DOI 10.5162/ettc2020/3.2

Intelligent SpaceWire Router with Multilevel Priority Arbitration and Multicast Packet Transmission

G. Skvarč Božič[1], *M. Plattner*[1]
[1] *Max Planck Institute for extraterrestrial Physics, Giessenbachstr. 1, 85748 Garching, Germany*
gaskvarc@mpe.mpg.de

Abstract:
SpaceWire (SpW) is a major communication standard used to connect systems in spacecrafts. It supports point-to-point and network connections. For the Wide-Field-Imager payload onboard the ESA ATHENA satellite, the next generation European x-ray observatory, we have designed an intelligent SpW router. It supports dynamic multilevel priority arbitration based on a maximum finder circuit and multicast packet transmission. The SpW router was implemented as a VHDL component to have a flexible design and a possibility to include it in other projects. The implemented SpW router has an RMAP supported configuration port for remotely configuring the router.

A test setup was designed to characterize and test the SpW router functionality. A routing delay and a packet router latency were determined. Simultaneous packet traffic from different sources with the same and different priority levels was emulated to demonstrate the arbitration functionality. Based on performed tests, the implemented router achieved a satisfying performance with the capability of handling transmission rates up to 200 Mbps. Moreover, the designed router increases payload system functionality, eases the design of more complicated networks, and enables a new way to design redundant systems with its multicast transmission support.

Keywords: SpaceWire, SpaceWire router, multilevel priority arbitration, multicast packet transmission, VHDL

Introduction

When designing a complex system, there is always a need for transferring data from one subsystem to another. Therefore, like for any other application, there are defined communication protocols for satellite onboard data-handling systems. Among those is SpaceWire (SpW), a communication protocol supporting point-to-point and network connections. It is used by several agencies, namely by ESA, NASA, JAXA, and Roscosmos for current and future space missions.

In the scope of ESA's ATHENA mission, the Wide Field Imager (WFI) instrument uses SpaceWire as the onboard data-handling system depicted in Fig. 1. Five Detector Electronics (DE) are connected through a SpW router (SWR) to a Central Processing Module (CPM) in the Instrument Control and Power Distribution Unit (ICPU). The CPM compresses the data from the detector electronics and sends it to the mass memory of the spacecraft (S/C). It is responsible for interpreting the telecommands from the spacecraft with which it controls the ICPU subsystems and detector electronics.

Despite several SpW router devices available on the market from which two are directly supported by ESA, a ten port SpW router AT7910E (Atmel) and an eighteen port SpW router GR718 (Cobham). Both support two-level, high, and low, priority arbitration. And a few commercial products such as Flexible SpaceWire router with 2 to 32 ports from 4Links, 4-port SpaceWire router UT2000SpW4RTR with round-robin output arbitration from Cobham, SpaceWire router from NEC Japan, and an open-source six-port SpaceWire router IP core with round-robin arbitration from Shimafuji Electric, inc. We saw the need for the improved capability of a SpW router as there is no multicast transmission available in the mentioned devices.

The ICPU remote and main unit are designed in cold redundant, single-point failure configuration. Cross-strapping between the units is pos-

sible and optional. Nevertheless, as an example use case for the SpW router, the multicast feature is expected to simplify the design of cross-strapping between nominal and redundant systems depicted in Fig. 1. Offloading the required logic from DEs to the SpW router. For example, instead of sending two identical packets from DE, only one multicast packet is needed. Also, implementing a multilevel priority arbitration adds a degree of flexibility where each application protocol can be assigned a different priority level. Moreover, priority levels can be changed during operation if needed.

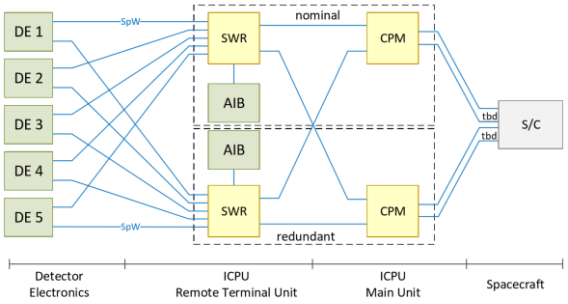

Fig.1.WFI SpW data-handling system.

SpaceWire standard

The SpaceWire standard is a standard for high-speed links and networks for use onboard spacecraft, easing the interconnection of sensors, mass-memories, processing units, and downlink telemetry sub-systems. SpaceWire was developed under ESA in the late ’90s and formally standardized by European Cooperation for Space Standardization (ECSS) to provide space users with directly applicable specifications. SpW standard is very similar to Myrinet, a local area network (LAN) for computer clusters, especially regarding data and control characters. Both use 9-bit data characters and several control charters to control the flow of data through a link [1].

