stephan Fischer, Uwe Liebstückel, Richard Wiest

Astrium GmbH, 81663 Munich, Germany

E-mail: paul.rastetter@astrium.eads.net, stephan.fischer@astrium.eads.net, uwe.liebstueckel@astrium.eads.net, richard.wiest@astrium.eads.net

1 ABSTRACT

The well established SpaceWire standard provides a point-to-point communication with hundreds of Mega-bit per second data transfer in full-duplex operation. To efficiently use SpaceWire devices in an on-board system many aspects have to be taken into account already early in the system definition phase. Among others the following points have to be considered: provided data rates, supported protocols, provided interface to configure and control the SpaceWire devices, interaction with software, compatibility of equipments from different suppliers assuming master/slave concepts, etc..

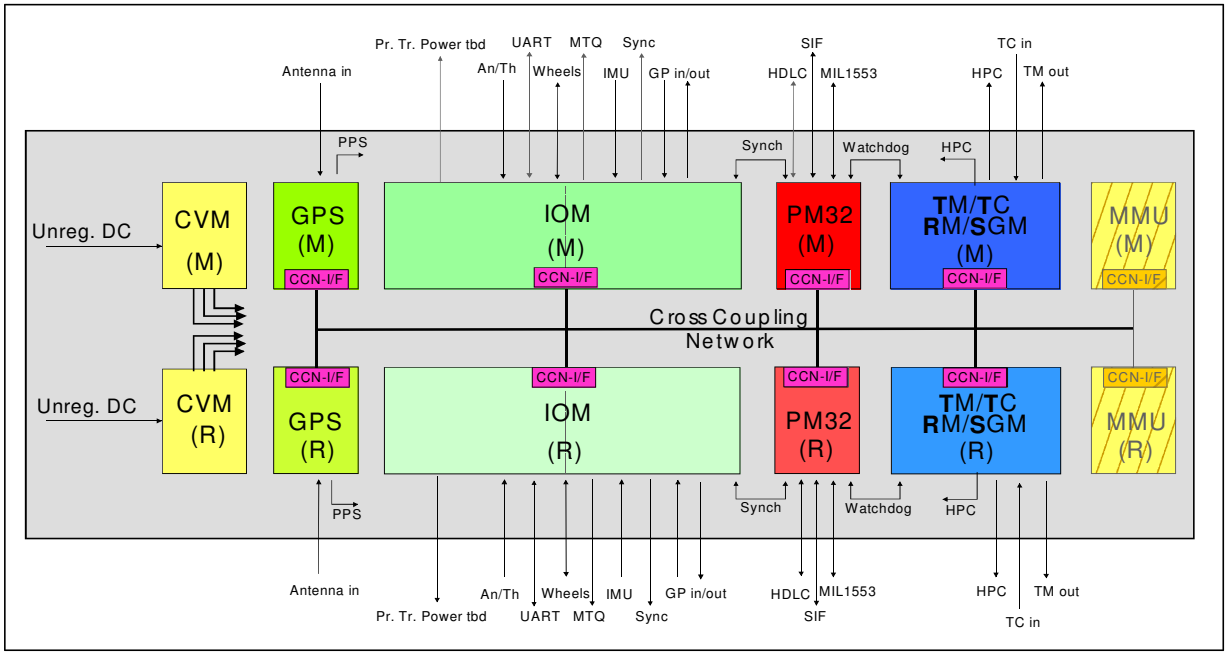
The paper will give an overview about some system concepts and their implementation using e.g. the SMCS332SpW, the SMCS116SpW and special FPGA implementations including a SpaceWire interface in a typical topology.

2 SYSTEM ASPECTS

In the definition phase of a SpaceWire based system several aspects have to be considered. One aspect is the proposed or required net work topology. Another is the selection of available devices to implement the required functions and interfaces.

The topology shown in the diagram, called "Cross Coupling Network", is a star topology. Star topology means that one module is connected directly to all other modules. In the diagram the module PM32 is the one which is connected to all other modules. The other modules like IOM, TM/TC or GPS have one main SpW link and one redundant SpW link. In a star topology each SpW link can operate on a different transmit rate depending on the necessarity of the connected module.

The aspects of interfacing between software and SpaceWire device, hardware-software interacting, protocol support and remote controlled SpaceWire devices will be discussed in the following chapters.



2.1 Interfacing between software and SpaceWire-device (on "intelligent" modules)

SpaceWire devices are often used to transfer data between modules. The data interfacing can be done in several ways which has different impacts on software. One way is to use a simple FIFO-like interface and another way is to implement DMA (Direct Memory Access). The FIFO-like interfaces require less or even no external hardware but put the load of transfer to software. Each byte/word (depending on interface width) has to be read/written by a software task, which limits the throughput.

Another way is to implement DMA capability into the SpW-device, but DMA requires memory. The memory can be on the board or inside the SpW-device itself. The data transfer is done in the following way. The processor writes the complete transmit SpW packet into the memory. After that the processor sets the transmit pointers inside the SpW device. From now on the device takes control of transmitting the SpW packet without any further interaction from the processor. The maximum length of one SpW packet depends on the implementation, e.g. length of pointers or additional control bits.

The receipt of a SpW packet with DMA works similar. The processor sets the receive pointers which define a receive area in the memory and it is informed by interrupt when a packet is received. The length of a receive packet is defined by the sender.

DMA itself is more complex in hardware, but offers high throughput and simple interface to software. SMCS332SpW and SMCS116SpW (in master mode) provide DMA capability.

2.2 Hardware-Software interacting

The interacting between hardware and software is important for the system and is done via interrupts and/or status bits. Therefore the interrupt handling is a major task in software. To

simplify the life for SW, a SpW device should support several interrupt/events/status bits. The interrupts should be maskable to allow a selection of bits of interest at the current time.

In the case of a FIFO-like interface the interrupts could be e.g. link connected, link error, byte, word transmitted/received. Using DMA the interrupts could be e.g. link connected, link error, packet transmitted, EOP/EEP received, receive area full. The advantage of the DMA is the fact that the interrupts can be issued on packet level instead of byte/word level. This reduces the interrupt frequency in comparison with a FIFO-like interface.

SMCS332SpW and SMCS116SpW provide all necessary interrupts to signal the events to software at packet level.

2.3 PROTOCOL SUPPORT

There are several protocols available on top of SpW. All protocols are characterized by a header part and a data part. The length of the header part may be not always aligned with 32 bits, which is mostly used by processors. Therefore an easy way to assemble the header part and data part to a SpW packet at the transmitting side of a SpW-link is necessary in DMA mode.

The SMCS332SpW supports the protocol packet generation by the header field bit. The bytes to be transmitted via SpW are always read in 32 bit mode. If the header field bit is set, the value of the first byte which is read from memory is used as a counter value. This value defines the amount of header bytes to be sent. The counter byte itself is not transmitted via SpW. This mechanism allows the software to generate headers with a length from 1 to 15 bytes. The complete length of a SpW transmit packet is defined by the transmit pointers.

The same mechanism could also be used if path address bytes have to be put in front of a SpW packet.

2.4 REMOTE CONTROLLED SPACEWIRE DEVICES (WITHOUT LOCAL PROCESSOR)

A remote controlled SpW device is characterized by containing a SpW protocol and several functions/interfaces like event counters, timers, RAM- or FIFO-interfaces etc. which are controlled and configured via a defined protocol. The protocol is defined by the remote SpW device itself. It can be an open/general protocol like RMAP or STUP or a propriety one.

3 SUMMARY

As discussed in the paper the performance of a system depends on several issues. There are several SpW devices available, like SpW-Router, RTC (Remote Terminal Controller), SMCS332SpW and SMCS116SpW, which can be used to design and implement a SpW-based system.

If there are specific requirements which can not be fulfilled by these existing devices, the SpW-Codec from ESA can be included in a FPGA together with the specific requirements.

Poster Presentations

UPCOMING FLIGHT-WORTHY SPACEWIRE COMPONENTS

Session: SpaceWire Components

Short Paper

Sandi Habinc, Jiri Gaisler

Gaisler Research, Första Långgatan 19, SE-413 27 Göteborg, Sweden

E-mail: sandi@gaisler.com, jiri@gaisler.com

ABSTRACT

As with any new technology used in space, it is crucial that components are available that implement it before anyone will consider using the technology on a broader scale. The SpaceWire technology is no exception. It is therefore important that flight-worthy components are made available to the general user to promote SpaceWire technology.

1 INTRODUCTION

This paper address this issue, presenting a number of upcoming flight-worthy components not only implementing SpaceWire links [1], but also taking the concept further with the implementation of the Remote Memory Access Protocol (RMAP) [2]. A variety of components have been based on the Gaisler Research SpaceWire IP core [4] that has been developed specifically for advanced system-on-a-chip products harbouring on-chip SPARC processors [5].

2 UT699 LEON3FT

The UT699 device [7] from Aeroflex implements a LEON3-FT processor with four SpaceWire interfaces and a variety of control interfaces such as CAN bus, Ethernet and PCI. The integrated multi-protocol SpaceWire core supports RMAP, GAP and GRDDP. The device and prototyping boards are available now from Aeroflex.

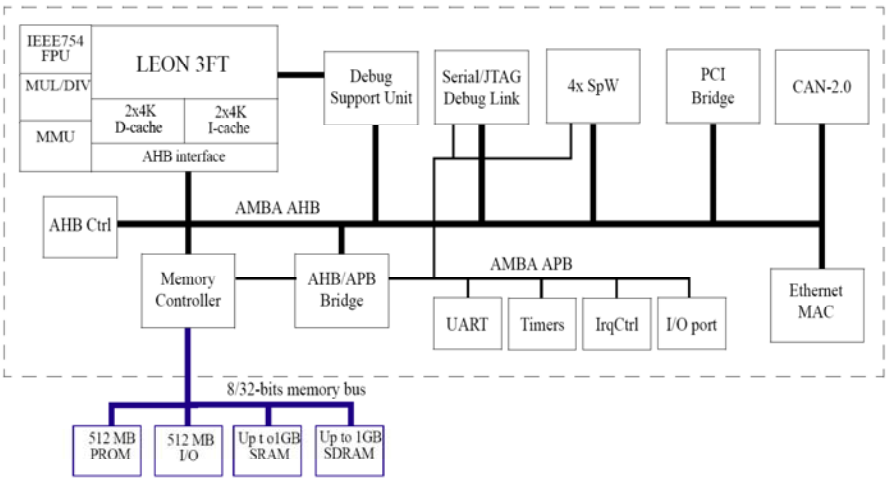


Figure: UT699 block diagram

3 COMPANION DEVICES

For processors not implementing SpaceWire interfaces directly, but which implement the PCI (Peripheral Component Interconnect) interface, for example the AT697F device from Atmel, the GR701A companion chip has been conceived based on the Actel RTAX FPGA technology. The GR701A companion chip provides the processor multiple SpaceWire, CAN and MIL-STD-1553B interfaces. This brings SpaceWire capability to the any processor with PCI interfaces such as PowerPC.

Next generation companion chips are already being planned, featuring for example CCSDS Telemetry Encoder and Telecommand Decoder, I2C Master/Slave, SPI, etc.

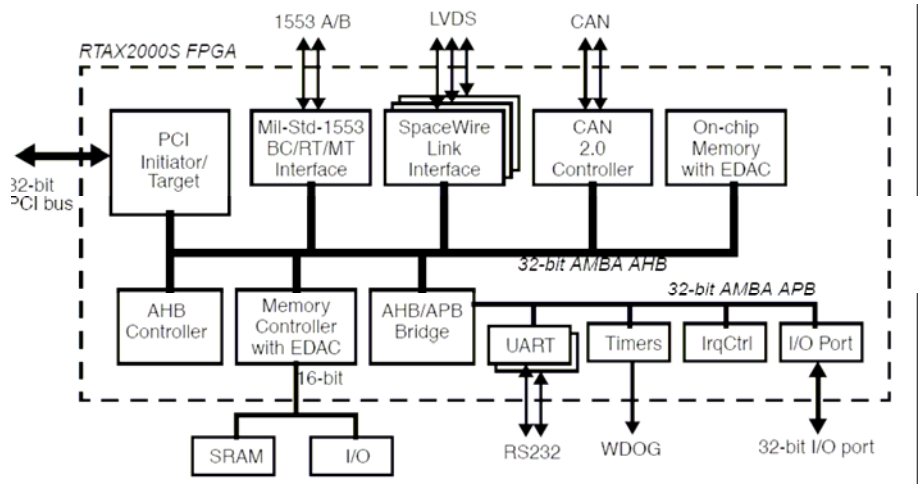


Figure: GR701A companion chip block diagram

4 LEON3FT-RTAX

Flight products shipped to customers today are from the LEON3FT-RTAX family. This is an implementation of the LEON3-FT SPARC V8 32-bit processor using the Actel RTAX FPGA technology.

