SpaceWire: Key principles brought out from 40 year history

Paul Walker, Barry Cook
4Links Limited
Bletchley Park, Milton Keynes, England MK3 6ZP; +44 1908 642001
paul, barry,@4Links.co.uk

ABSTRACT: SpaceWire was mentioned many times at SmallSat 2005, particularly in the sessions on standards and modularity. But there was no paper describing what SpaceWire is or the concepts behind it. This paper uses the evolution of SpaceWire over the last 40 years to describe the concepts and where they are being used, and from these suggest the opportunity that SpaceWire provides for the future.

INTRODUCTION

SpaceWire is a recent technology for space, having been standardized by ECSS (European Cooperation for Space Standardization) in January 2003[1]. Early versions of it are flying on several missions, and it is planned for use on many missions worldwide. As a simple interface that can be used for a wide variety of different purposes, SpaceWire appears to offer an enabling technology for the "Building Block Architecture" of the Vision For Space Exploration[2]. While recently standardized, SpaceWire has evolved over many years, following a few key principles and concepts that are the foundation of its wide application and use.

1960+ A MODULAR COMPUTER

In the 1960s, a computer would be built from several different boxes, such as processor, memory, disc controller and communications controller. One way to connect the boxes together was to use a simple standard interface between any of these boxes, so that they could each access the others independently of each other.

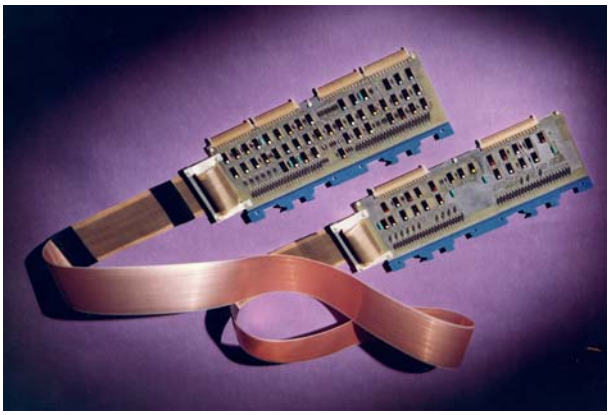

Figure 1. Standard Interface on Modular One Computer

The key concepts of this standard interface were:

- Keep the bus inside each box, so that the whole system is not sharing a single bus;
- Use an asynchronous interface, so that each box can run at its optimum speed and there is no need for global synchronization;
- Use a symmetrical interface, so that any box can be connected to any box;
- Have flow-control across the interface so that data is not lost even if buffers are full (but this may result in reduced performance if a communication is blocked).

These key principles resulted in a number of benefits:

- The system was scalable, so that systems could be built with any number of processors, memories, and peripherals;
- There were few constraints on the topology of the system, so that systems could be built with any shape as well as any size;
- Multiple units could be configured for redundancy and fault-tolerance;
- The system was truly modular, in that a huge variety of systems could be built from a comparatively small number of building blocks.

While the Modular One computer systems built with these interfaces were never used in space, they were used by the European Space Agency for Ground Support and Operations. The chips described in the next section were flown in space, on a number of missions.

1980+ SYSTEM ON CHIP, SERIAL INTERFACES

During the 1980s, it became clear that it would be possible to put a complete computer on a single silicon chip, including processor, memory, and interfaces. One

of the first examples of this was the INMOS transputer[3]. This had the conventional external memory bus similar to other microprocessors, but it also had four serial interfaces or "links" that inherited the key principles of the Modular One interfaces.

The block diagram in Fig. 2 is taken from early publicity material that INMOS produced for the transputer (the IMS T424), clearly showing the significance of the four serial links. Fig. 3 shows a packaged die of the later T800 floating-point transputer, with the four links on the left towards the top.

