Implementation of an AFDX Interface with Zynq SoC Board in FPGA

Fernando Molina[1], Pablo Corral[2, *], Miguel Aljaro[3], Guillermo de Scals[4], Alberto Rodriguez[2]
[1]*Department of Engineering, Creative Electronic Systems, Geneva, Switzerland*
[2]*Department of Communications Engineering, University Miguel Hernandez of Elche, Elche, Spain*
[3]*Department of Computers Engineering, University Miguel Hernandez of Elche, Elche, Spain*
[4]*Department of Applied Physics, University Miguel Hernandez of Elche, Elche, Spain*
pcorral@umh.es

***Abstract*—This work is based on the Hardware development of the Transmission part for the communication inside the satellite. Our goal is move as much as possible of the software part into the Field-programmable gate array (FPGA) matrix due to the single event upsets (SEU). This project is part of the collaborative project called "Mission: NET@SPACE". It was chosen by the European Commission under the Seventh Framework Program for Research (FP7) to develop an Avionics Full Duplex Switched Ethernet (AFDX) demonstrator based in FPGA. It has to be able to receive and transmit frames and enhance the robustness. The scheduling of the protocol should also be moved into the hardware, by still keeping a small footprint of the whole design. In this paper, we introduce the theory and used technologies, the project flow and development, including the decisions and milestones, to arrive at the end to the further possibilities and conclusions.**

***Index Terms*—Aerospace electronics; Aerospace simulation; AFDX; Latency measurements.**

I. INTRODUCTION

Avionics Full Duplex Switched Ethernet (AFDX) is a trademark of Airbus and a specific implementation of ARINC (Aeronautical Radio, Inc.) Specification 664 Part 7. AFDX is a deterministic network based on the same hardware as Ethernet (IEEE 802.3) [1]. Airbus developed this network to reduce weight on the planes and to provide a guaranteed bandwidth and Quality of Service (QoS). It is aimed for real-time, safety-critical applications which are generally mandatory in avionic systems. AFDX was developed around the year 2000 and was used on the planes A380 and A350. Later, a similar implementation was used on the Boeing 787 Dreamliner. The protocol is able to replace simultaneously multiple standard buses with low throughput, e.g., ARINC 429, ARINC 629 or MIL-STD 1553.

One of the predecessors of ARINC 664 Part 7, also known as AFDX, was the ARINC 429 formulated in the late 1970s which can still be found in some active and retired aircraft series [2]. This was one of the first standards ever made in avionics. This standard had as basic unit; the word, and two different coexisted in these networks: data words and message control words. However, ARINC 429 defined a unidirectional and simplex bus so that a station could be attached to multiple buses and operate as either sender or recipient. Because of this, a severe challenge was assumed when interlinking. Even when having few stations, the configuration could present important complications, and also this affected the weight of the aircraft.

The first commercial plane that uses AFDX was the A380, the biggest plane in the world, or the A350. It was first developed by the company "Airbus" and later provided on the ARINC 664 Part 7 as a standard. That is why Boeing, the direct competition of Airbus, has also started to use it such as in Boeing 787. AFDX networks were used to reduce a high percentage of the wires found inside of the aircraft. ARINC 429 uses wires to interconnect every sensor or end system (ES) with monitors or ES. AFDX uses just one wire shared by Virtual Links (VL).

II. STATE OF THE ART

AFDX added two new parameters related to aircraft networks. First of all, an AFDX switch, with similar characteristics to LAN switches, and the End System (ES) responsible to send and receive the data to/from the network.

– AFDX Switches: These devices are developed to connect the End Systems between them or with other switches together and to check that the key parameters like delivery, latency, and jitter are inside one specific range. This task creates a big overhead - compared to Ethernet switches - and slows the switching speed down. The main topology used with AFDX Switches and End systems is a star formation.

– AFDX End Systems: These systems are designed to work as a receiver, transmitter or both and they make use of the final network. Every End System is capable to

Manuscript received 16 May, 2020; accepted 18 September, 2020.

support multiple applications that can connect to the AFDX network.