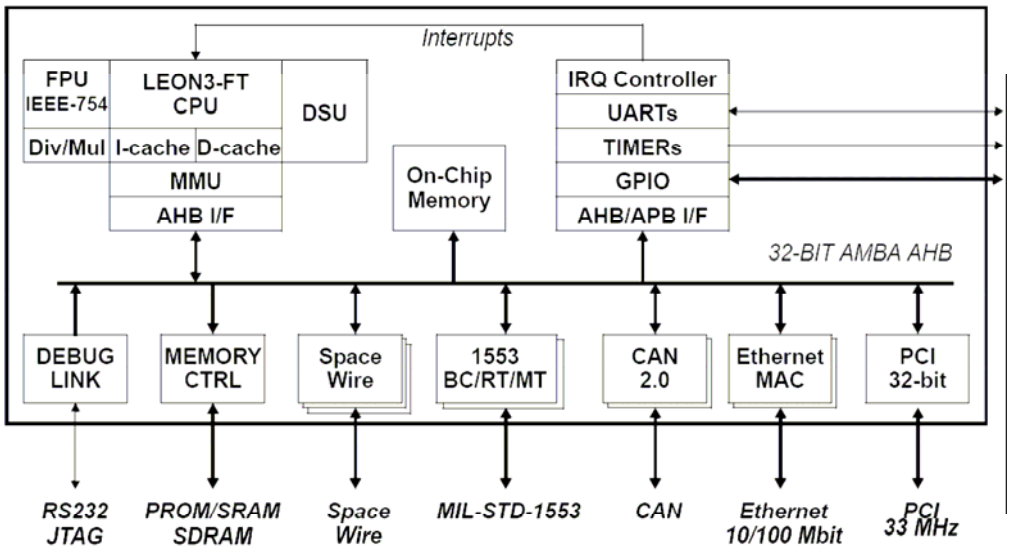


Figure: LEON3FT-RTAX block diagram

The LEON3FT-RTAX processor is provided in multiple configurations, covering both instrument and spacecraft control applications. The configurations offer up to three SpaceWire links with RMAP support, as shown in the table below.

Configuration	CID1 (IC1)	CID2 (IC2)	CID3 (SC1)	CID4 (SC2)	CID5 (SC3)	CID6 (SC4)	CID7 (PC1)	CID8 (PC2)
LEON3FT Integer Unit	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hardware multiply & divide					Yes	Yes		Yes
Multiply & accumulate								
Single-vector trapping	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Power down mode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Memory Management Unit					Yes	Yes		
Floating Point Unit	Yes	Yes	Yes				Yes	Yes
Debug Support Unit	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
UART Debug Link	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
JTAG Debug Link								
On-Chip Memory	4 kbyte		2 kbyte					
1553 RT	1							
1553 BC/RT/MT			2					
SpaceWire		2		3	2		2	
CAN 2.0B	1				1			
PCI Initiator/Target						Yes		
PCI Arbiter						Yes		
Ethernet MAC 10/100 Mbit						1		2
Memory Controller	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SDRAM Support				Yes	Yes	Yes	Yes	Yes
Standard peripherals	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Additional 16-bit							Yes	
Package	CQFP352	CQFP 352	CQFP 352	CCGA 624	CCGA 624	CCGA 624	CQFP 352	CQFP 352

Table: Overview of LEON3FT-RTAX Actel RTAX2000S devices

5 PROTOTYPE DEVELOPMENTS

The first ASIC silicon with integrated SPARC processor and SpaceWire interface was the GR702RC radiation-hard test device, which has been manufactured on the 180 nm technology to assess performance, design flow and radiation behavior of cores using the Ramon Chips' radiation tolerant cell library. The ASIC has been successfully validated and undergone radiation testing. The follow-on GR712RC radiation-hard device will include a dual-core processor system with multiple SpaceWire interfaces.

In parallel, the LEON3-FT processor and SpaceWire interface are being used in the design driver named LEON-DARE that is used in an ESA activity [8] to validate and qualify the Design Against Radiation Effects (DARE) library, developed by IMEC under ESA founding. The device will comprise the LEON3-FT processor, a high-performance floating-point unit, a memory management unit and multiple SpaceWire interfaces with full RMAP support.

6 CONCLUSIONS

All the above components are being delivered or being developed with the objective to provide the general user, without any limitations, with flight-worthy components implementing the SpaceWire technology. The rich variety of components and producers ensures that there is always a component suitable for the specific task at hand to be solved.

The affluent spectrum of components based on space industry standard technology such as the SPARC V8 processor and the SpaceWire interface ensures availability of compatible although competing flight-worthy components. This should make the usage of SpaceWire technology an easy choice.

7 REFERENCES

1. "SpaceWire standard - Links, nodes, routers and networks", ECSS-E-ST-50-12C, 31 July 2008

2. "SpaceWire protocols", ECSS-E-ST-50-11C, July 2008

3. "GRLIB IP Library User's Manual", Gaisler Research, www.aeroflex.com/gaisler

4. "GRLIB IP Core User's Manual", Gaisler Research, www.aeroflex.com/gaisler

5. "The SPARC Architecture Manual", Version 8, Revision SAV080SI9308, SPARC International Inc.

6. "LEON3-FT SPARC V8 Processor - LEON3FT-RTAX - Data Sheet and User's Manual", Gaisler Research, http://www.aeroflex.com/gaisler/doc/leon3ft-rtax-ag.pdf

7. "UT699 LEON 3FT/SPARC V8 Microprocessor", Aeroflex Colorado Springs, http://ams.aeroflex.com/ProductFiles/DataSheets/LEON/UT699leon.PDF

8. ESA Contract Number 19916/06/NL/JD

The Design and Performance of SpaceWire Router-network using CSP

Session:Components

Short Paper

Kazuto Tanaka, Satoshi Iwanami, Takeshi Yamakawa, Chikara Fukunaga*
Tokyo Metropolitan University, Hachioji, 192-0397 Japan

Kazuto Matsui
Prominent Network Inc., Tokyo, 104-0032 Japan

Takashi Yoshida
Smartscape Inc., Tokyo 151-0051 Japan

Abstract

We have designed an IEEE1355 network router as an Intellectual Property (IP) core. The basic idea of the router design has been evaluated, refined and verified from the point of view of robustness and security using CSP (Communication Sequential Processes) method, one of formal design methods [1]. Functionality of the router has been confirmed by implementing it in a network formed with several TPCOREs. TPCORE[2] is our homemade processor that can execute the same instruction set as transputer. Since all the network components have been implemented in a single FPGA chip, we realized Network on Chip (NoC). In this report we discuss the functionality of the router, modification of a TPCORE for IEEE1355 and performance of the network.

1 TPCORE

TPCORE has been developed by the authors' group a few years ago to make a simple parallel network system without introducing any operating system. We have known that we implemented such a system if we could use a transputer, which has been developed by an English company called Inmos Limited in 1980s and used worldwide extensively in that days. Since a parallel processing language occam[1] is closely related to the transputer architecture, programs written in occam were only able to run in a transputer or a transputer network. A transputer has

*Correspondong author:fukunaga@tmu.ac.jp

[1]Occam has been originally developed by Inmos Limited inspired by CSP[3]. Re-

four external serial link interfaces. This interface is called as os-link conveniently[2]. Connecting an os-link port with one of another transputer, we can form a network with several transputers in a plug-and-play manner. TPCORE has been developed as an IP core to execute fairly all the instruction set of the transputer while the inner architecture was different entirely. It can, therefore, run a program written in Occam, and make a network as the transputer since a TPCORE has also four os-link inerfaces. The clock frequency of TPCORE when it is implemented in an FPGA of Xilinx virtex4Lx160[5] is 24 MHz. If we form a TPCORE network with os-link in this chip, a mesh type connection with maximum 16 (4×4) TPCOREs can be implemented. The os-link is also operated in 24 MHz.

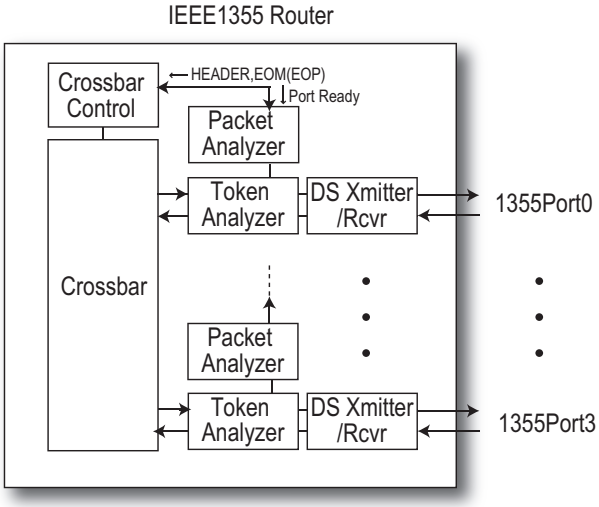


Figure 1: Block diagragm of presently developed IEEE1355 Router

2 IEEE1355 Network-Router with TPCOREs

Although the router currently developed has been based on IEEE1355 (hereafter this is also denoted as DS) standard, but SpaceWire, we expect that the router is adapted as a SpaceWire router with minor modification from similarity of these two standards. Block diagram of the router is shown in fig. 1. It is operated in 48 MHz, all the router operation is also synchronized with this clock. The number of DS ports is four, these ports are fully compliant with the DS protocol with bi-directional and high speed(48 M bit/sec.) transfer communication. Since a cascade link of a router to another one is possible, the network can be made further complex even if the number of the ports is only four. A non-blocking and wormhole crossbar switch enables a packet arriving at any port within two clocks

cently its complier which can be run under multi-thread environment of Linux has been developed[4].

[2]The os-link is a bit-serial protocol with unit of byte. For a byte transfer, sender sends it with 2-bit start- and 1-bit end-bit. Then receiver returns two bit acknowledge.

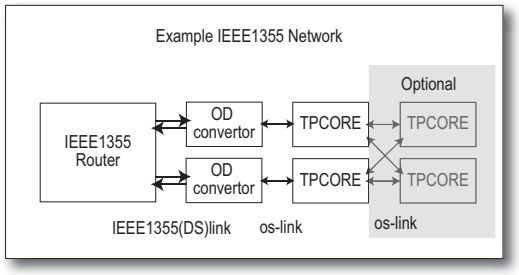


Figure 2: The example network setup with two (four) TPCOREs

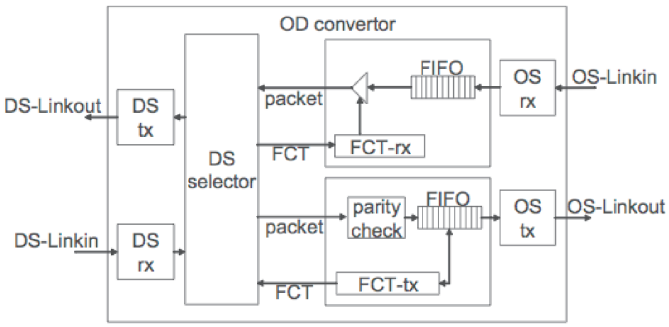


Figure 3: Block diagram of a bi-directional OD convertor (IEEE1355 $\leftrightharpoons$ os-link)

to relay over the other port of which the number is specified in the header part of the packet. Every DS port consists of DS transmitter/receiver, Token analyzer and Packet Analyzer. Token Analyzer picks up the parity bit, and checks it with parity actually observed in the previous token, it also transmits FCT (Flow Control Token) if there is a room to accept further tokens, or transmit tokens if FCT is received. Packet Analyzer picks up the destination port, which is written in the header part of a packet, and tells the Token Analyzer to start data transfer if the port is opened, and try to close the port when it receives EOP or EOM.

Figure 2 shows a network setup for the router functionality test. Since a TPCORE can communicate with each other using os-link, we have to implement an extra circuit to convert os-link to DS-link. The OD convertor does not only convert the data protocol to both direction (os $\leftrightharpoons$ DS-link) but also issue/receive FCT (Flow Control Token) signals. In fig. 4 an example of the signal sequence in OD convertor is demonstrated in which one byte data of 00111111_2 is formated into os-link, then converted to DS-link, and reverted to os-link. In fig. 5 the time sequence to transmit FCT is demonstrated. It is needed to send total 606 bits on DS-link for 32 byte data transfer. We measured this duration as 12.616μs while theoretical expectation is 12.625μs. The difference is less than the clock width(0.02μs).

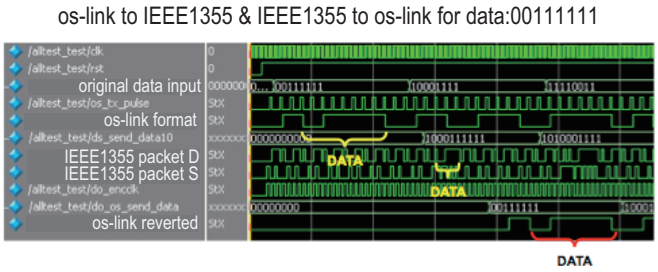


Figure 4: Time sequence of OD convertor. From a TPCORE, data:00111111_2 in os-link format is inputted, and converted to IEEE1355 protocol in an OD convertor, and this chart shows also data reverted to os-link format.