SpW, as a communication standard, has a defined protocol stack. The SpW protocol stack is composed of a Network layer, a Data Link layer, an Encoding layer, and a Physical layer. Compared to the well-known OSI model, the Network and Physical layer of the SpW protocol stack match the OSI model, whereas the Data Link and Encoding Layer both fall under the Data Link layer in the OSI model as can be seen in the Fig. 2.

The point of interest for this paper is the Network layer on which the SpW router operates. Other lower protocol layers are fulfilled by the SpW Coder/Decoder (CODEC) in router ports. No furthers explanation for these layers is provided since this is out of the scope of this paper. The focus is on the router architecture and the control logic associated with it.

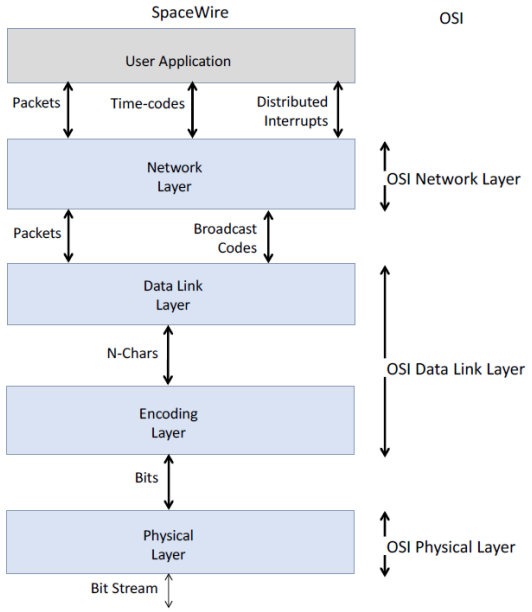

Fig.2.SpaceWire protocol stack compared to the OSI model [2].

Devices on a Network layer operate on packets. SpW packet structure depicted in Fig. 3 consists of a header, cargo, and end of packet marker (EOP).

Fig.3.SpaceWire packet structure.

The header of a SpW packet holds the address of the destination, and it is used to route a packet through a SpW network correctly. The Cargo section of the packet contains actual data which can be encapsulated in any user-defined protocol. Each packet is ended with an end of packet marker, which notifies a network device about the end of transmission. Moreover, it provides information about whether a packet transmission completed without any errors.

In theory, the SpW network is considered a switch-based network like Autonet, Myrinet, and ServerNet. Switch-based networks are composed of links and a set of switches, which together usually form an irregular network topology. Each switch has a set of ports where a subset of these ports is connected to other switches, a subset to network nodes, and some are left open. The network supports bi-directional full-duplex links with support for multiple links between two switches. However, the only guarantee of such networks is that the network is connected. They provide the flexibility that is required in LANs and the necessary scalability for designing scalable systems. However, their

irregular topology results in complicated routing. Therefore, in switch-based networks, no routing algorithm can be employed on a router level. There are two possible routing techniques. One of them is source routing. A source provides the packet's destination in terms of a path that needs to be taken through a network to reach its destination. And distributed routing where a routing table, which contains the routing information, is added to each switch in the network. Before any packets are transmitted, a network mapping algorithm needs to be executed to populate the routing tables.

Based on the above-described routing techniques, there are two primary addressing mechanisms in the SpW standard. One is called path addressing, and the other one logical addressing. Path addressing, as the name suggests, specifies the whole path that the packet needs to traverse to reach its destination. Therefore, the header of a path addressed packet consists of multiple characters where each character specifies one turning point in the network. Logical addressing, on the other hand, requires only one character which uniquely identifies a destination. However, for logical addressing to work routers in the network need a routing table where each routing table entry maps a corresponding logical address to one of the output ports or multiple output ports in case of multicast.

The purpose of the router is to connect multiple nodes and to route packets using the wormhole switching technique from any input port to any output port based on information within the header. A packet can be forwarded through two or more ports if a multicasting feature is provided. The number of external ports in the router is limited to 31. Whenever two or more arriving packets have the same output port as the destination, arbitration is needed to resolve the output port contention. Together with low-error rate, low footprint, low-cost, low-latency, full-duplex, point-to-point links, they form a SpW network. Providing high-speed (2 Mbps to 200 Mbps), bi-directional, full-duplex connection between two network nodes.

Based on SpW standard specifications, a SpW router shall comprise of[2]:

- One or more ports that interface to the SpW network.
- A switch matrix which connects an input port to an output port.
- A routing table which, together with the leading byte of a packet (header), determines through which port a packet is forwarded.
- A configuration node that is accessible through the port 0 (configuration port) and enables configuration of the router and port parameters.

The SpW standard includes some additional features which are not listed here since they were not implemented in the current version of the presented SpW router. Mainly to simplify the design and to shift the focus on to the control logic.