Even if it is not a device, it should be pointed out that AFDX HW-Links consist of normal full-duplex Ethernet cables and connect an End System to a Switch. Every link is set up by using two cables for redundancy. Due to this redundancy, two identical networks are created. These networks are called "A" and "B", the same data are sent through both of them to minimize the error rates when transmitting. A concept called "Virtual Link" (VL), that has similar characteristics as WLAN, permits the origin End Systems sending data to the desired destinations. Without these links, we could not select the devices that we want to send, and we would send data to all devices without discrimination. These links send only in one way (no way back), so they are unidirectional.

This kind of links sends information about pressure, humidity, temperature from the engines, state of the wings, information about control panels, the movies and the audio of each passenger seat at the same time using the same network, but this information varies in each aircraft [3]. This information is processed and redirected using the AFDX switches by means of Fast Ethernet cables (100 Mbps) to its corresponding End System. Within the AFDX networks the type of traffic is differentiated according to the priority, which can be high or low. In this way, it is possible for the network to prioritize certain data (pressure, temperature or humidity of the engines) over other data (movies and audios for passengers). Anyway, two redundant networks always exist, just in case one brakes down.

Currently, the use of AFDX networks has become widespread and is considered a success. In the medium term, the use of this type of network with AFDX Switches and AFDX End Systems is proposed in the automotive industry for the next generation of connected vehicles.

A. *Bandwidth Allocation Gap*

BAG stands for "Bandwidth Allocation Gap" and is measured in ms. It defines the period of a VL and sets the maximum amount of packages which can be sent in one sending period (128 ms). The BAG number has to be by design an element of powers of 2 until 128. Setting BAG equal to 2 permits the VL to send 64 messages in one period. All these messages are equally distributed and have to be sent at the right moment within a maximum jitter of 500 us.

B. *Jitter*

The AFDX standard also introduces a jitter for every End System. The jitter is a deviation of the theoretical sending time. Depending on the amount of transmitting VLs and their bags, an End system will have a certain amount of jitter.

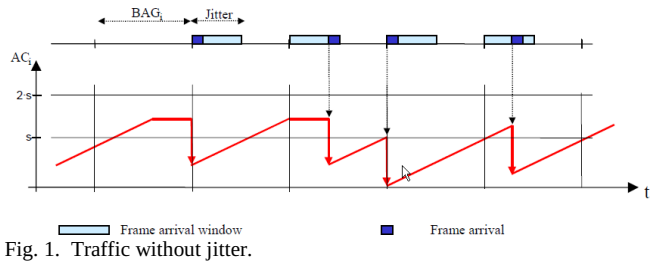

Fig. 1. Traffic without jitter.

This largest jitter of 500 us is fundamental to the demonstration of determinism for AFDX. With this limit, it is possible to check if the scheduling is done right. In the Fig. 1 and Fig. 2, we can see the differences of a traffic simulation with and without jitter.

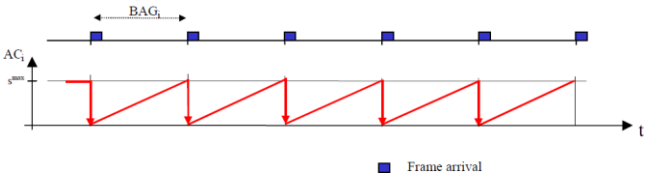

Fig. 2. Traffic with jitter = BAG/2.

C. *AFDX Frame*

A data flow is uniquely identified within AFDX network by the set of UDP/TCP (User Datagram Protocol/ Transmission Control Protocol) destination port, IP (Internet Protocol) destination address, MAC (medium access control) destination address, and the physical Ethernet connection(s) of the receiving ES. In order to use the standard Ethernet frame for AFDX, two bits of the 48-bit long MAC destination address have to be set to one. For routing all these addresses, include a Virtual Link Identifier (16 bits).

– Bit 40: Indicates the group address (always = 1).

– Bit 41: Indicates the locally administered address (always = 1).

– Virtual Link Identifier: Value to identify the virtual links defined by network-designed.