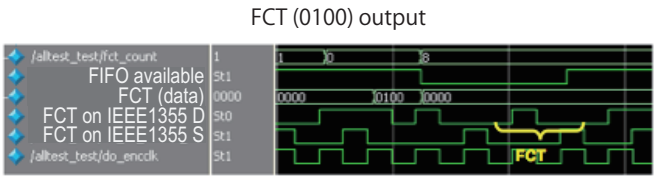


Figure 5: Timing chart to demonstrate the functionality of FCT transmission. If output FIFO for DS to os-link conversion part of an OD convertor has a room to accept data (FIFO available high), then FCT signal (bit sequence 0100_2 is transmitted over DS-link to prompt the data transmission side to send further eight tokens.

References

[1] K.Tanaka et al., "Proposal of CSP based Network Design and Construction", presented in this conference

[2] M.Tanaka et al., "Design of a Transputer Core and its implementation in an FPGA", Proceedings of Communication Process Architecture 2004 (CPA2004) held in Oxford, England, pp.361-372, IOS press.

[3] C.A.R.Hoare, "Communication Sequential Processes", Prentice-Hall Inc., 1985

[4] P.H.Welch and D.C.Wood, "The Kent Retargetable Occam Compiler", Proceedinga of Communication Process Architecture 1996 (CPA1996) held in Amsterdam, The Netherland, pp.143-166, IOS press.

[5] Xilinx, inc. Virtex 4: http://www.xilinx.com/products/virtex4/index.htm

HIGH SPEED MULTIPLEXING FOR SPACE WIRE INTERCONNECT

Session: Components

Short paper

Vladimir Katzman, Vladimir Bratov, Sean Woyciehowsky, Jeb Binkley

ADSANTEC Inc. 27 Via Porto Grande, Rancho Palos Verdes, CA 90275, USA

Glenn P. Rakow

NASA Goddard Space Flight Center, Greenbelt, MD 20771, USA

vkatzman@adsantec.net vbratov@adsantec.net woycieho@adsantec.net
jbinkley@adsantec.net glenn.p.rakow@nasa.gov

ABSTRACT

Future high performance computing in orbit requires new data interconnects operating at multi-gigabit data rates while preserving the open system Plug-and-Play architecture. Key components for these data links are high-speed multiplexers. Multiplexers manufactured by various commercial companies fail to address the majority of space radiation tolerance requirements or satisfy some of them with significant cost and performance penalties. Our company has developed radiation tolerant multiplexers that operate at data rates from 1-50*Gb/s* using either a BiCMOS or CMOS process. A specific multiplexer satisfies the requirements of the Space Wire standard.

1. INTRODUCTION

Intra-satellite robust data interconnects that operate at multi-gigabit data rates, featuring open system Plug-and-Play architecture, are required for the next generation of space exploration systems. High-speed time division multiplexers/serializers (MUXs) are critical components in these high data rate serial links. Various commercial companies currently offer high performance MUXs (e.g. Texas Instruments, Broadcom, AMCC, NEL, Yokogawa, Sierra Monolithic, etc.), but unfortunately they fall short of NASA's space-borne electronic radiation tolerance requirements including resistance to total ionizing dose (TID). The most widely known and studied structure that exhibits sensitivity to ionizing radiation is the silicon dioxide–silicon (SiO_2–Si) interface and its nearby regions [1, 2] that are found in all silicon MOS devices. A variety of radiation hardening-by-technology methods have been developed to mitigate radiation-induced upset and latch-up in silicon electronic circuits [3-5], but unfortunately they required special radiation-hard (RH) manufacturing processes that are overly expensive, require too much lead-time for most of today's space missions, and have limited available circuit components. Hardened by architecture or design techniques suitable for commercial processes either significantly increase the cost/area of the chip or incur excessive power/speed penalties.

Over the past few years, the Advanced Science and Novel Technologies Company (ADSANTEC) has developed a proprietary approach to the design of TID-tolerant integrated circuits including high-speed time division multiplexers. ADSANTEC's approach is to leverage the advantages of high-end commercial fabrication technologies and differential circuit architectures. IC fabrication technologies utilized by ADSANTEC include an 180*nm* SiGe BiCMOS and a 90*nm* CMOS commercial process. Hetero-junction bipolar transistors (HBTs) available in the BiCMOS processes are inherently TID-tolerant since they do not contain sensitive oxides near active regions. ICs utilizing HBTs as active devices satisfy both the speed and radiation requirements of space electronics, but require the application of current-switching architectures such as the CML or ECL logic families. The fully differential structures of the cells included in these logic families provide a better speed-power performance and higher radiation tolerance than their single-ended counter parts [6] in circuits with busy switching cycles, such as multiplexers.

A similar approach can be utilized in SCL logic circuits based on MOS transistors. Unfortunately, n-type MOS devices require application of special radiation-protection techniques. These practices limit the minimum achievable gate width and thus can support multi-gigahertz data rates only in deep sub-micron technologies with critical dimensions not exceeding 90*nm*. ADSANTEC has successfully utilized both CML and SCL architectures in its fuly TID tested RH multiplexer products as described below.

2. TID TOLERANT MUX CMU

This 10:1 MUX with internal clock multiplying unit (CMU) operating at a 1.25*Gb/s* data rate was fabricated in a 90*nm* CMOS process as part of a two-channel serializer-deserializer (SerDes) test chip with its microphotograph presented in Fig. 1. It consumes about 60*mW* of power and its 10:1 multiplexation ratio is fully compatible with the Space Wire standard that utilizes the 8B10B data encoding technique.

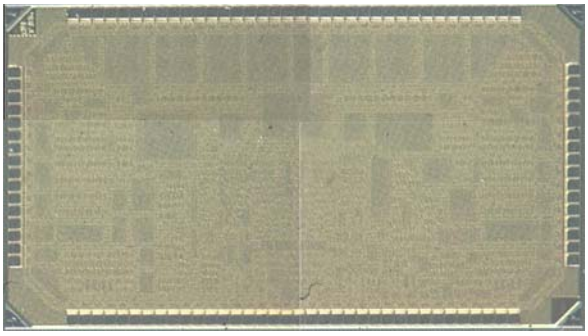


Fig. 1. 1.25Gb/s SerDes Chip.

The chip utilizes a specially designed SCL logic library with n-MOS transistors as active devices. The unique features of this MUX are its adaptive LVDS data input buffers with increased common-mode (CM) voltage range, a combinational serializing architecture, and the CMU with a fully differential ring voltage-controlled oscillator (VCO) and charge pump.

2.1. LVDS Input Buffer

The proprietary LVDS input data buffer [7] utilizes a novel adaptive architecture shown in Fig. 2 where signals with high CM voltages are fed directly into an open-drain SCL current switch (BUF1), while signals with low CM voltages are preprocessed by two pFET source followers (Shift) before being delivered into another open-drain SCL current switch (BUF2). A pair of load resistors connected to V_{CC} is used to sum the outputs of the two current switches. In order to avoid fluctuations of voltage swings across the loading resistors due to current deviations in the two BUF cells at different V_{CM} levels, controlled redistribution of current between BUF1 and BUF2 is included.

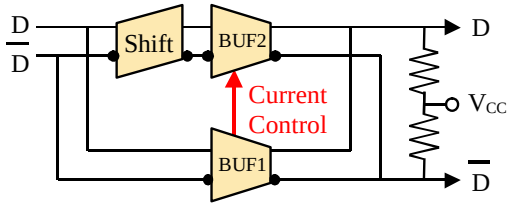
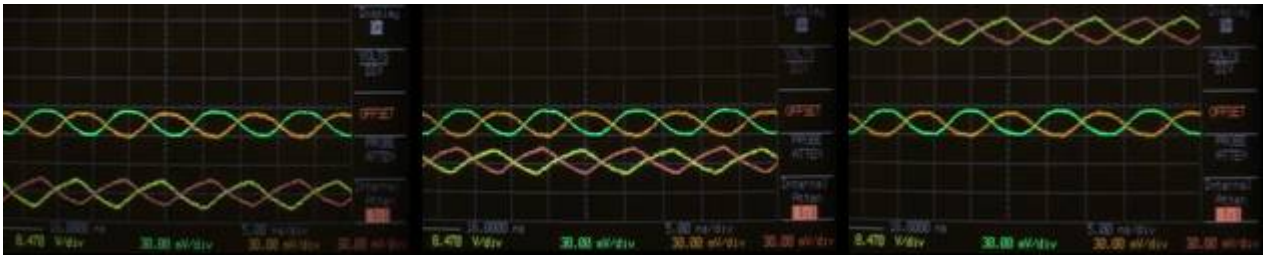


Fig. 2: LVDS Input Buffer's Block Diagram

The buffer operates at data rates above 2*Gb/s*, satisfies all the specifications of the LVDS standards [8, 9], and is capable of accepting LVDS differential signals with any CM voltage level between the negative and positive supply rails. This performance characteristic exceeds the requirements of the LVDS standards and allows for the buffer to tolerate a large amount of ground offset between transmitter and receiver. The three screen captures presented in Fig. 3 demonstrate the effectiveness of the developed adaptive technique. In all three figures, the output signal (green and brown) remains stable while the CM levels of the input signals (yellow and red) vary from 0*V* (Fig. 3a), through 500*mV* (Fig. 3b), to 2.5*V* (Fig. 3c).



a. b. c.

Fig. 3: Output Signal for Input CM Levels of 0V (a), 500mV (b), and 2.5V (c).

2.2. Combinational Architecture

Due to the serializer's 10:1 multiplexation ratio, a traditional tree-type architecture can not be used to serialize the input data streams. Instead, a combinational structure utilizing two unique 5-bit parallel-to-serial registers and one high-speed 2-to-1 multiplexer was constructed. Each stage of the register (MDF) includes three SCL cells: 2:1 multiplexer ("mux") and two D-type latches combined into a flip-flop "mdff" as shown in Fig. 4.

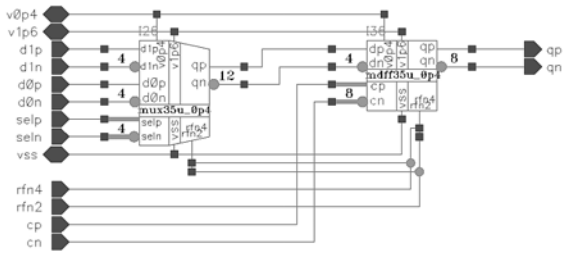


Fig. 4. MDF Schematic

The first input of the “mux” cell (“d1”) is connected to the output of the previous bit while the second one (“d0”) operates as the input of the current bit. The cell is controlled by a special select pulse (“sel”) with the length equal to one period of the register clock (“c”) and the period equal to five clock periods. The pulse is applied to all five MDA stages of the register and allows for the input data to pass into the flip-flops during one clock period. Within the other four clock periods, the “mux” cell is switched to the opposite state and the latched data is shifted through the register. The outputs of two registers are further serialized by another “mux” cell.

2.3. DIFFERENTIAL CHARGE PUMP AND RING VCO

The high frequency clock and select signals required for the shift register’s operation are generated from a low speed external clock signal by the on-chip CMU. The CMU employs a PLL architecture that contains several components including a differential charge pump and ring VCO. The design of a differential charge pump leads to a more radiation tolerant stage that eliminates the drawbacks of a differential-to-single-ended signal conversion. The ring VCO utilizes ADSANTEC’s proprietary SCL variable delay cells featuring the previously discussed RH approach. The cell consists of an SCL buffer and an SCL latch sharing the same loading resistors. The tail current is redistributed between the buffer and the latch by means of a linear selector controlled by a differential select signal. Shifting more current into the latch increases the total delay. To ensure stable operation of the cell, a certain initial current is provided for the buffer by a separate tail current source. Fig. 5 demonstrates the input reference clock and the corresponding 1.25*GHz* clock generated by the VCO and divided by the internal divider of the PLL.

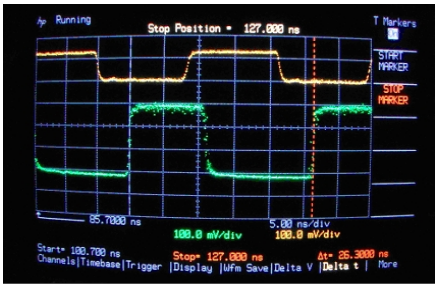


Fig. 5. PLL Operation.