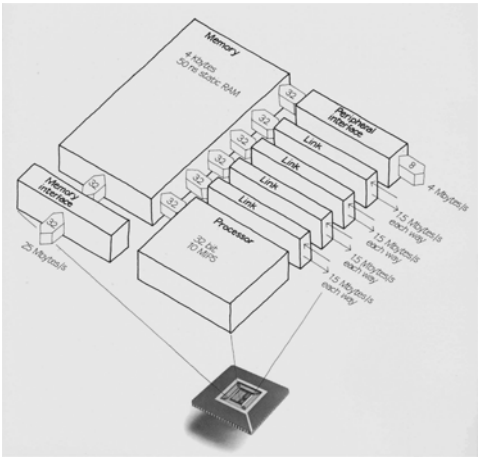

Figure 2: Block diagram of the transputer, with its four serial interfaces

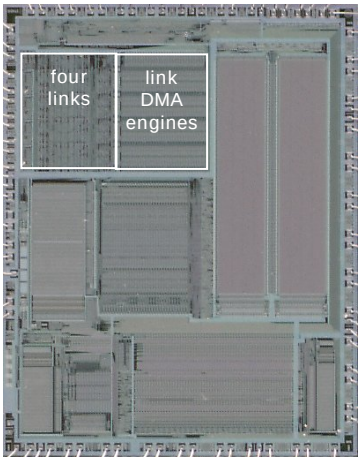

Figure 3: Chip photo of the T800 transputer, showing the area taken by the serial links

The cost benefits are clearly visible from Fig. 3. Overall, the four links, including the physical layer interface, all the serializing and de-serializing (SERDES) and DMA logic for each direction for each link, take up about the same space as the fixed-point processor. By comparison the on-chip RAM, the floating-point processor and the memory interface (including all its pins) each take up significantly more chip area.

At the time the transputer was introduced, a 10Mb/s Ethernet interface needed a chip-set of three chips, whereas a serial link needed around 2% of a single chip on the transputer and its DMA engine another 2%.

Performance of the early transputer links was modest, but at 20Mbits/s in each direction (full-duplex) a single link was well over twice the performance of an Ethernet connection. With the four links per transputer running full-duplex at 20Mbits/s, total serial throughput was 160Mbits/s per transputer.

As well as keeping the key principles of the Modular One interfaces, the transputer links added the following:

- They were serial interfaces, to reduce pin count and to simplify connections between chips;
- They used DMA to access the transputer's memory, with very low processor overhead per packet.

TRANSPUTER SERIAL LINKS IN SPACE

The space industry recognized the potential of the transputer and its links for building fault-tolerant networks on-board spacecraft.

Missions included the Cluster group[4] from ESA, many satellites from SSTL[5], and the SOHO[6] collaboration between ESA and NASA. In fact the transputers used in these missions were not specifically designed as Rad-Hard, but they were from batches selected for radiation tolerance and designed into fault-tolerant networks.

The SOHO satellite continues to send back images of solar corona discharges, such as the image in Figure 4.

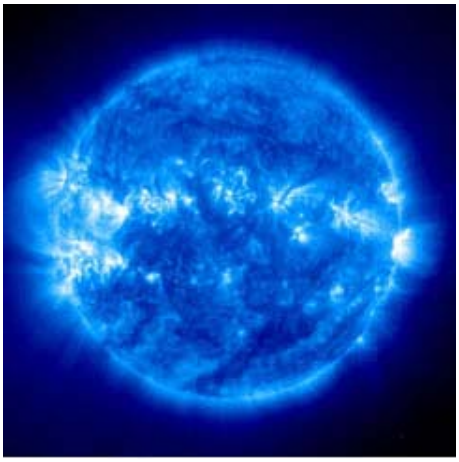

Figure 4: Image taken by the EIT instrument on SOHO

MODULARITY

In the early days of the development of the transputer, it was found that a useful way to explain the ideas was to compare the transputer with toy building blocks such as Lego™ and K'Nex™ [7]. These use a very simple standard interface that can be used to connect a wide variety of different building blocks, in order to build an even wider variety of constructions. The serial links of the transputer were such a simple and easily usable interface, and they encourage modularity.