A configuration node of a router was mentioned. There is a defined communication protocol for accessing the memory of a remote network node called Remote Memory Access Protocol (RMAP). It can be used for configuring and updating the router and port parameters from anywhere within the SpW network [3].

SpaceWire Router Architecture

SpW router architecture presented in this paper is based on a generic router model depicted in Fig. 4, router architectures of currently available devices mentioned in the introduction, and Network-On-Chip routers presented in [4, 5]. Key components of a router are [6]:

- **FIFO buffers**. They are used for storing messages in transit. They are present in input and output ports. Size of which depends on the employed switching technique.
- **A switch**. Physically connects input ports to output ports. Fully connected in high-speed implementations.
- **Routing and arbitration unit**. Responsible for executing a routing algorithm, forwarding incoming messages to the right output port, setting the switch, resolving conflict regarding simultaneous access of an output port. Also known as control logic and switch allocator.
- **Link Controller (LC)**. It controls the flow of messages over a physical link.

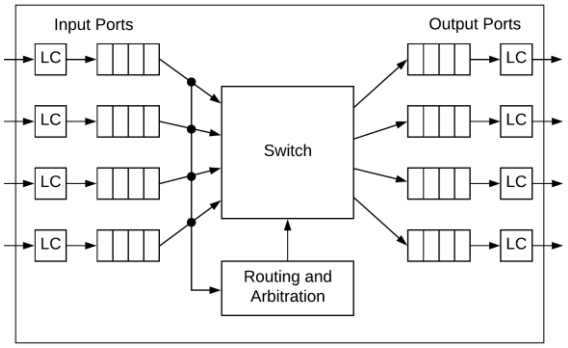

Fig.4.Generic router model [6].

The SpW router uses a wormhole switching technique as it enables the use of small input and output buffers. Considering the router is implemented as a VHDL IP-core, small buffers are desired. One consequence of the wormhole switching technique is that packets can span through multiple routers at once, which can lead to blocked paths and, in the worst case, a deadlock. Certain preventive measures can be implemented on the router level, such as packet timeout. However, a degree of caution should be taken when designing a network, including wormhole-based routers.

Considering specifications from the SpW standard, a few additional components were added to the generic router model to form the complete SpW router architecture depicted in Fig. 5.

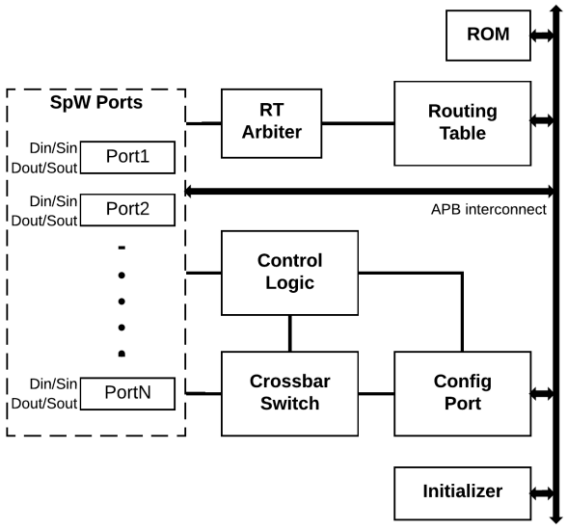

Fig.5.SpW router architecture.

One desired feature is the possibility of remotely configuring the SpW router. For accomplishing this, a configuration port or node, which is accessible via port 0, is added. The purpose of the configuration node is to create an interface between SpW and internal interconnect. Its task is to decode and encode RMAP packets. When an RMAP command is decoded, it is translated into corresponding internal interconnect command. If decoded RMAP command requires a response (e.g., read command), then data from the internal bus is encoded into the RMAP response packet. The core logic of the RMAP decoder was taken from an open-source router implementation from Shimafuji Electric Inc. [7]. It was adapted for the custom control logic, different internal interconnect, and the RMAP packet Cycle Redundancy check (CRC) calculation was changed from a look-up table implementation to on the fly calculation.

The configuration port is connected to all router register via the Advanced Peripheral Bus (APB). All registers and the routing table are memory mapped to comply with the RMAP standard. APB was selected as it is simple to implement, and there is no need for a higher performance bus since only register read/write operations are performed. The data width of the interconnect matches the width of the registers. Therefore, only one bus access is needed when manipulating the register data. It is true that the routing table, which covers a larger memory space, is connected to the same bus. However, this does not justify the use of a more complex bus as the routing table is in usual cases accessed only by its entry fields, which also match the APB data width.

All registers are implemented with fabric D flip-flops; therefore, the content of registers is lost after power loss. The same goes for the routing table, which is based on the inferred SRAM cells of the target FPGA device. As an option, if the target device has an included Read-Only Memory (ROM), the router can be initialized to a default state after a power cycle or reset of the device. Initialization is done by the initializer block, which reads data from the ROM and writes it to an appropriate register or routing table entry. If ROM is not present on the target device, the SpW router must be remotely initialized using RMAP before sending any normal SpW packets.