3. TEMPERATURE-COMPENSATED DELAY LINES

A delay cell similar to the one described above, but designed in the CML architecture is used in ADSANTEC’s delay line family, which are intended for precise data alignment in different

channels of Space Wire interconnects. The delay adjustment with sub-picosecond accuracy is achieved by utilization of a standard interpolation technique while improved temperature stability of the delay value is provided by means of a unique adaptive compensation technique utilizing variable delay cells in both the short and long data paths. The control signals that increase the delay of the delay cells at lower temperatures to compensate for the decreasing delay of the other circuitry are generated by special blocks using HBT's base current as a reference. The delay lines utilize HBTs as active devices in all blocks and thus satisfy the radiation tolerance requirements. Delay lines with 120*ps* and 240*ps* delay adjustment ranges and DC to 15*GHz* frequency range have been successfully fabricated and tested. They are available as commercial products from ADSANTEC.

4. CONCLUSIONS

(1) TID tolerance has been demonstrated for high-speed SCL circuits utilizing MOS transistors fabricated in a commercial 90*nm* CMOS technology. The tested circuits include a complete 10:1 multiplexer operating at 1.25*Gb/s* data rate. A TID-tolerant PLL and a unique LVDS input buffer with extended common-mode voltage range designed in the same technology have been successfully fabricated and tested. TID tolerance has been demonstrated for a 12.5*Gb/s* CML-based multiplexer designed in the SiGe BiCMOS technology. Wide frequency range data delay lines with improved temperature stabilization have been successfully designed and tested using the same TID-proof approach.

REFERENCES

1. T. Ma, P. Dressendorfer, "Ionizing Radiation Effects in MOS Devices and Circuits", John Willey & Sons, New York, 1989.
2. J. Schwank, "Total Dose Effects in MOS Devices", IEEE NSREC Short Course, 2002.
3. G. Messenger and M. Ash, *The Effects of Radiation on Electronic Systems*, Van Nostrand Reinhold, New York, 1992.
4. L. Cohn et al., *Transient Radiation Effects on Electronics (TREE) Handbook*, Defense Nuclear Agency Rep. No. DNA-H-95-61, 1995.
5. A. Holmes-Siedle and L. Adams, *Handbook of Radiation Effects*, Oxford University Press, Oxford, UK, 1993.
6. G. Niu, R. Krithivasan, J. Cressler et al., "A Comparison of SEU Tolerance in High-Speed SiGe HBT Digital Logic Designed with Multiple Circuit Architectures", IEEE Trans. On Nuclear Science, v. 49, No. 6, Dec. 2002, pp.3107-3114.
7. V. Bratov, J. Binkley, V. Katzman, A. Bratov, A. Otero, G. Rakow, "Universal Input Buffer for Programmable Logic Devices", 9th MAPLD International conference, Washington, DC, September 26-28, 2006
8. "IEEE Standard of Low-Voltage Differential Signals (LVDS) for Scalable Coherent Interface (SCI)", IEEE Std. 1596.3-1996, 21 Mar. 1996, ISBN 1-55937-746-1.
9. "Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits", *ANSI/TIA/EIA-644-1995*, Telecommunications Industry Association, March 1996.

ELIMINATION OF COMMON MODE VOLTAGE REQUIREMENTS FOR LVDS USED IN SPACEWIRE

Session: SpaceWire Components

Long Paper

Jennifer Larsen

Aeroflex Colorado Springs

4350 Centennial Blvd. Colorado Springs, CO 80907

E-mail: *Jennifer.Larsen@aeroflex.com*

ABSTRACT

The signaling levels and physical layer used in SpaceWire is LVDS, as defined in the SpaceWire Standard ECSS-E-12A [3]**.** One of the limitations of LVDS seen over the years is the ±1 V common mode voltage range around the driver offset voltage (+1.25V typical). This common mode voltage can pose problems when using LVDS to communicate between multiple systems that have different power references and common chassis grounds are not available. Operating through a transformer eliminates the tight common mode voltage issue and allows a ground offset of at least ±7 V. An evaluation was preformed using 1:1 transformers to couple the UT54LVDS031LV LVDS Driver with the UT54LVDS032LV LVDS Receiver.

1 BACKGROUND

Low Voltage Differential Signaling, LVDS, is useful in applications that require low power, low noise, and high speed point-to-point communications. The physical layer of SpaceWire uses DS Encoded LVDS to communicate serial full-duplex bidirectional data. Communications are encoded and carried out by using two differential signals in both transmit and receive directions.

LVDS uses Field-Effect Transistors to control the direction of the constant 3.5mA current source. This current is terminated by a 100Ω termination resistor across the inputs of the Receiver to generate a +/-350mV signal. SpaceWire uses LVDS to Transmit and Receive the Data-Strobe encoded data. A Single Point-to-Point SpaceWire Link contains 4 Drivers and 4 Receivers that can drive up to 10m of cable and are terminated at each end by nine-pin micro-miniature D-type plugs.

Input/Output signal levels are defined by ANSI/TIA/EIA-644, this is an electrical signaling standard only, and does not define a protocol. The protocol used for SpaceWire is derived from IEEE 1355-1995. These standards as well as other information on SpaceWire are defined in the SpaceWire Standard Document ECSS-E-50-12A.

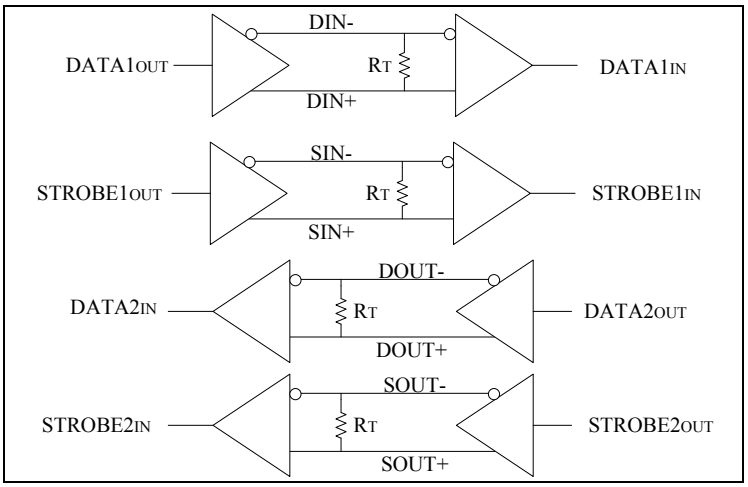


Figure 1. Single point to Point SpW Link

One of the limitations of LVDS is the ±1 V common-mode range around the driver offset voltage (+1.25V typical) which aids in the rejection of system noise that is coupled between the differential pairs. Common mode voltage can pose problems when using LVDS to communicate between multiple systems that have different power references, and common chassis grounds are not available.

LVDS Receivers can tolerate a minimum of ±1V ground offset between the Drivers ground and the Receivers ground reference. This ground offset range is commonly referred to as the Common Mode Range and is illustrated in figure 2.

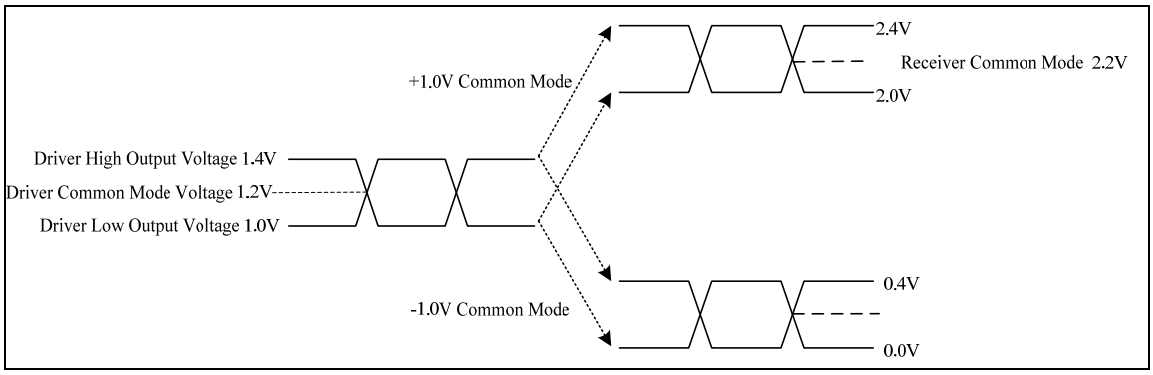


Figure 2. Driver/Receiver Common Mode Range

Common mode signals appear on both positive (+) and negative (-) lines of the LVDS devices. Common mode voltage (VCM) is the DC average of the absolute value of the + and - signal voltages with respect to the device ground.

$$V_{CM} = \frac{(|V+|)+(|V-|)}{2}$$

The offset voltage of the LVDS Driver is 1.2V, and the receivers can tolerate an offset of ±1V offset. This Driver offset makes the common mode range of the Receiver +0.2V to +2.2V.

Using transformers to couple the LVDS differential pairs can eliminate the common mode voltage and allow LVDS to be used in systems that do not share a common power and ground references. Transformers are used to couple the driver side of the circuit from the receiver side. This transformer allows the switching AC signal from the driver into the receiver side without electrically connecting the two LVDS components. The DC common mode will not be transferred because there is zero potential difference between the transformer windings, resulting in the DC common mode being blocked and the AC data signal to pass.

2 COMMON MODE RANGE EXPANSION

Using transformers to couple the LVDS differential pairs can eliminate the common mode voltage and allow LVDS to be used in systems that do not share a common power and ground references. These non common ground references may be outside the ±1V common-mode range that is accepted by the Receiver. The transformer's isolation feature will couple the Driver side of the circuit to the Receiver side without direct wire connections, and negate the difference in ground references.

The ideal transformer transfers alternating, differential mode current only. As the Driver changes logic states from 1 to 0 the current through the primary winding of the transformer will alternate. Alternating current flow through the primary winding creates a changing magnetic field. This magnetic field induces a voltage that charges the secondary windings. Current flows through the transformer when a load, the Receiver, is added to the secondary winding.

Inserting a 1:1 transformer on the receiver side of the LVDS system, before the termination resistor, will allow the Driver and Receiver to communicate while being on non-common ground planes. The common mode voltage is a DC signal and will not be transferred from the primary to secondary windings. There is zero potential difference across the transformer windings and therefore no magnetic field is generated. The following figure shows the implementation of a LVDS Driver and Receiver that are coupled together using a transformer.

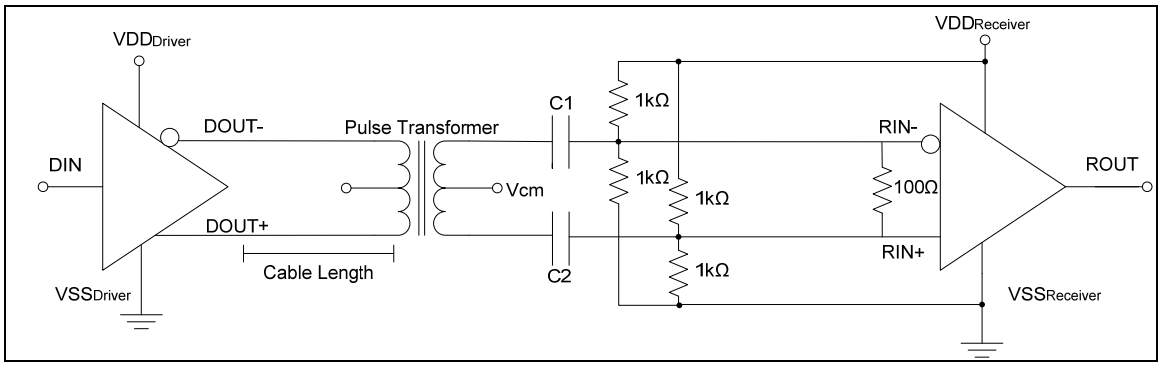


Figure 3. Transformer coupled LVDS System
NOTE: all additional components are at the Receiver end of the system.

The transformer will allow the ground references, specifically $V_{SSDriver}$ and $V_{SSReciver}$, to have a static offset between them without interfering with communications. The transformer will ignore the difference in the common mode and the data will be transferred as if the Driver and Receiver were on the same ground plane.

3 EVALUATION

Experimental success is based on demonstrating functional equivalence of a transformer coupled physical layer to the resistive terminated physical layer as specified in TIA/EIA-644A. Aeroflex LVDS Driver (UT54LVDS031LV) and Receiver (UT54LVDS032LV) were used in this evaluation. The proper signaling levels needed to be transferred from the driver's outputs through the transformers windings to ensure proper output at the receiver. The Receiver Datasheet input parameters must be met by the equivalent system to ensure that data at ROUT is acceptable.