The opportunity was taken to propose a standard transputer module, or TRAM, which used the serial links as their interface. These were printed circuit boards about half the size of a credit card, with just sixteen pins. In effect they were 16-pin Dual-Inline-Packages (DIPs) with 3.3" between the pins instead of the conventional 0.3" between the pins. These modules were very popular and were made by INMOS and by a number of other companies.

1990+ TRANSPUTER LINKS TO IEEE 1355

Towards the end of the 1980s, a new generation of the transputer was planned, taking the links to 200Mbits/s and adding some important new principles:

- Adding a minimalist packet protocol, consistent with the general move towards packet communication and switching;
- Adding a network protocol so that the packets could be routed through a network of routing switches;
- Adding virtual channels, so that a variety of different communications can share the same physical links.

The TRAM standard had been popular as a way to construct systems inside a box. The new 200Mb/s links provided the opportunity to create a standard for connections between boxes, and the author proposed what was initially an internal standard in the late 1980s. Colleagues at INMOS, together with other contributors in Europe, took this forward to create the IEEE 1355 standard. To keep the standard simple, we left out the network and virtual channel protocols, but all the previous principles that have been outlined were included in IEEE 1355.

Notable among the contributors were CERN, who built a large test system with 1024 links, over which they ran a soak test for three months, logging 10^{17} bits transferred without a data error on a link (At one point during the test, a thunderstorm upset the computer and network that were controlling the test, but there was no failure on the links.)[8].

Also among the contributors, even in the early 1990s, was Dornier SatellitenSysteme (DSS, subsequently EADS-Astrium, in Munich).

The IEEE 1355 standard was confirmed in 1995, after which the European Space Agency and a number of other organizations in the space industry joined the activity.

For what at the time were probably correct commercial and political decisions, the new transputer and the 1355 standard were abandoned by the company that had taken over INMOS. The standard was used by Canon, who needed to adapt some aspects of the standard for a networking application, and the original standard had not been designed for space and so a new standardization activity was launched by the European Space Agency. This activity became SpaceWire.

IEEE 1355 AND EARLY SPACEWIRE IN SPACE

During the development of the SpaceWire standard, there was clearly an interest in using the 1355 standard, or drafts of the SpaceWire standard, for space applications. EADS-Astrium Munich commissioned a chip that was available in a RAD-Hard version, and this chip is flying on Rosetta[9], on Mars Express[10] and on Venus Express[11]. Early versions of SpaceWire are also flying on SWIFT[12] and on other missions classified for commercial or other reasons.

2000+ SPACEWIRE

Compared with IEEE 1355, the SpaceWire standard:

- Uses LVDS rather than PECL, for low power
- Uses space qualified connectors and cable
- Corrects an initialization bug in 1355
- Removes some ambiguities in 1355
- Includes a simple Network Layer protocol
- Adds Time Code distribution

Apart from these changes, the SpaceWire standard embodies the key principles that have been outlined:

- Bus kept inside each unit, not over entire system;
- Serial interface;
- Asynchronous interface;
- Symmetrical interface;
- Flow-control across the interface;
- Minimalist packet protocol;

And these qualities, as before, bring the benefits of scalability, topological flexibility, fault-tolerance and modularity

The standard is cleanly layered, with minimal overlap or interaction between the levels. The levels defined are:

- Physical level: two signal pairs in each direction, PCB traces, connector and cable;
- Signal level: LVDS including failsafe, terminations, Data-Strobe signal encoding on the two pairs, signalling rate, skew and jitter;
- Character level: Data characters, Control characters, Time Codes, parity, character(s) to be sent at initialization or after error, host interface encoding;
- Exchange level: Normal Characters (that are passed through the network) and Link Characters (that are local to a single physical connection), flow control, clock recovery, initialization state machine, errors and error recovery, Time Code distribution;
- Packet level: destination address, cargo, end-of-packet markers;
- Network level: Wormhole routing, path addressing, logical addressing, header deletion, group adaptive routing, how to do broadcast or multicast, network errors and recovery