The routing table fulfills the requirement for SpW logical addressing. Moreover, it holds the information about the priority level of a logical address and some additional control flags. At this point, it is composed out of three data banks of size 256x32-bit with the possibility of expending the number of banks. Each bank represents a different routing table entry field. A destination field which is used in normal logical addressing meaning the logical address is mapped to one output port. A multicast set field that maps one logical address to multiple output ports when multicast packet transmission is enabled by the multicast enable flag. A control field where the 8-bit encoded priority level of a logical address, logical address deletion flag, and group adaptive routing enable flag are stored. The first 32 routing table entries are reserved for direct external port addressing when the path addressing is used. With this, there are 223 available logical addresses left plus the reserved logical address 0xFF. Logical addresses directly map to the addresses of the routing table entries; therefore, no additional address decoding is needed when accessing the routing table. The header of an incoming SpW packet is directly used as an address applied to the routing table. Routing table entry fields are 32 bits wide, which matches the SpW

standard limit of 31 external ports plus the port 0. This way, each bit position corresponds to the external port number with the same number as the bit position.

The routing table is a common resource for all input ports. Therefore, all requests are processed by an arbitration unit, which resolves the contention for the routing table access. Since the priority level of a packet is unknown before accessing the routing table, it makes sense to employ a fair arbitration such as round-robin.

SpW ports are the link controllers of the SpW router. Each SpW port is composed out of a SpW CODEC, which is responsible for maintaining an active link between two network devices. And a packet processor that is responsible for extracting the routing information from an incoming packet, communicating with control logic, and controlling the flow of incoming packets.

All SpW ports and port 0 are connected through a fully connected non-blocking crossbar switch. Full connectivity allows routing packets from any input port to any output port. Non-blocking characteristic allows multiple connections at once as long as an output port is not the same destination for multiple input ports. When there are no active requests for an output port, it is disconnected from all input ports.

Crossbar switch allocation is controlled by the control logic. Control logic is divided into two subcomponents, an arbitration unit that handles requests from input ports. And a switch allocator that generates correct control signals for the crossbar switch based on the given grant signals from the arbitration unit. Each output port is associated with its priority arbitration and switch allocator logic.

Arbitration in the control logic is designed as a two-stage arbiter inspired by the work presented in [5]. It follows a return-to-zero four-phase handshake protocol for the request and grant signal pair. There is no dedicated release signal; the de-asserted request signal releases the output port. The control logic itself is designed as a three-stage pipeline (see Fig. 6). In the first pipeline stage (Stage0), the request and priority signals are captured. In the second pipeline stage (Stage1), the priority signals are masked with the request vector to process only priority levels of active requests. The highest priority is determined with a k-bit maximum/minimum finder circuit based on Array-Based Topology (AbT), a fast and efficient circuit topology of a max/min finder circuit [8]. When only one high priority request is found, the arbitration process is complete, and the one-hot encoded position of this request (Addr OH) is passed to the grant signal generator in the next pipeline stage (Stage2). After that, the grant signal is passed to the output grant signal on the next rising clock edge.

A problem occurs when there are multiple requests with the same highest priority (among current active requests). If that is the case, requests with the same highest priority are isolated and passed to the second arbitration stage in the third pipeline stage (Stage2) of the control logic. Since these requests have the same priority, a fair round-robin arbiter determines which of those receives a grant signal. At the end of the pipeline, a multiplexer selects the appropriate grant vector based on whether there were multiple requests with the same highest priority or not.

Fig. 7 illustrates the above-described arbitration process in case of packets with matching priority. The example is divided into four frames. In the first frame, tow packets are waiting to be forwarded through the same output port o2.