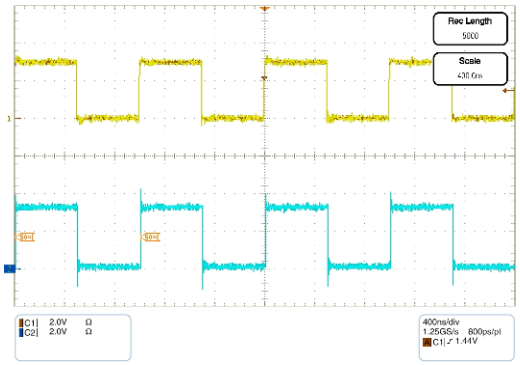
System jitter performance (ROUT) was also an experiment parameter that was characterized. Jitter is defined as variation of a periodic signal. The jitter in the signal may be due to variations in frequency, period, amplitude, or phase. Jitter can be measured by collecting data on the signal of interest and seeing how the signal varies over time. For the transformer coupled LVDS experiment to be successful a minimal amount of jitter can be introduced by the transformer. Looking at the data eye for the jitter produced at each of the frequencies, each of the crossings was acceptable.

4 INITIAL T-330SCT PULSE TRANSFORMER DATA

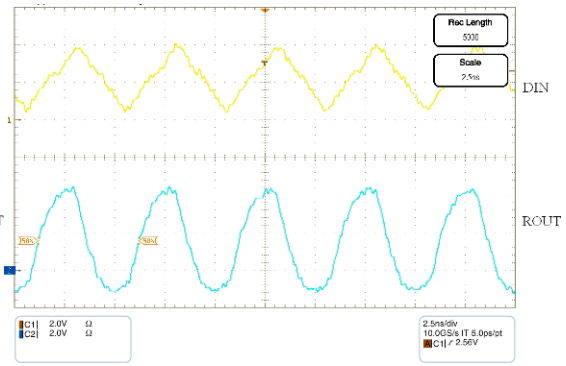
Data was collected for DIN vs. ROUT for 1 meter and 10 meter cable lengths with data rates of 1MHz (2Mbps), 100MHz (200Mbps), 200MHz (400Mbps), at temperature 25°C. Additional considerations: (refer to Figure3.)

C1 = C2 = 0.1uF for DC Isolation
1kΩ resistors used for Common Mode Voltage on Receiver side
Driver/Receiver parts used: UT54LVDA031LV UT54LVDS032LV
Cable Type = RG174 50Ω SMA connectors

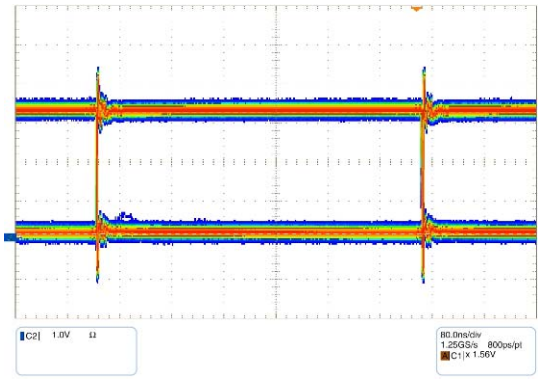
4.1 1 METER DATA



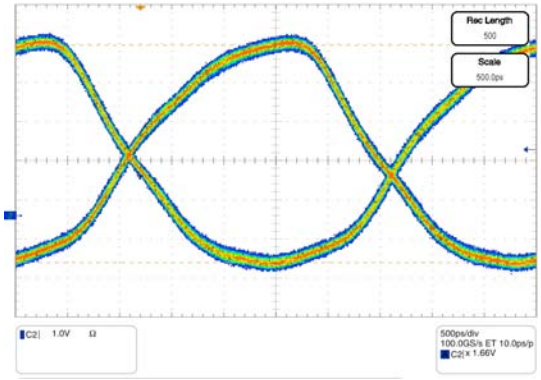
1 meter cable 1MHz



1 meter cable 200MHz

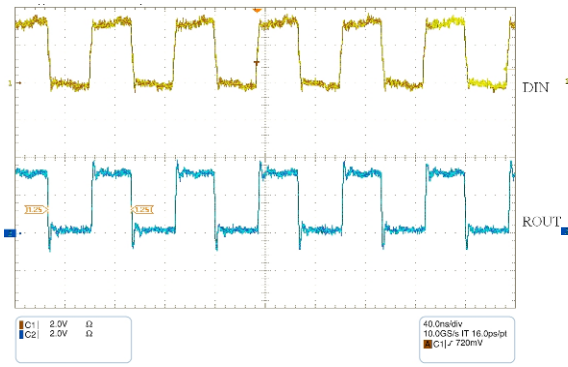


1 meter Jitter Eye at 1MHz

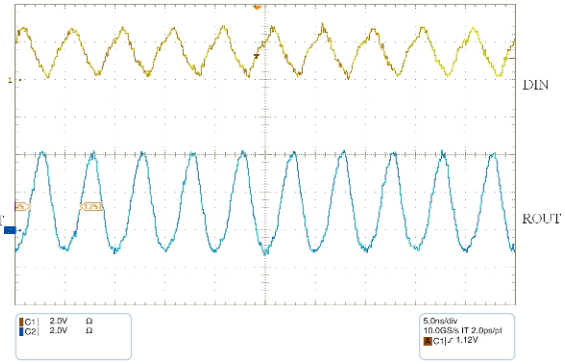


1 meter Jitter Eye at 200MHz

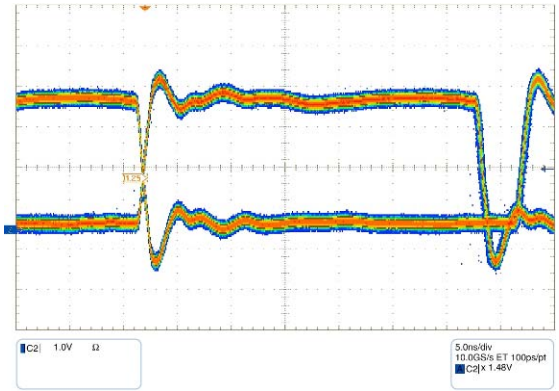
4.2 10 METER DATA



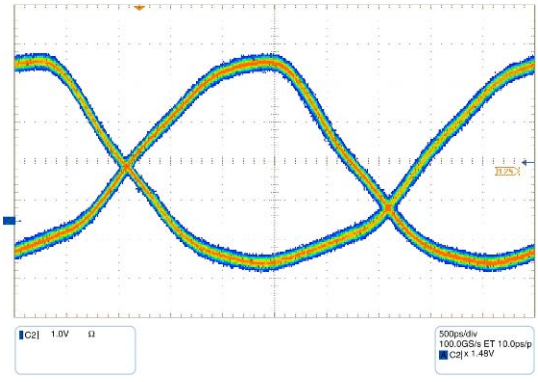
10 meters 15MHz



10 meters 200MHz



10 meters Jitter Eye 15MHz



10 meters Jitter Eye 200MHz

5 CONCLUSION

Using transformers to couple the LVDS physical layer of the SpaceWire allows for a common mode voltage difference between the Driver and Receiver exceeding ±7.0V. Transformer selection is frequency dependent. A transformer must be selected to accounts for signaling speeds and system parasitic characteristics. Transformer coupling the physical layer of SpaceWire will allow LVDS communications between nodes that have ground references outside the ±1V common mode range.

6 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-50-12A", The Netherlands, January 24, 2003.

[4] Aeroflex, "UT54LVDS031LV 3.3-VOLT QUAD DRIVER Datasheet", Colorado Springs, Colorado, February 2006.

[5] Aeroflex Colorado Springs, "UT54LVDS032LV 3.3-VOLT QUAD RECEIVER Datasheet", Colorado Springs, Colorado, March 2006.

OPERATIONALLY RESPONSIVE SPACE DATA INTERFACE STANDARDS AND PROTOTYPES EMPLOYING SPACEWIRE

Session: Poster

Short Paper

Ramon Krosley

Design_Net Engineering, 10311 West Hampden Ave, Suite A107, Lakewood, CO 80227, USA

Brian Davis

Space/Ground System Solutions, Inc. 4343 Fortune Place, Suite C, W. Melbourne, FL 32904, USA

Paul Jaffe

Naval Research Laboratory Code 8243, 4555 Overlook Ave SW, Washington, DC 20375, USA

E-mail: rkrosley@design-group.com, brian.davis@sgss.com, paul.jaffe@nrl.navy.mil

ABSTRACT

The rapid integration, launch, and deployment of satellites in response to emerging needs has been termed "Operationally Responsive Space" (ORS). One vision of ORS calls for the positioning in a depot of interchangeable satellite payloads and spacecraft buses with a common interface. Upon direction to deploy a particular mission, the appropriate payload would be selected and integrated with a bus, and the space vehicle would be launched. To support such a system, standardized hardware and software interfaces are needed between the payload and bus. For the development of ORS Bus Standards, the SpaceWire standard (ECSS-E-50-12A) has been specified as part of such a payload-bus interface for high rate data. Other data standards such those promulgated by the Consultative Committee on Space Data Standards (CCSDS) have been leveraged and extended to address ORS needs. The result has been encapsulated in the document "ORS Standard Data Interfaces: Bus to Payload, Bus to Ground". This document includes data exchange protocols, data transport formats, packet definitions and field definitions for ORS implementations. The intent has been to levy interface standards where these standards will lead to efficient system integration, robust operational capabilities, and improved system flexibility while reducing overall system cost. Prototyping efforts, including those associated with the U.S. Department of Defense's TacSat-4 experimental satellite, and those as part of Design_Net Engineering's Flight Software Standards Testbed have shown the utility and efficacy of the data interface standards.

1 CONTEXT

The strategy for achieving the goals of the ORS program relies upon the ability to assemble commodity parts without engineering changes. The granularity of the parts is a strategic decision. The implementations in this report treat payload and bus as the parts which must be interchangeable. The technical problems to be solved to achieve interchangeability are

similar when the parts are subsystems or components. SpaceWire offers a common ground in which to solve these technical problems, because it can connect parts at all the levels of granularity identified here.

2 STANDARDS

A clear requirement to enable interchangeable commodity parts is to standardize the interface between them. This section identifies a subset of the standards chosen to implement the requirements of the ORS program, specifically, those standards that are relevant to the SpaceWire implementations in this report. These standards must span the architectural layers from physical to application, in order to achieve the requirement of interchangeable subsystems.

2.1 SPACEWIRE

Two SpaceWire standards guided these implementations: the base standard and the plug-and-play standard. The base standard [1] defines the physical architecture of a data network in a spacecraft. The plug-and-play standard [2] facilitates discovery of the topology the network, and includes a special service for the application layer by defining an xTEDS packet. The xTEDS packet carries metadata that describes the data interface of a SpaceWire endpoint.

2.2 CCSDS PACKET FORMATS

The CCSDS definition of packet formats [3] provides a widely adopted logistical envelope for the data that flows in a space mission. The CCSDS packet headers define how to deliver packets in a way that is independent of the physical medium for distribution. Because of their independence of the physical medium, the CCSDS headers provide the means for application software to carry packets across a variety of physical media, including SpaceWire and space-ground links.

2.3 STANDARD DATA INTERFACES

The preceding standards provide essential infrastructure, but they do not completely span the architectural layers. One of the contributors to this paper (Davis) authored with input from the ORS Integrated Systems Engineering Team (ISET) a document [4] that integrates the SpaceWire and CCSDS standards, and provides the missing information to define the specific application interface needed for general ORS missions. This document includes data exchange protocols, data transport formats, packet definitions and field definitions for ORS implementations. It is freely available for download at:

https://projects.nrl.navy.mil/busstandards/

as

ORSBS_004_DATA_IF_STD_R2_6Dec07.124818.pdf

3 IMPLEMENTATIONS

Three SpaceWire networks have been implemented using the standards mentioned above. The complexity of these networks has increased chronologically as more spacecraft components have joined.

3.1 TacSat-4

The SpaceWire in TacSat-4 runs from point to point, without routers [5]. This implementation [6] has served to test the concepts in the data interface standard. TacSat-4 assures that the data interface standard covers the exchange of information needed for a successful mission.

The TacSat-4 Bus CT&DH and Flight Software fully implemented and is compliant with the Bus/Payload SpaceWire Interface as defined by the Data Interface Standards Document. The CT&DH provides for two SpaceWire links; one for payload communications, the other for ground test related communications. The data protocols, data transport envelopes, message structures and fields are implemented per the Data Interface Standards Document. The CT&DH provides a logical unit lookup table and DMA capabilities to route data from the SpaceWire interfaces to SSR segments based on the data communications destination address.

3.2 Universal Interface Electronics

MicroSat Systems, Inc. developed and tested against ORS Bus prototype hardware and software the Universal Interface Electronics (UIE), a versatile multifunction unit that employs SpaceWire and RS-422 interfaces. Its different interfaces and various configurations allow it to perform as a protocol translation, power distribution, data storage, or telemetry collection and consolidation node. The UIE employs the LEON3 processor in an Actel RTAX2000 FPGA and also includes a Xilinx Virtex II FPGA for mission-specific configurations.

3.3 Flight Software Standards Testbed

The SpaceWire implementation in the Flight Software Standards Testbed (FSST) forms a small network with a few routers which link a small set of computer boards. This implementation was built to explore emerging software standards for spacecraft data systems, in order to maintain and to extend the ORS program.