The MAC source address must be a unique identifier. Also, it includes the information of which interface the package was sent from.

– Interface ID (3 bits): Indicates the interface the frame was sent from: "001" network A, "010" network B.

– Bit 40: Indicates the unique address (always = 0).

– Bit 41: Indicates the locally administered address (always = 1).

– User Defined ID (16 bits): Identifies all hosts in a network.

Every frame at the end of the payload has a sequence number - just in front of the MAC CRC (cyclic redundancy check) field. This number is bound to a VL and incremented with each transmitted frame on the AFDX network. Initially - or after a reset - this number has to be zero to communicate a fresh start to the receiving ES. The sequence number makes it possible to detect missing frames. It is an 8-bit number, and after being incremented to 255, it should be wrapped around to 1 as 0 is used for communicating a restart [4].

III. WORK ENVIRONMENTS

In this section, we are going to describe the main hardware used in the topology for the AFDX test as we can see in Fig. 3. The Zynq-7000 is a family of system-on-chip (SoC) devices developed by Xilinx® that combines programmable logic (PL) with a hard-coded processing system (PS) [5]. Specifically, the programmable logic is a 28 nm FPGA, and the processing system is a dual-core ARM Cortex-A9 based processor. We had used two different developing boards: zc7002, mounting xc7020 SoC with

85 K Logic Cells, 560 KB Block RAM, 220 DSP slices, and zc7006, mounting xc7045 SoC with 350 K Logic Cells, 2180 KB Block RAM, 900 DSP slices [6].

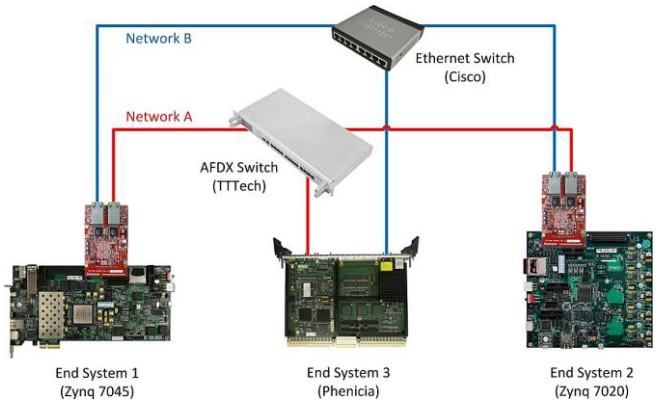

Fig. 3. Topology for the AFDX test.

These are designed with useful features, such as on board DDR3 socket, a variety of connectors like HDMI, RJ45, USB, and PCI-e; two FPGA Mezzanine Card (FMC) ports, an SDIO interface, etc.

The Phenicia board is an End System developed by CES. In our case, it is used only in order to observe the reception frames circulating on the AFDX network. It could also be configured in transmission, but it would need to implement to "hand" the content of the frames that you want to send (e.g., definition bit-by-bit addresses, checksums, data, etc.).

TTEthernet is a fault-tolerant security-critical real-time communication protocol that integrates time-activated, speed-constrained standard Ethernet data streams into a physical infrastructure. TTEthernet switches provide the skills for robust partitioning between these three classes of traffic, enabling mixed criticality systems [7].

IV. Tests and Results

The testing of our design was done first by test benches for each developed core. Then, to validate the whole system, tests have been created in software. There have been multiple tests for checking distinct parts as we can see in Table I. They can be separately enabled in the main user task. All the hardware tests do not depend on the XML file and load specific data into the hardware. The other tests use the XML configuration for testing.

TABLE I. TESTS' TABLE.

Test type	Function
Hardware test queuing	Test the transmission of queuing frames.
Hardware test sampling	Test the transmission of sampling frames.
Hardware test queuing and sampling	Test the transmission of both frame types.
Hardware test window table	Test the functionality of the window table.
User test attaching	Tries to attach sampling and queuing ports depending on the XML cong.
User test update sampling	Test the mutability of sampling frames.
User test Queuing ports	Sends repeatedly file like data through queuing ports.
User demo application	An example of how to use the API for sending data through the AFDX network.