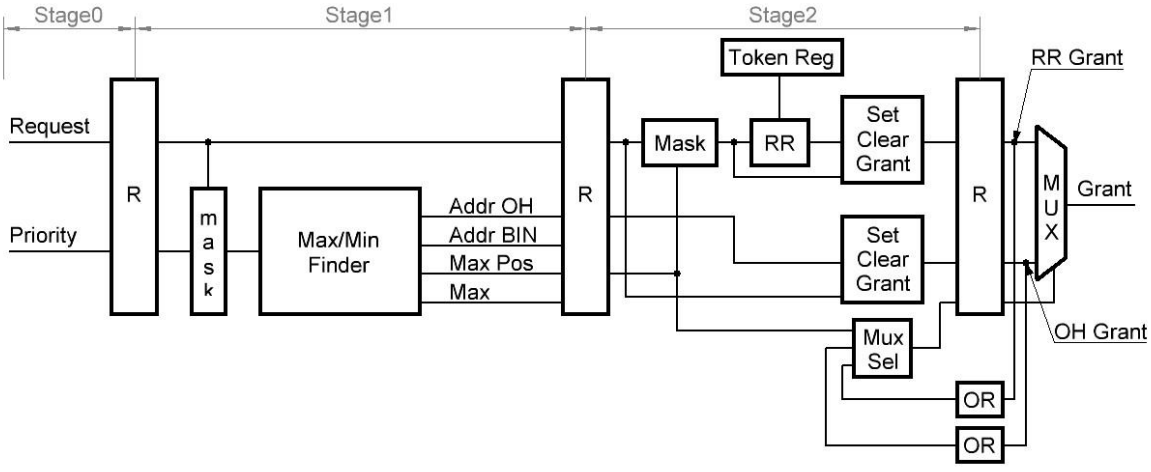

Fig.6.Control logic pipeline [9].

Meaning round-robin arbitration takes place. The last input port to have access to the output port o2 was the input port i4. Therefore, in the second frame, the input port i1 is selected, as it is the first one in line based on the previous access. Meanwhile, during the transmission, another packet with the same address arrives at the input port i2. In the third frame, input port i2 is selected even though the packet arrived later that the one on input port i3. Because of the round-robin arbitration, the port i2 is higher in line then the port i3. Finally, in the fourth frame, the input port i3 is granted access to the output port o2.

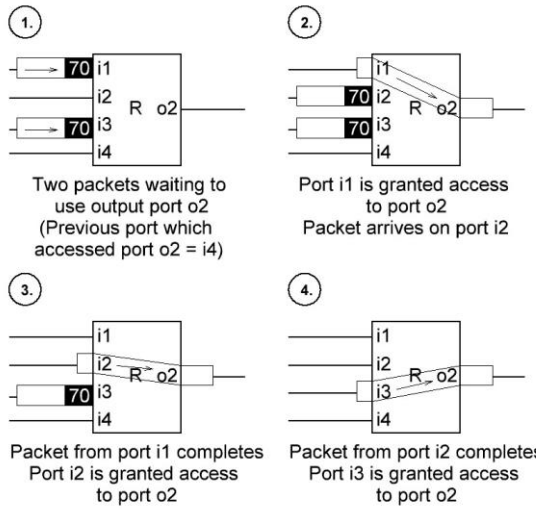

Fig.7.Arbitration of packets with matching priority [9].

Fig. 8 illustrates another arbitration example this time with packets with different priority levels. The example is divided into six frames where packets with logical address 35 have higher priority than packets with logical address 70. The first frame in this example is the same as in the previous example. However, in the second frame, during transmission of a packet from the input port i1, a packet with higher priority arrives at the input port i4. Therefore, in the third frame, input port i4 is selected as the next port to access the output port o2 after the completed transmission of a previous packet. During the transmission of the packet from the input port i4, another lower priority packet arrives at the input port i1. In the fourth frame, no high priority packets are present at input ports. Previous input port with access to the output port o2 when a packet with address 70 was present, was the input port i1. Meaning, the first input port in line is port i2. However, there is no packet waiting on this port. The first place is passed to input port i3, which is selected, and the packet is forwarded to the output port o2. In the fifth frame, after the previous packet has completed transmission, the input port i1 is selected. The packet from the input port i1 is forwarded to the output port o2. In the sixth frame, the last packet has completed transmission.

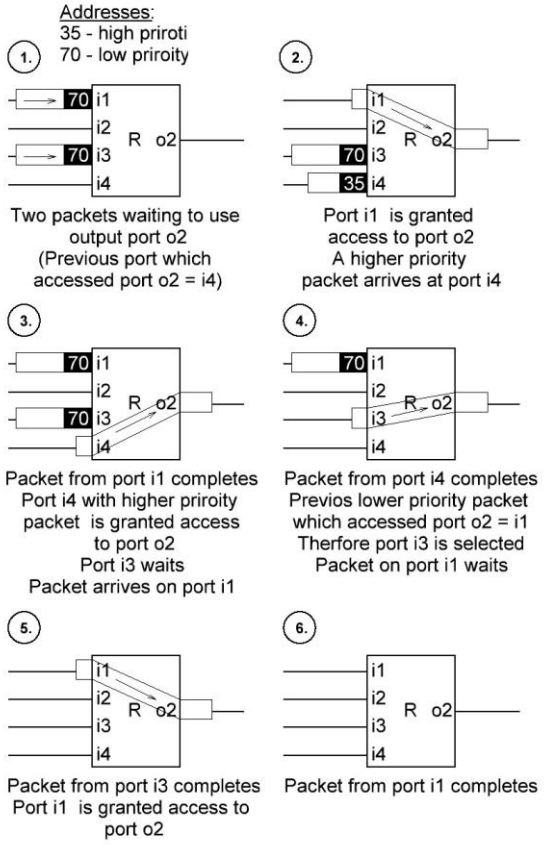

Fig.8.Arbitration of multiple packets with different priorities [9].