The FSST contains application software objects that behave like the standard bus and the standard payload as defined in the ORS data interface standard. A flight simulator provides a high-fidelity representation of the environment of a virtual spacecraft responding to typical mission scenarios, with FSST flying the virtual spacecraft in the role of the flight data system. One of the processors in the FSST acts as a bridge to route Ethernet packets from/to virtual instruments in the flight simulator and across SpaceWire to/from flight software objects in the FSST. Because the ORS and Space Plug and Play Avionics (SPA) standards share a common physical and link layer implementation in SpaceWire, the Reconfigurable Processor Boards (RPBs) used for the FSST development are usable for either of these complementary standards approaches.

Figure 1. The FSST RPB stack

The FSST is currently engaged in the following SpaceWire-related experiments.

3.3.1 CONFORMANCE TESTING

A manufacturer of a standard bus or a standard payload should be able to connect their equipment to a SpaceWire port of the FSST, and communicate with their counterpart in the FSST. The FSST can be configured to deliver and to receive the same packets as flow in TacSat-4 across a point-to-point SpaceWire link.

3.3.2 SPACEWIRE PLUG-AND-PLAY

The FSST uses as its operating platform the Satellite Data Model (SDM), developed by Utah State University for PnPSat. The SDM uses SpaceWire plug-and-play to discover the physical topology of the network in which it finds itself. SDM reads metadata (called xTEDS) from each of the endpoints in its network to discover the application data and services that are present. In other words, the interchangeability of components in SDM requires that discovery occur on two architectural levels: the physical and the application.

On PnPSat, SDM does not use the xTEDS packet defined in the SpaceSire plug-and-play standard. This raises the question whether the xTEDS packet is needed in that standard. Some mechanism is needed for SDM to obtain metadata from a device, and solicitation as defined in the SpaceWire plug-and-play standard resembles the mechanism implemented in PnPSat. A better question is whether SpaceWire plug-and-play can do more to facilitate application-level discovery. For example, the LL protocol ID used by PnPSat is a candidate extension for the SpaceWire plug-and-play standard at the level of discovery of applications.

4 REFERENCES

1 “SpaceWire - Links, nodes, routers and networks“, ECSS-E-50-12A, 2003.

2 “SpaceWire Plug and Play Draft A”, http://tech.groups.yahoo.com/group/SpaceWirePnP/files/Draft%20Specification/, 2008.

3 “Space Packet Protocol”, CCSDS 133.0-B-1, Sections 4.1.1, 4.1.2, 4.1.3, 2003.

4 “Operationally Responsive Space Standard Data Interfaces: Bus to Payload, Bus to Ground ORSBS-004”, Naval Center for Space Technology Report NCST-S-SB008, 2007.

5 “Operationally Responsive Space Payload Developers Guide (PDG) ORSBS-003”, Naval Center for Space Technologies Report NCST-IDS-SB001.

6 Jaffe, P., Clifford, G., Summers, J., “SpaceWire for Operationally Responsive Space as part of TacSat-4”, International SpaceWire Conference, Dundee, Scotland, September, 2007.

2 MHz Link Initialization of MDP onboard MMO

Session: SpaceWire test and verification

Short Paper

Masahiro Taeda, Shigeru Ishii, Atsushi Nakajima, Hiroshi Ikebuchi,

Yoshikatsu Kuroda

Mitsubishi Heavy Industries Ltd,

1200, O-Aza Higashi Tanaka, Komaki, Aichi, Japan 485-8561

Takeshi Takashima

Department of Space Plasma Physics, Institute of Space and Astronautical Science (ISAS), Japan Aerospace Exploration Agency (JAXA)

3-1-1 Yoshinodai, Sagamihara, Kanagawa, Japan 229-8510

Yasumasa Kasaba

Department of Geophysics, Graduate School of Science, Tohoku University

6-3, Aoba, Aramaki, Aoba-ku, Sendai, Japan 980-8578

E-mail: masahiro_taeda@mhi.co.jp, shigeru_ishii@mhi.co.jp, atsushi_nakajima@mhi.co.jp, hiroshi_ikebuchi@mhi.co.jp, yoshikatsu_kuroda@mhi.co.jp, ttakeshi@stp.isas.jaxa.jp, kasaba@pat.geophys.tohoku.ac.jp

Abstract

Mission Data Processor (MDP) onboard Mercury Magnetospheric Orbiter (MMO) for the BepiColombo mission is the payload controlling and mission data handling instrument, which has SpaceWire interfaces to the payloads.

The extremely small power resource of the MMO caused to lower the data signalling rate between the payloads and MDP down to 2 MHz, which is much lower than the original initialization speed of 10 MHz. We thus investigated the ability of the link initialization of 2 MHz, by both the theoretical analysis and the practical test.

We found that it strongly depends on the internal latency of the SpaceWire interfaces to connect each other. And for the payload with its base clock as low as 2 MHz to minimize the power dissipation, the implementation of the interface logic is strongly restricted. In practice, our onboard test showed that the link successfully connected when both nodes are “Link Enable” mode, while it failed to connect when the payload side adopts “Auto Start” mode.

Therefore, we propose to limit the link setting mode only “Link Enable” for the MMO case.

1 INTRODUCTION

The extremely small power resource of the MMO satellite require MDP and the each payload to use its system oscillator as lower frequency as possible, because lowering the base clock is the most effective to lower the power dissipation.

This restriction caused to lower the data signalling rate between payloads and MDP down to 2 MHz, which is much lower than the original initialization speed of 10 MHz We thus investigated the ability of the link initialization of 2 MHz.

2 2 MHZ LINK INITIALIZATION TESTS

In the verification of 2 MHz link initialization, we checked it two ways; the theoretical analysis and the onboard test. In the analysis, we assumed that the data signalling rate is just 2 MHz, exchange timeout is nominal, and took the internal latency of the both nodes into account, as the actual SpaceWire interface model. We then performed the onboard test of initialization, using MDP and the dummy payload unit with MHI SpaceWire IP. At the test, we set the base clock of the SpaceWire interface of the dummy payload to 2 MHz.

We investigated two case of the settings as the payload; (1)"Auto Start" mode, and (2)"Link Enable" mode, while only "Link Enable" mode is adopted for MDP.

2.1 "AUTO START" MODE CASE

In the case of "Auto Start" mode, the payload transit its state from "Ready" to "Started" when detecting first NULL packet from MDP, and send NULL packet to MDP. Within the exchange timeout of 12.8 us (nominal) at MDP, SpaceWire interfaces transmit two NULL packets.

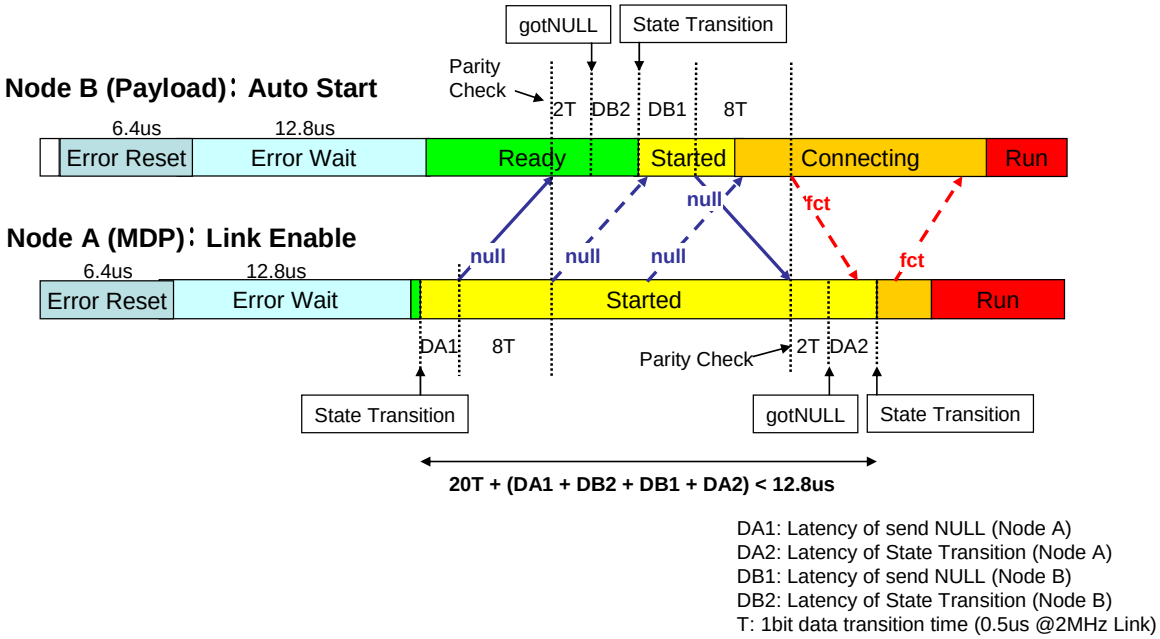


Fig. 1 The analysis of the link initialization at "Auto Start" mode

When the data signalling rate is 2 MHz, the total internal latency of the both nodes must be shorter than 2.8 us, since the total needed transmission time of the 8bit NULL and extra 2bits for parity checks is 10.0 us, as shown in Fig 1.

For the payload with the base clock of 2 MHz, the internal latency at the payload should be lower than 5 cycles (including, gotNULL check, state transition, and NULL packet sending, etc), which is hard to implement in practice even if the latency of the MDP is ignorable.

In fact, we found it failed to connect each other with 2 MHz link by the onboard test.

2.2 "LINK ENABLE" MODE CASE

In the case of "Link Enable" mode, both MDP and Payload automatically transit from "Ready" to "Started" state, and send NULL each other. Thus, it needs only one NULL transmission within the exchange timeout of 12.8 us (nominal).

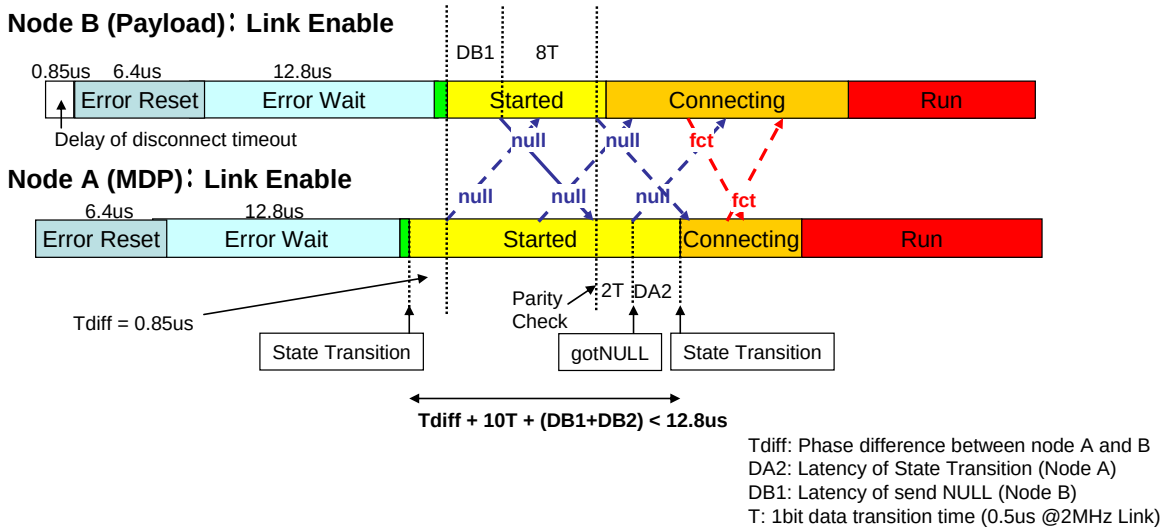


Fig. 2 The analysis of the link initialization at "Link Enable" mode

In this case, limitation of the total latency for the both nodes becomes longer up to 6.95 us, as shown in Fig. 2. This means the allowed latency of the payload can be longer to 13 cycles, which is much easier and implementable in practice. And our onboard test confirmed that it successfully connected with 2 MHz link.

3 CONCLUSION

We found that it is possible to connect the SpaceWire link successfully when both MDP and the payload interfaces set the "Link Enable" mode without requiring strong restriction of the implementation to the payload, even in the condition that the base clock and the data signalling rate of the payload is as low as 2 MHz.

We, hence, propose to add the interface requirement to limit the link setting mode only "Link Enable" for the MMO case, although further investigation is needed to consider the worst case of the exchange timeout.