A. TX Frame Format Validation

The demonstrator is based on AFDX message exchange between simulated end-systems, in our case, the development of the FPGA systems in the Zynq boards. The first step of validation is to check that the frames sent by the transmission part of the end-systems are compliant with the required AFDX standard.

The main frame format can be analysed on a PC connected to the AFDX network through its Ethernet interface. The Wireshark tool is used to expand the frame content and analyse in detail the conformance to IP standard protocol unless CAD-X can display the frame contents and show the parts of protocols that stay common between Ethernet and AFDX. This tool is not performed enough to catch all frames. Its timing resolution is not precise enough to control the respect of BAG policy.

Wireshark tool is used to check the validity of the calculation of the checksums inserted in the IP header and at the end of the frame (FCS).

It is to be noticed that UDP checksum is not used in transmission mechanism as it is not mandatory.

Figure 4 presents an example of Wireshark tool display for the Queuing frames for each end-system.

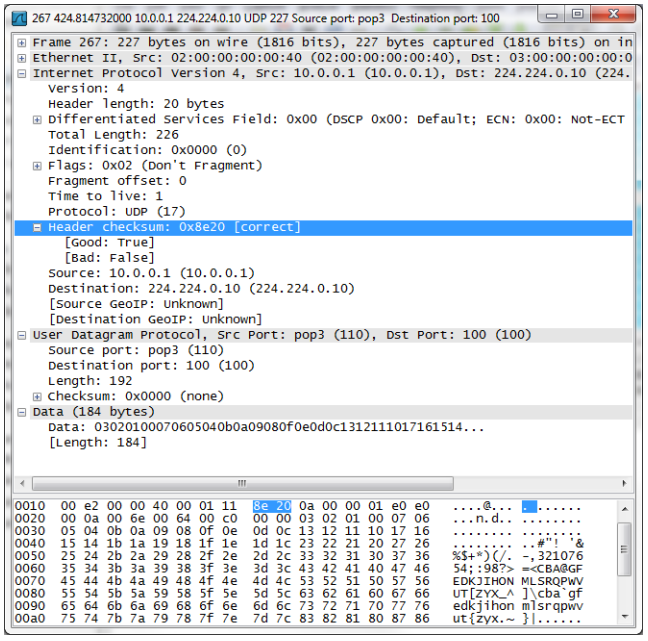

Fig. 4. Check with Wireshark of a Zynq™ 7045 frame associated to Queuing Port.

In order to analyse more in detail and to check the specific parts for AFDX, CES CAD-X is also used. Figure 5 and Figure 6 show examples of CAD-X display of captured frames on both Networks A and B for both end-systems Zynq™ 7045 and Zynq™ 7020.

In each captured frame, IP header checksum has been highlighted and is compliant with the one shown in the Wireshark tool. It can be noticed that there is a difference of 4 bytes in the frame size captured from the Wireshark tool and the CES CAD_X tool. This difference is due to the frame checksum which is included in the frame count by the CAD-X tool.

Figure 7 shows a console terminal for the two development boards during normal execution. It can be

noticed that printout does not discriminate the sampling received frames from queuing ones. This should be part of a further enhancement.