Multicast packet transmission is implemented in a way that the packet is forwarded through the selected output ports only when all of them can receive a new packet. Otherwise, the packet has to wait for this condition to be true.

Fig. 9 illustrates how the control logic handles multicast packet transmission. The example is divided into four frames where the multicast is enabled for the logical address 70. Indicated by a set Bit 0 in the multicast-set. Set bits on the other positions in the multicast-set denote the output ports through which the multicast packet should be forwarded. In the first frame, a multicast packet arrives at the input port i1. However, it is not immediately forwarded because one of the output ports specified by the multicast-set is busy. In the second frame, the input port i1 waits for the availability of all ports within the multicast-set. In the third frame, the packet that previously occupied the output port o2 has completed transmission. Now the multicast packet can be forwarded through the targeted output ports, o2 and o4. In the fourth frame, the multicast packet completes its transmission.

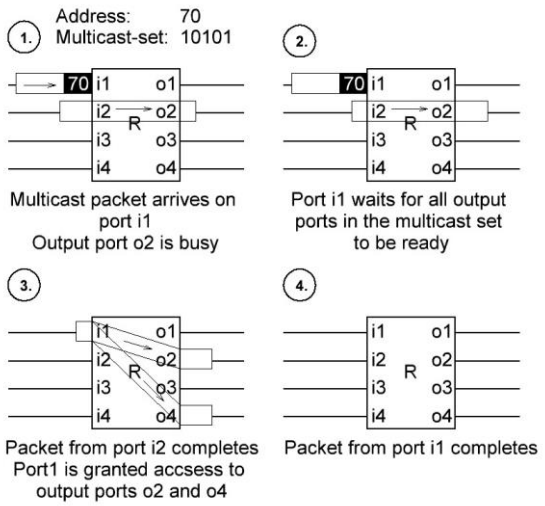

Fig.9.Multicast packet transmission [9].

Results

The SpW router IP-core was tested on a Microsemi RTG4 FPGA. It was configured as a four-port router with an internal clock frequency set to 50 MHz.

Besides an adequate control logic within the router, there are two additional important characteristic parameters of a router. A router delay and router propagation delay or packet router latency. The latter is, of course, dependent on the packet length. The router delay indicates how long does the router need for processing an incoming packet and determining its destination. Whereas, the packet router latency determines the time a packet needs to traverse through the router.

Test setup for measuring the packet router latency included NI PXI system with RMAP SpaceWire expansion card from STAR-Dundee and an RTG4 development kit with FMC SpW/SpFi card also from STAR-Dundee. The PXI system was running a LabVIEW program for two source nodes, logging, and a destination node. Both sources were configured to sand the same size packets to the same destination to demonstrate the influence of an output port contention alongside the packet router latency.

Preliminary tests were done on a Windows machine where the execution of tasks cannot be precisely controlled. It turned out that for traffic with a packet length of 2kB, 3kB, and 4kB, the packets were transmitted just right that when a packet from the first source completed transmission, a packet from the second source arrived at the router. Therefore, it did not experience any additional latency, except its own. In other cases, the measurements were as expected, as can be seen in Fig. 10. In continuous traffic, for this scenario, sources interchangeably wait on each other.

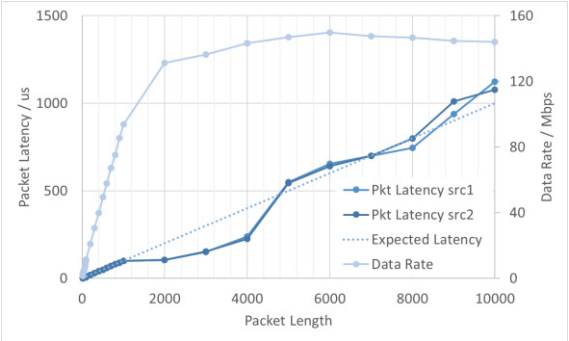

Fig.10.Packet router latency [9].

The router delay was determined by measuring the time difference between two trigger signals, which correspond to the events when the start of a packet was received at an input port and later at an output port. The router delay is different for each addressing mechanism. Fig. 11 depicts the measurement for packets using path addressing. The router delay is equal to 7 clock cycles, which results in a router delay of 140 ns at a 50 MHz clock.