This paper will summarise a few of the main characteristics, particularly those that are different from some other networking standards:

- Data-Strobe encoding
- Low-level flow-control
- Packets
- Packet routing
- Time Codes and their distribution

Data-Strobe encoding

There is a need in any communication system for a means of recovering the clock from the received signals. In long-distance communication, this tends to be with a phase-locked loop per channel, which would be possible for space but which needs analog circuitry that is undesirable in space electronics. An alternative is to send a clock signal on a separate wire, but this has tight demands on skew between the signals. SpaceWire uses a Strobe signal on a separate wire, which is Gray-coded with the signal wire so that for each bit transmitted, there is a transition on either the Data or the Strobe signal. This still needs the skew to be controlled, but is more relaxed in this respect than separate clock and data. The technique was originated in IEEE 1355 and was subsequently adopted by IEEE 1394/FireWire. It is one of the contributing factors in SpaceWire being a simple, digital, circuit, without needing analog electronics.

Low-level flow-control

Flow-control is often seen as a high-level protocol, and indeed for long-distance communication needs to be so. The lack of flow-control at a low level, however, requires buffers large enough that they (almost) never overflow. SpaceWire permits low-cost circuits with small buffers, and the flow-control ensures that data is preserved and that the buffers never overflow. Having larger buffers than the minimum permitted improves overall network performance, but the flow-control allows implementations of SpaceWire that can have less logic and less buffering than conventional RS232/422 UARTs, even though SpaceWire runs orders of magnitude faster than these UARTs.

Packets

SpaceWire uses minimalist packet format, with header, cargo, and packet termination. For a point-to-point connection not via a routing switch the header can be zero length; for a routed packet, the header is a destination address that can be as long as necessary. The cargo can similarly be as long as necessary, and no limit is defined in the standard. In practice, most systems will benefit from imposing a form of Maximum Transfer Unit (MTU) to prevent a long packet blocking other traffic in the network. The packet termination is a single control character, either End-of-Packet (EoP) or Error-End-of-Packet (EEP).

After the standard was issued, it was agreed to include a protocol identifier (PID) as part of the header, between the destination address and the cargo. As in other standards such as Ethernet and Internet Protocol, the PID allows a variety of different higher-level protocols to interoperate on the SpaceWire network without interfering with each other.

The minimalist packet protocol of SpaceWire provides what is absolutely necessary and no more. If extra information is required in a header, such as the source of the packet, a checksum, or a protocol to be encapsulated on SpaceWire, these can all be added. All that is added, however, needs to be generated and checked for each packet, which can impose substantial delays in processing each packet. The simple raw SpaceWire packets provides a very efficient communication system with very low processing overheads as well as low overheads on packets.

Packet routing

SpaceWire can be used with or without routing switches, and satellites can include point-to-point connections as well as a network with routing-switches.

When using routing switches, SpaceWire packet switching uses "Wormhole Routing" so that the front of a packet can have left the routing switch before the end of the packet has arrived.

The SpaceWire standard requires that routing switches provide what the standard calls Path Addressing, and permits them to provide what it calls Logical Addressing. In each case, the first data character of a packet seen by the routing switch is used as a routing header to determine which output port of the routing switch the packet is routed to.

In Path Addressing, values of the first data character from 1 to 31 result in the packet being output to port 1 to 31 respectively. The special value of zero results in the packet being used internally by the configuration/management port of the routing switch. After the character has been used to address a particular output port, the character is no longer required and so is deleted.

In Logical Addressing, values of the first data character of a packet are used to index a look-up table to determine the output port. In this case the character is not normally deleted, as the same character can be used in several routing switches to steer a route through the network. For small networks such as tend to be used on satellites, logical addressing can provide an exceptionally low overhead for routing the packets.