SWIMμν: AN ULTRA-SMALL GRAVITATIONAL-WAVE DETECTOR WITH SPACECUBE2 ON A JAXA'S 100KG-CLASS SATELLITE

Session: Poster Presentations

Short Paper

Wataru Kokuyama, Masaki Ando, Koji Ishidoshiro, Kakeru Takahashi, Kazuhiro Nakazawa, Takayuki Yuasa, Teruaki Enoto, Kimio Tsubono

Department of Physics, The University of Tokyo, Bunkyo, Tokyo 113-0033, Japan

Shigenori Moriwaki

Department of Advanced Materials Science, The University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Chiba 277-8561, Japan

Akito Araya, Akiteru Takamori

Earthquake Research Institute, The University of Tokyo, 1-1-1 Yayoi, Bunkyo, Tokyo 113-0032, Japan

Yoichi Aso

California Institute of Technology, M/C 18-34, Pasadena, CA 91125, USA

Takeshi Takashima, Tadayuki Takahashi, Motohide Kokubun, Tetsuo Yoshimitsu, Hirokazu Odaka, Takehiko Ishikawa, Shin-ichiro Sakai, Tomoaki Toda, Tatusaki Hashimoto, Ayako Matsuoka

Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency, Sagamihara, Kanagawa 229-8510, Japan

Keiko Kokeyama

National Astronomical Observatory of Japan, Mitaka, Tokyo 181-8588, Japan

Shuichi Sato

Department of System Control Engineering, Hosei University, Koganei, Tokyo 184-8584, Japan

E-mail: kokuyama@granite.phys.s.u-tokyo.ac.jp, ando@granite.phys.s.u-tokyo.ac.jp, koji_i@granite.phys.s.u-tokyo.ac.jp, kakeru@granite.phys.s.u-tokyo.ac.jp, nakazawa@amalthea.phys.s.u-tokyo.ac.jp, yuasa@amalthea.phys.s.u-tokyo.ac.jp, enoto@amalthea.phys.s.u-tokyo.ac.jp, tsubono@phys.s.u-tokyo.ac.jp, moriwaki@hagi.k.u-tokyo.ac.jp, araya@eri.u-tokyo.ac.jp, takamori@eri.u-tokyo.ac.jp, aso@granite.phys.s.u-tokyo.ac.jp, ttakeshi@stp.isas.jaxa.jp, takahasi@astro.isas.jaxa.jp, kokubun@astro.isas.jaxa.jp, kikko@nnl.isas.jaxa.jp, odaka@astro.isas.jaxa.jp, ishikawa.takehiko@jaxa.jp, sakai@isas.jaxa.jp, toda@isas.jaxa.jp, hashimoto.tatsuaki@jaxa.jp, matsuoka@isas.jaxa.jp, keiko.kokeyama@nao.ac.jp, sato.shuichi@k.hosei.ac.jp,

ABSTRACT

SWIMμν is an ultra-small gravitational-wave detector on SDS-1, a JAXA's small satellite for technology demonstration, to be launched in FY 2008. SWIMμν aims at: (i) observing gravitational wave from orbit for the first time, (ii) being a precursor for future space gravitational wave detector missions, (iii) detecting vibrational motion of the satellite which is not well-studied noise source of a space gravitational-wave detector. In this paper we present an overview of SWIMμν and a SpaceWire-based data processing system used in it.

1 INTRODUCTION

Gravitational-wave is a ripple of the space-time, predicted by A. Einstein in 1916 as one of the consequences of the General Relativity. In spite of strong efforts for a long time, gravitational-waves have not been directly detected because of thier extreme weakness. Direct detection of gravitational-wave will enable us to unveil the hidden parts of the universe such as central engine of gamma-ray bursts and origin of supermassive black holes.

In Japan, it is planned to launch spacecraft named DECIGO [1] to detect low-frequency (<10Hz) gravitational-waves. DECIGO will have enough sensitivity to detect stochastic gravitational-wave background from inflation. As a precursor mission for DECIGO, DECIGO pathfinder (DPF) [2] is proposed. DPF has been selected as one of 15 candidate missions for JAXA's small scientific-satellite program. As a first step to DPF, we developed SWIMμν, an ultra-small gravitational-wave detector in SWIM (SpaceWire Interface Demonstration Module). SWIM is a module loaded on SDS-1, which stands for Small Demonstration Satellite-1: JAXA's 100kg-class satellite for space technology demonstrations. SWIMμν will be the first gravitational-wave detector in orbit.

2 OVERVIEW OF THE DETECTOR

Fig. 1 shows photos of SWIMμν along with SpaceCube2 [3], a space-qualified computer with SpaceWire interface. SpaceCube2 is used for data acquisition and detector control.

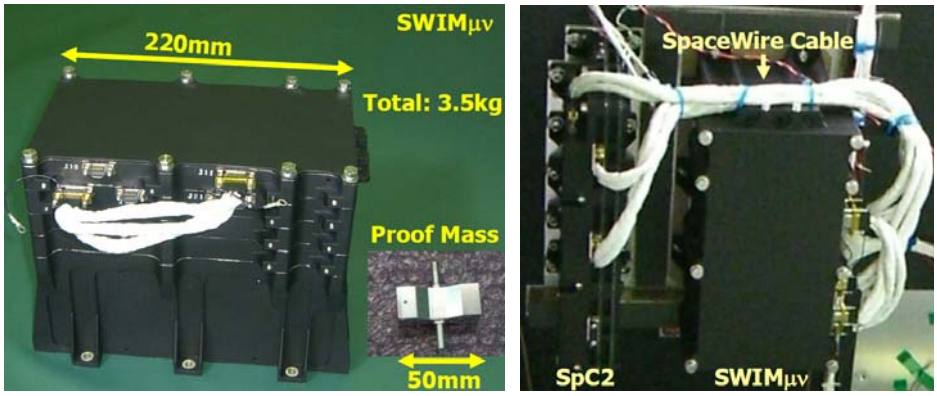


Fig.1: (Left) Photo of SWIMμν and a proof mass within it: SWIMμν contains two proof masses shown in the photo. (Right) Photo of SWIMμν and SpaceCube2 attached on SDS-1 spacecraft: SpaceWire cables between SpaceCube2 and SWIMμν are wrapped with glass-fiber tape. Courtesy: JAXA

Fig. 2 is a schematic diagram of SWIMμν and SpaceCube2.

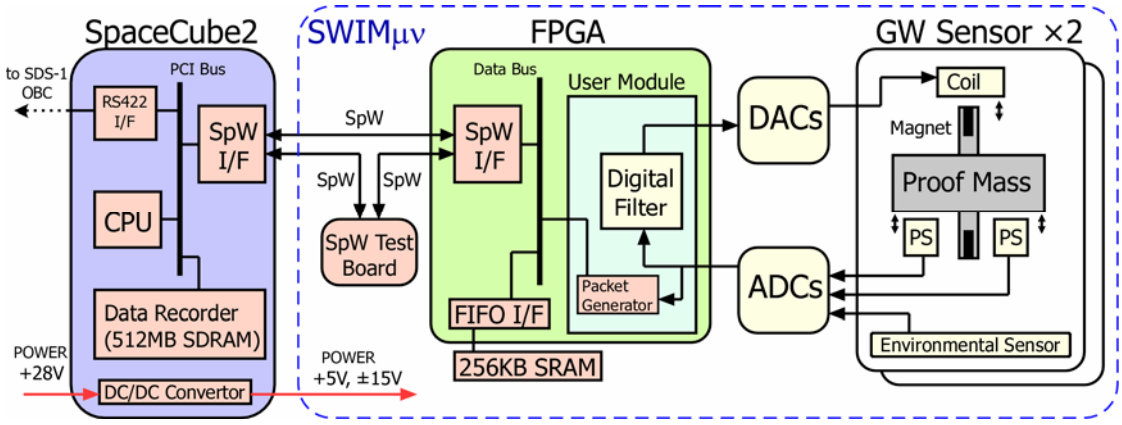


Fig.2: A schematic diagram of SWIMμν and SpaceCube2. We implement digital PID filters on the FPGA.

2.1 Gravitational Wave Sensor

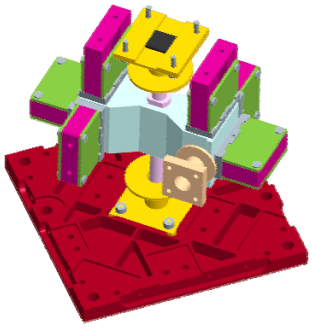


Fig. 3: Structural view of the sensor: 6 photo sensors (pink box) and 4 coils (yellow cylinder) are attached to frame, surrounding the proof mass in the center.

Fig.3 shows internal structure of the gravitational-wave sensor module, which consists of a proof mass, displacement sensors, actuators, and environmental sensors.

Each module has a 50g-weight bar-shaped proof mass made of aluminium, which does not touch the frame. To sense displacement between the proof mass and the frame, six photo sensors are placed around it. They are infrared reflection-type photo sensor; a LED in a sensor emits infrared light and photo diodes (PD) detect the light reflected from the surface of the proof mass. The intensity of light, which depends on the distance between the proof mass and the photo sensor, is read out. Because of insufficient room and electricity, its displacement noise is about 10^{-9} m/Hz$^{1/2}$ at 1Hz, not as sensitive as ground-based laser interferometers. For proof-mass actuators, four Nd-magnets are attached to it. Coils around them can actuate the proof mass. Sensor modules also have on-chip gyros and accelerometers as environmental sensors which will be utilized to calibrate the main sensor.

Gravitational-waves would make differential torques to the two proof masses, which are placed orthogonal to each other. The sensor works also as a sensitive accelerometer and therefore SWIMμν can detect vibration of the satellite which is not well-studied noise source of space gravitational-wave detectors.

2.2 Digital Feedback System

Digital Feedback system keeps the two proof masses at the balanced point in the following way: Relative displacement signals between the proof mass and the frame are detected by 6 photo sensors. The signals are multiplexed and converted to 16-bit digital value by ADCs. FPGA acquires the data and applies digital PID filters to them.

Filtered data are sent to DACs which control current in coils. Then coil-magnet pair generates opposite force to external disturbances and consequently keep the proof mass at the balanced position.

The signals obtained from the feedback system may contain disturbances induced by gravitational-waves. The data will be searched for gravitational-waves in off-line analysis.

2.3 DATA ACQUISITION SYSTEM

In order to collect data from sensors, we adopted SpaceWire-based data acquisition framework. SpaceCube2, a computer equipped with SpaceWire interface, and a FPGA in SWIMμν communicate with each other by SpaceWire/RMAP (Remote Memory Access Protocol) at the speed of ~1Mbps.

We collect science data as follows: First, data packets generated by FPGA from sensor signals are stored tentatively in 256KByte SRAM FIFO buffer. The buffer can keep 5.4 second data since the FPGA creates data at the rate of ~ 380kbps. The amount of data in the buffer is indicated by FIFO controller on FPGA, and is checked from SpaceCube2 via SpaceWire every 0.5sec. When the buffer filled over a half, SpaceCube2 begins to pull out of data via SpaceWire. The received data packets are stored in Data Recorder (DR), or 512MB SDRAM within SpaceCube2. Using API of the middleware, detector control software on SpaceCube2 can easily access DR. Finally the data are sent to ground station via SDS-1 OBC (On Board Computer) connected with SpaceCube2 by RS422 serial link.

SpaceCube2, FPGA in SWIMμν and SpW test board are connected triangularly by SpaceWire as shown in Fig.2. We can change data transfer route by sending a command to SpaceCube2. That will demonstrate communication redundancy of SpaceWire/RMAP which is an advantage over conventional network protocols for spacecraft.

3 CURRENT STATUS OF SWIMμν

We have already finished development and performance test of SWIMμν. As shown in Fig.1 (right), it has been mounted on SDS-1 which is now under overall examination for launch with H-2A rocket from Tanegashima in FY 2008.

4 REFERENCES

1. Naoki Seto, Seiji Kawamura, and Takashi Nakamura, “Possibility of Direct Measurement of the Acceleration of the Universe Using 0.1 Hz Band Laser Interferometer Gravitational Wave Antenna in Space”, Physical Review Letters, **87** (2001), 221103

2. Masaki Ando et al., “DECIGO Pathfinder”, Journal of Physics: Conference Series, July 2008, **120** 032005

3. Tadayuki Takahashi, et al, “SPACE CUBE 2 - AN ONBOARD COMPUTER BASED ON SPACE CUBE ARCHITECTURE”, International SpaceWire Conference 2007, Dundee, UK, September 2007

SpaceWire application for flexible and rapid development of the scientific satellite platforms

Session : Poster Session
Short Paper

Shin-ichiro Sakai[*1], Seisuke Fukuda[*1], Shujiro Sawai[*1], Nobutaka Bando[*1], Takahiro Yamada[*1], Tadayuki Takahashi,[*1]
[*1] *Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency (ISAS/JAXA)*

Yasumasa Kasaba[*2], Masaharu Nomachi[*3], Hiroki Hihara[*4], Naoto Ogura[*4], Kenichi Baba[*4] , Tetsu Saito[*5], Takashi Kominato[*6], Takayuki Tohma[*6]
[*2] *Tohoku University,* [*3] *Osaka University,* [*4] *NEC TOSHIBA Space Systems, Ltd.,*
[*5] *NEC Aerospace Systems, Ltd.,* [*6] *NEC Corporation*

E-mail : sakai@isas.jaxa.jp, fukuda@isas.jaxa.jp, sawai@nnl.isas.jaxa.jp

ABSTRACT
This paper briefly introduces an idea to use SpaceWire for attitude and orbit control subsystem (AOCS) data interface. Conventional Japanese scientific satellites were custom-made satellites, and the AOCS computer architecture also follows this philosophy. However, cost-effective satellites with rapid development time are also required today, to complement the flagship scientific missions. Recently a program for small scientific satellites was started in ISAS/JAXA in response to this expectation, to find appropriate way to achieve flexible standard satellite bus. For this program with new philosophy, the AOCS architecture should be also reconsidered. Distributed architecture seems to be suitable approach, and SpaceWire is assumed to be an appropriate method to connect processor and distributed I/O modules. This paper introduces this new architecture using SpaceWire, with simple estimation of required performance for SpaceWire.