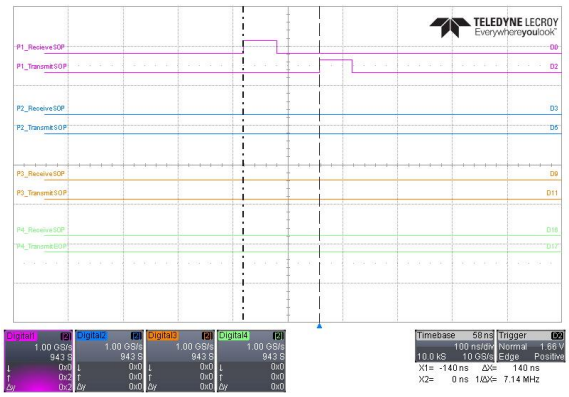

Fig.11.Router delay in case of path addressing [9].

Fig. 12 depicts the measurement for packets using logical addressing. In this case, the router delay is larger. As expected, since there are more processing steps in the packet processor, besides that, access to the routing table takes additional clock cycles. Altogether it takes 12 clock cycles to process a logically addressed packet, which results in a router delay of 240 ns at a 50 MHz clock. In case when multiple logically addressed packets arrive at the same time. The first packet is delayed for 12 clock cycles. However, subsequent packets from other input ports are only delayed for an additional 6 cycles, which is the time needed for the routing table access and an issued grant signal from the control logic. It is not the full 12 clock cycles because packets are already partly processed, and the input ports are just waiting for the routing table access.

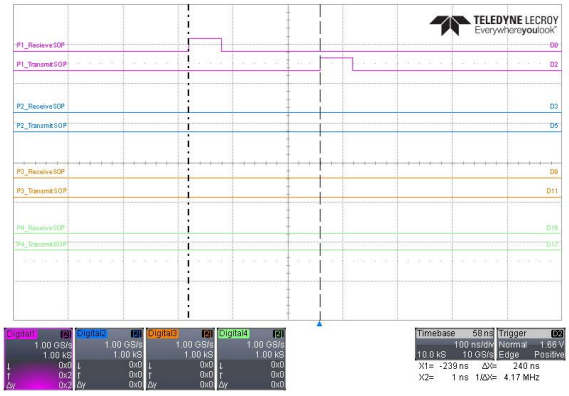

Fig.12.Router delay in case of logical addressing [9].

Fig. 13 and Fig. 14 give an overview of individual SpW router subcomponents resource usage compared to the resources used by the entire SpW router. The most significant component regarding the usage of Look Up Tables (LUTs) is the control logic (CL), followed by SpW ports (SpWP), and the configuration port (CP) (see Fig. 13). The size of the other components is negligible.

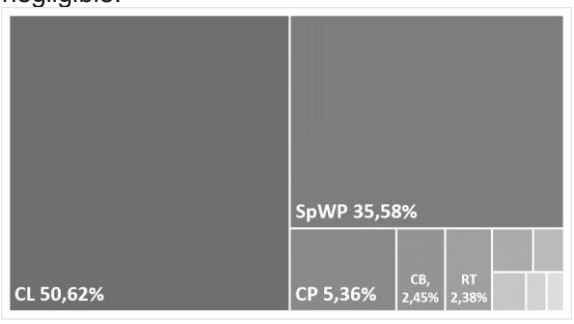

Fig.13.SpW router components LUT usage.

The most significant component regarding the usage of D Flip-Flops (DFFs) are the SpW ports, followed by the control logic, configuration port, routing table arbiter (RTA), and routing table (RT) (see Fig. 14). The RTG4 FPGA has 151824 LUTs and DFFs. Overall resource usage for a four-port SpW router was 13642 LUTs (8,99 %) and 5173 DFFs (3,41 %).

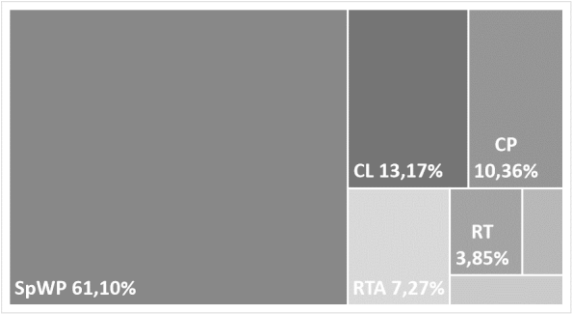

Fig.14.SpW router components DFF usage.

A couple of tests were performed to demonstrate the functionality of the control logic. The test setup was the same as for the packet router latency test. Though, for this test, all four SpW router ports were utilized. Fig. 15. depicts a snapshot of packet traffic where all four sources were transmitting packets with the same priority to the same output port. Packets follow the round-robin patter as it expected for output port contention among packets with the same priority.