Time Codes and their distribution

It is useful for all the subsystems on a satellite to have a reasonably consistent view of time, and SpaceWire provides a means of distributing such a consistent view. Time Codes are special sequences of characters which take priority over the normal data in a packet and are distributed to all nodes in the SpaceWire network. A small amount of jitter is normally introduced, both in the generation and distribution of Time Codes, resulting in a few microseconds variation in the view of time from different nodes in the network. A scheme has been proposed[13] that is completely compatible and interoperable with the standard, where the jitter in Time Code generation and distribution can be reduced to a few tens of nanoseconds[5].

WHERE SPACEWIRE IS BEING USED

SpaceWire is planned for use on a wide variety of different missions, throughout the world. The European Space Agency plans to use SpaceWire for most, if not all, of its future missions. A number of national missions, such as Taiwan's Argos satellite, are using SpaceWire. Key US missions are the James Webb Space Telescope[14], the Lunar Reconnaissance Orbiter[15], and GOES-R[16].

At last year's Small Satellite conference, AFRL reported interest in SpaceWire, because they saw it as an enabling technology for the modularity, scalability and reconfigurability required for Responsive Space.

THE FUTURE, 1: HOW SPACEWIRE WILL DEVELOP

The SpaceWire Working Group has already defined the Protocol Identifier so that multiple protocols can interoperate on a SpaceWire network, and has defined a Remote Memory Access Protocol (RMAP). A number of other protocols are being defined, particularly to encapsulate CCSDS and IP packets in SpaceWire, and we can expect to see more such encapsulation. A new protocol for SpaceWire has been developed in the US for reliable data transfer, like TCP but much simpler than TCP because it does not have to run over a global network with billions of nodes.

Last year's Small Satellite exhibition included two examples of SpaceWire running at 400Mbits/s or faster, whereas must current uses are between 10Mbits/s and 200Mbits/s. The current RAD-Hard silicon imposes limits on the speeds that can be used but new ASIC chips, and PHY chips which handle just the high-speed front end, will make it easier to use SpaceWire at higher speeds.

A current ESA project is SpaceFibre, which aims to take the SpaceWire protocols up to between 1Gb/s and 10Gb/s, using a different physical layer that might include versions for both fibre and copper.

New capabilities will evolve. NASA have suggested extending the use of Time Codes such as to include, for example, a 1pps signal for time and a trigger signal for a number of instruments. Such suggestions provide enhanced capability but there is concern in some quarters if they would not be interoperable with chips and instruments that have been built to the standard. It is clear that improvements will be more welcomed if, like the low-jitter Time Code proposal, they are fully interoperable with all existing devices.

Some years ago, the author's company demonstrated a Plug and Play system over SpaceWire[17], which was well received. It was argued at the time that satellites are fixed configurations with no need for Plug and Play. Once such a plug-and-play capability is used, however, it can be used for the unexpected changes in system configuration and hence can assist Fault Detection Isolation and Recovery (FDIR). There may also be benefits from plug-and-play for the manned space program, where configurations are expected to change over time. And for Responsive Space, launching a satellite in a few days from mission definition means there is no time for system configuration or software development. The system must just plug together and work, so plug and play is necessary and AFRL together with NRL are proposing a plug-and-play system using SpaceWire (with USB)[18].

THE FUTURE, 2: HOW USE OF SPACEWIRE WILL EVOLVE

Most of the early uses of SpaceWire have been as medium- to high-speed replacements of point-to-point links such as RS422. A typical configuration would be to connect an imaging instrument to a DSP processor, or to cross-connect a pair of instruments to a pair of processors.

To some extent, this use of point-to-point links without routing switches has been because there have not been Rad-Hard routing switches available. These are being developed, however, by ESA, by NASA, and by a number of companies, and we can expect to see them used to construct simple networks.

NASA's James Webb Space Telescope is using routing switches to build quite a large but simple network.