1 INTRODUCTION

The Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency (ISAS/JAXA) has been launched numerous scientific satellites for many years. These satellites were designed individually and optimized for each mission. However, "cheaper and faster realization of unique space experiments" is also requested today, to complement the flagship scientific projects. In response to this request, ISAS/JAXA has recently started a program for small scientific satellites series (Fig.1.) [1].

One typical approach to develop a satellite with low cost and short development time is to use so-called "standard satellite platform". However, if discussing on the scientific purpose, a mission and another one usually request different function and performance for satellites, thus it is not practical enough to apply completely fixed platform for scientific missions. More effective approach seems to apply "semi-order-made" platform, or a standard platform which has some flexibility within a defined range. For example, if a platform has two options for gyro sensor, each project can select more appropriate one for its purpose.

An important issue to develop such "semi-order-made" platform is to discriminate which part of the platform should be adaptive and which part should be fixed. It should be considered carefully from many aspects, however, this paper focuses the design of attitude and orbit control subsystem (AOCS), and especially focuses on the data interface between a computer and sensor/actuator components in AOCS. Following sections introduce our novel AOCS architecture for "semi-order-made" platform, which uses SpaceWire as data interface.

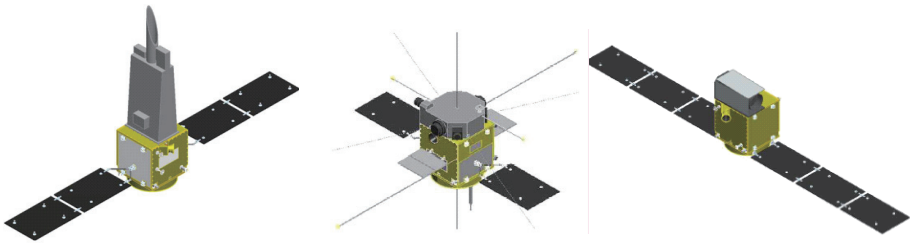


Figure 1: ISAS/JAXA's small scientific satellites series [1].

2 SpaceWire for AOCS Data Interface

2.1 Why SpaceWire?

Figure 2(a) shows traditional AOCS data interface architecture which has been used for many ISAS/JAXA scientific satellites. An AOCS computer unit consists of several boards, which are connected with some kind of local bus. A few (typically one) boards are for processing devices, and the others are to provide data interface for AOCS components. The design of the unit is usually optimized from the view point of resource, for example, several analog lines from AOCS components are connected to one A/D converter. Such design is effective to minimize parts numbers or power consumption, however, it is not so easy to adopt this unit to another scientific satellite. If some of the AOCS components are changed, then usually some boards of the unit must be re-designed. It is also not easy to add new board to the unit, since it changes the mechanical characteristics of the unit.

Thus another approach has been discussed for the series of small scientific satellites. The aim of this new approach is to divide the AOCS computer unit physically, into a unit independent of the AOCS components selection, and units dependent on components. The former is the processing unit, and the latter should be the interface units. From this viewpoint, new architecture indicated in Fig.2(b) seems to be suitable for "semi-order-made" AOCS. The components named "attitude control interface module (ACIM)" is a small unit which has a function to interface certain types of components. For example, "ACIM-IRU" has capability to interface some IRU (inertial reference unit), and "ACIM-RW" is able to interface certain type of reaction wheels (RWs). When an IRU is changed to another one, only "ACIM-IRU" must be re-designed. Processor unit or the other ACIMs can be exactly the same. In addition, this newly-designed ACIM ("ACIM-IRU") will be probably reused in another satellite, since a kind of AOCS components usually does not have very wide variety of products.

One thing necessary for this approach is an appropriate data interface between the processor unit and the ACIMs. This interface must behave just like local bus, thus following features are required:

- High speed connection. Throughput is not usually important, but low latency is strongly requested.
- Simple implementation. It should be simple enough to be implemented with only small FGPA or ACISs.
- Applicable for space use.
- Reasonable cost, low power consumption.

Considering these requested features, SpaceWire should be one good solution. Another candidate is the MIL 1553B interface, however, SpaceWire has advantages today such as link speed or cost.

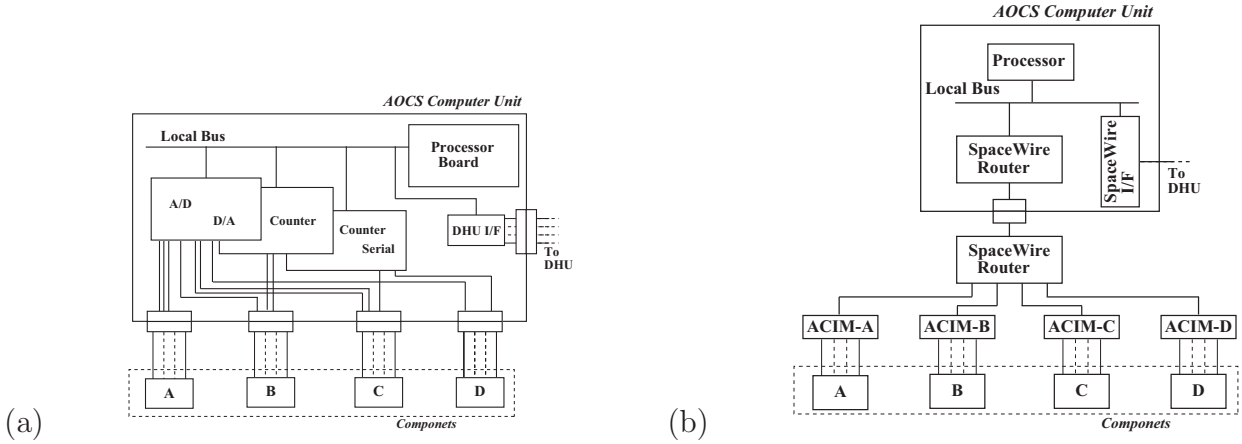


Figure 2: Conventional (a) and novel (b) AOCS data interface architecture.

2.2 Requirements for AOCS Data Interface

This subsection describes general requirements for a data interface from the AOCS design view point. Usually, AOCS consists of feedback controllers, to control attitude, reaction wheels speed, etc. For feedback controllers, latency of data interface is critical issues. Both for analog and digital controllers, latency behaves as so-called "dead time" for the closed loop system. It causes degradation of control performance, and in worst-case, might cause the system instability.

Typical feedback controller for ISAS/JAXA satellites' AOCS is a digital controller implemented as a software on a processor, with sampling or control frequency of 8Hz∼32Hz. Digital or discrete-time digital controller can be generally formulated as,

$$\boldsymbol{y}_j = C\boldsymbol{x}_j + D\boldsymbol{u}_j, \tag{1}$$

$$\boldsymbol{x}_{j+1} = A\boldsymbol{x}_j + B\boldsymbol{u}_j, \tag{2}$$

where $\boldsymbol{u}, \boldsymbol{y}$ are the input and output of the controller, $\boldsymbol{x}$ is the state variables of the controller, and A, B, C, D are the parameter matrices of the controller. If a discrete-time digital controller is designed with 8 Hz sampling frequency, it means that eq. 1 and eq. 2 are calculated every 1/8 [sec]. This sampling frequency of digital controller must be higher enough than the control bandwidth of the system. The control bandwidth indicates the system performance, such as robustness against the disturbance or response to the command input. Conventional AOCSs have control bandwidth of 0.01∼0.1Hz, and for conservative design, the sampling frequency should be at least 10 or 20 times higher than the control bandwidth. Therefore, conventional attitude feedback controllers have sampling frequency of some hertz. Minor loop, such as reaction wheel speed controller, or advanced attitude controller for agile satellite sometimes has sampling frequency of some tens of hertz.

Here we assume a digital controller which has sampling frequency of 64 Hz. As described above, this assumed frequency will be high enough even for some advanced or agile satellites in near future. Then the computational time for one period is $1/64 = 15.6$[ms], and the appropriate I/O time empirically seems to be less than 1-2[ms]. If the processor is connected to 10 AOCS components or ACIMs, each communication times should be less than 100-200 [μs]. To estimate the communication time, the typical data size is assumed to be 256 byte for one components or ACIM. One sample data size for usual AOCS components is usually not so much larger than this assumption. Based on our rough-order estimate, 256 byte read out from one ACIM takes about 130 [μs], when the SpaceWire is used with link speed of 48 MHz, with taking account of the overhead for SpaceWire and RMAP. Thus the SpaceWire and RMAP seem to have enough capability from the view point of link speed or latency.

Another important aspect is the synchronization accuracy of sensor sampling timing, such as

the timing between IRU and star sensors. For example, if some agile satellite changes its attitude with angular velocity of 1[deg/sec], then the timing uncertainty of 100 [μs] causes 0.36 [arcsec]. Some satellites today requires attitude knowledge better than 1 [arcsec], therefore, synchronization accuracy of about 10-100 [μs] might be required at most. SpaceWire seems to be able to provide this class of accuracy using time code mechanism.

3 SpaceWire for AOCS Ground Test Equipments

The discussion in subsection 2.1 is true of the ground test equipments. For series of small satellites developments, test system with slightly different configuration is necessary for each satellite. Therefore, flexible and cost-effective ground test equipments are requested. Figure 3 and 4 are the schematic drawings, for software development phase and for static closed loop test phase, respectively. Since the concept, "only change what must be really changed", is exactly the same for onboard AOCS system previously discussed, the details of this idea are omitted here. However, we believe that the idea described in this paper will prove its worth, only when the concept of flexible ground test equipments also becomes reality.

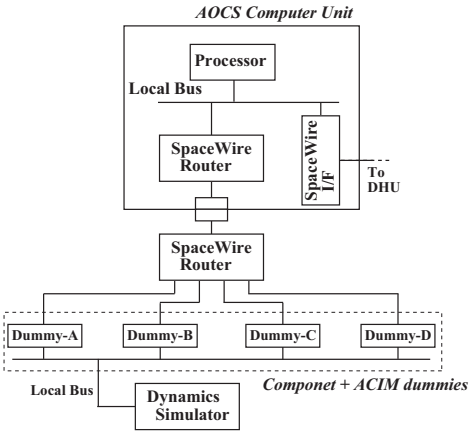


Figure 3: Configuration of ground test equipments for software development.

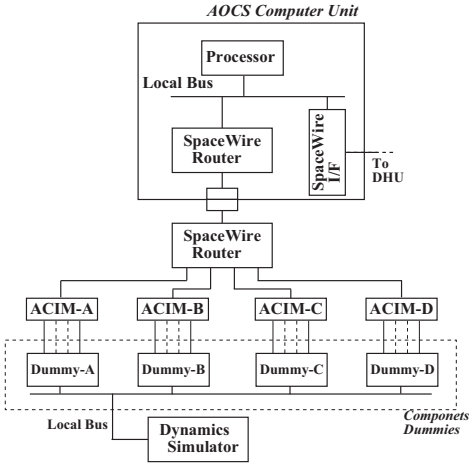


Figure 4: Configuration of ground test equipments for static closed loop test.

4 Conclusion

This paper briefly introduces an idea to use SpaceWire for AOCS data interface. Conventional Japanese scientific satellites were custom-made satellites, and the AOCS computer architecture also follows this philosophy. However, cost-effective satellites with rapid development time are also required today, to complement the flagship scientific missions, and thus another appropriate architecture should be taken into account. The basic discussion was described in this paper, however, it should be continued from various viewpoints. For example, also interface for telecommand should be discussed, and in that case, concept of QoS should be also applied.

The new approach introduced in this paper was first studied for the series of small satellites, however, soon found to be also applicable for middle or large class satellites, since the most severe constraints today is the cost, rather than the weight or the size. Thus it is considered to apply the same approach to the "ASTRO-H" satellite, which is ISAS/JAXA's middle class X-ray satellite.

References

[1] Seisuke FUKUDA, et. al. Flexible standard bus for isas/jaxa small scientific satellite series. In *Proc of the 26th International Symposium on Space Technology and Science*, Hamamatsu, Japan, 2008.