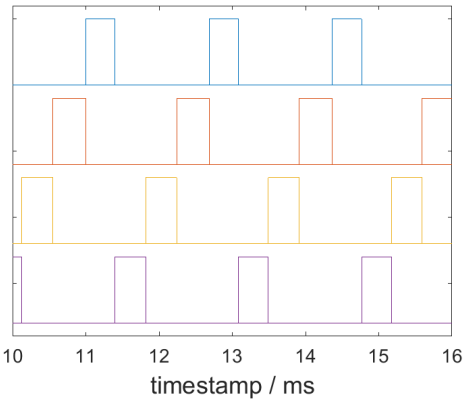

Fig.15.Snapshot of packet traffic, all packets have the same priority.

In another test, the last source was transmitting packets with a higher priority than all other sources, which were transmitting packets with the same lower priority. However, still through the same output port in order to demonstrate that a higher priority packet is granted access as soon as the previous transmitting packet completes its transmission. Snapshot of such traffic is depicted in Fig. 16. Black vertical lines indicate when a high priority packet arrived. The high priority packet is granted access to the output port as soon as the currently transmitting packet completes its transmission. Upper three sources in Fig. 16 still follow the round-robin pattern.

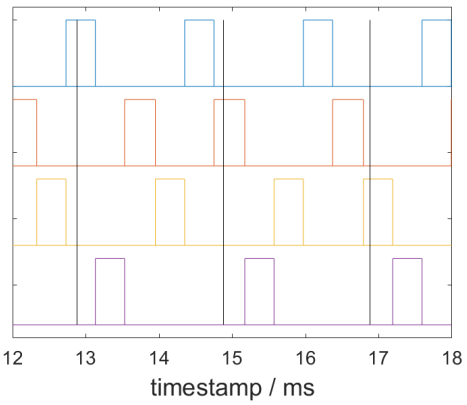

Fig.16.Snapshot of packet traffic, packets from the last source have a higher priority than the others.

Conclusion

Due to missing multicast and multilevel priority features in currently available devices, a design of an intelligent SpW router is investigated.

Essential aspects of the SpaceWire standard, which are relevant for the SpW router operation, are presented in the scope of this paper.

Description of the implemented SpW router IP-core is provided in the form of sperate components, determining their purpose and functionality. The focus is on the control logic as it provides the brains of the router and it is implicitly responsible for the flow of packets through the router.

Though, one crucial aspect of the SpW router, or rather the SpW CODEC, which can pose a serious implementation challenge, is not described. It is a SpW clock recovery circuit in the SpW CODEC [10].

Preliminary tests show that the SpW router meets all defined design goals. Two important characteristic parameters are defined, a router delay, and packet router latency. The control logic arbitration unit follows the implemented algorithm. Though, it is hard to showcase the multicast traffic with the current test setup.

New test setup and procedures are in work for evaluating the individual SpW router performance as well as performance on a system (network) level. Including deterministic sources, better network traffic logging, more extensive network, better visualization, and more.

References

[1] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su, “Myrinet: A Gigabit-per-Second Local Area Network”, IEEE Micro, vol. 15, pp. 29–36, (1995); doi: 10.1109/40.342015

[2] ECSS-E-ST-50-12C-Rev1 SpaceWire - Links, Nodes, Routers and Networks, ESA Requirements and Standards Division Std., Rev. 1.0, (May 2019)

[3] ECSS-E-ST-50-52C SpaceWire - Remote Memory Access Protocol, ESA Requirements and Standards Division Std., (Februray 2010)

[4] B. Chemli and A. Zitouni, “A Turn Model Based Router Design for 3D Network on Chip,” p. 8, (2014); doi: 10.5829/idosi.wasj.2014.32.08.1254

[5] M. Dridi, S. Rubini, M. Lallali, M. J. S. Florez, F. Singhoff, and J.-P. Diguet, “DAS: An Efficient NoC Router for Mixed-Criticality Real-Time Systems,” in 2017 IEEE International Conference on Computer Design (ICCD). Boston, MA: IEEE, (November 2017), pp. 229-232; doi: 10.1109/ICCD.2017.42

[6] J. Duato, S. Yalamanchili, and L. M. Ni, Interconnection Networks: An Engineering Approach. San Francisco, CA: Morgan Kaufmann, (2003); ISBN: 1-55860-852-4

[7] SpaceWireRouterIP 6PortVersion. shimafujigit. Accessed 28.8.2019. [Online]. Available: https://github.com/shimafujigit/ SpaceWireRouterIP 6PortVersion, (May 2019)

[8] B. Yuce, H. F. Ugurdag, S. Go¨ren, and G. Du¨ndar, “Fast and Efficient Circuit Topologies for Finding the Maximum of n k-Bit Numbers,” IEEE Transactions on Computers, vol. 63, no. 8, pp. 1868–1881, (August 2014); doi: 10.1109/TC.2014.2315634

[9] G. Skvarč Božič, "Intelligent SpaceWire Router Design", Master's thesis, (2019)

[10] Implementing SpaceWire Clock and Data Recovery in RTG4 FPGAs Application Note, 3rd ed., Microsemi, (March 2018)