Routing switches can impact fault-tolerance, allowing different parts of the system to tolerate different numbers of faults. For example a daisy-chain (without the ends joined together or any cross-connections) does not tolerate some single faults. Connecting the two ends of the daisy-chain so that there is a ring is a simple way to provide tolerance of a single failure, whether the failure is in a node or a link. With three links per node, networks can be constructed which tolerate two failures, and in general, for n links per node, networks can be constructed to tolerate n-1 failures.

Many of the SpaceWire systems being built are modelling earlier systems based on a bus and on a global memory access model. Hence the first protocol to be defined is the memory mapping protocol, RMAP. For many applications this model is appropriate and minimal cost. For other applications, a network model such as Ethernet or the Internet is appropriate. These different models and their protocols can happily co-exist over a SpaceWire network, just as private Microsoft and other protocols co-exist with TCP/IP over Ethernet.

So far, apart from the work led by AFRL, the full benefits of modularity offered by SpaceWire are not being realized. There is a growing consensus that SpaceWire is the one interface standard that seems to come closest to meeting the widest variety of application needs for the space industry, and so it must be seen as a prime candidate as the interface for NASA's Building Block strategy. To what extent the details of modularity will be internationally agreed or will be private within national organizations is yet to be seen.

CONCLUSIONS

SpaceWire has been an outstanding success in international collaboration, which has resulted in its use worldwide.

While apparently new technology, SpaceWire has a legacy going back 40 years, and a significant element of that legacy has proved itself in space missions that have been flying for many years.

The legacy of a simple interface that can be used for almost anything has been retained by preserving a number of key principles. These key principles provide modularity, scalability, and reconfigurability, and are far more important than the implementation details.

Early uses of SpaceWire have been evolutionary and have not therefore exploited the full benefits that might be available from using SpaceWire. As more experience and confidence is gained, we can expect more of the benefits to be realized.

Benefits will be realized from evolution of SpaceWire itself, and also from evolution in the uses of SpaceWire.

This paper has suggested some possible directions for such evolution — it will be interesting to see if the predictions come to pass.

REFERENCES

1. European Cooperation on Space Standardization, ECSS-E-50-12A SpaceWire - Links, nodes, routers and networks, 24 January 2003
2. NASA, The Vision for Space Exploration, February 2004

3. INMOS Limited, IMS T424 Transputer Reference Manual, Bristol, England, 1985
4. http://www.sussex.ac.uk/space-science/missions.html#CLUSTER%20II
5. http://www.ee.surrey.ac.uk/SSC/CSER/UOSAT/missions/posat1.html
6. http://sohowww.nascom.nasa.gov/
7. Paul Walker The Transputer: a Building Block for Parallel Processing. Byte Magazine, Volume 10, Number 5, May, 1985.
8. S Haas, D A Thornly, y Zhu, R W Dobinson and B Martin, The Macramé 1024-Node Switching Network Microprocessors and Microsystems, IEEE 1355 Special Issue, V21, Nos 7,8, 30 March 1998
9. http://sci.esa.int/rosetta/
10. http://mars.esa.int/
11. http://sci.esa.int/venusexpress/
12. http://swift.gsfc.nasa.gov/docs/swift/swiftsc.html
13. Barry Cook, Reducing time code jitter, International SpaceWire Seminar, ESTEC, November 2003, available at http://www.4Links.co.uk/reducing-time-code-jitter.pdf
14. http://www.jwst.nasa.gov/
15. http://lunar.gsfc.nasa.gov/missions/
16. http://science.hq.nasa.gov/missions/satellite_67.htm
17. Paul Walker, Barry Cook, The 4Links Plug and Play SpaceWire Demonstration, from http://www.4Links.co.uk/4L-PnP-SpaceWire-Demo.pdf
18. Jim Lyke, Scott Cannon, A Plug and Play system based on USB for spacecraft components, 19th AIAA/USU Small Satellite Conference, Logan UT, August 2005