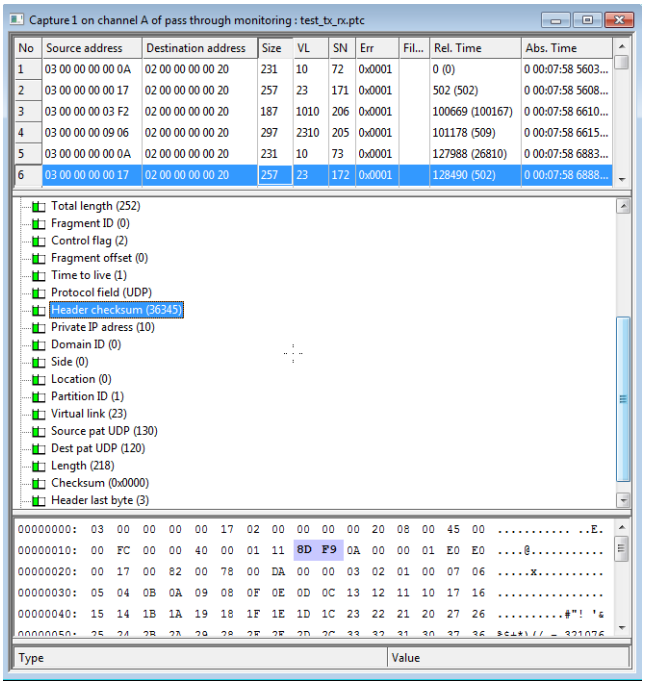

Fig. 5. Example of CAD-X display for Zynq™ 7045 Sampling port Network A.

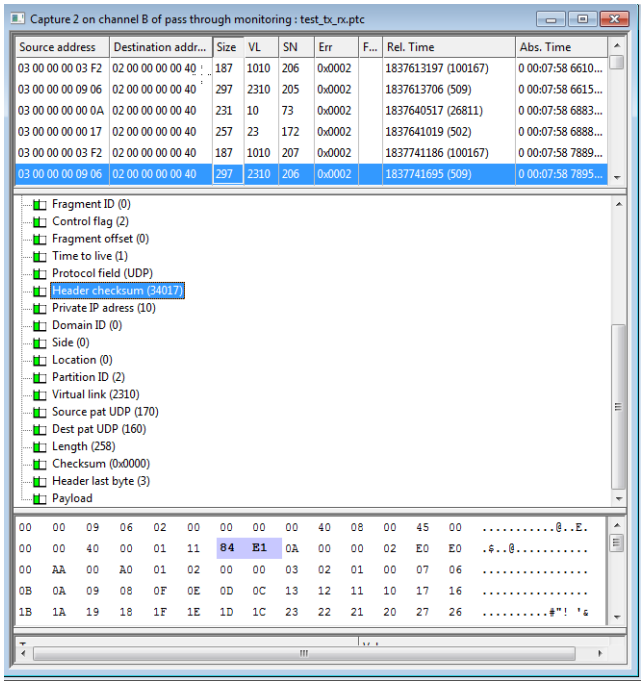

Fig. 6. Example of CAD-X display for Zynq™ 7020 Queuing port Network B.

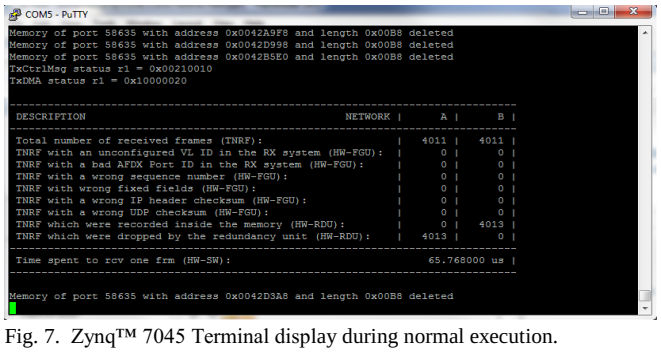

Fig. 7. Zynq™ 7045 Terminal display during normal execution.

This capture also shows that no erroneous frame is detected by the receiver, except the first one received by the Zynq™ 7020. This is due to the asynchronous application starting of both end-systems. As a consequence, the first frame received by the Zynq™ 7020 does not have a sequence number equal to "0" since the Zynq™ 7045 began to increment this value earlier in the frames it has already sent. The first frame is rejected by the Zynq™7020 allowing it to resynchronize the sequence numbers for the next received frames associated with the VL.

B. Latency Measurement

The latency test was made through the CAD-X tool by sending multiple frames in one window timeframe, and then checking how long it takes for the hardware to send a frame. The CAD-X displays the time between the receptions of two following frames also testing different lengths of frames. The probed values never exceeded 20 µs, for a maximum allowed latency is 150 µs [8].

C. FPGA Footprint Measurements

The transmission mechanism implemented within the hardware does not change fundamentally the footprint.

It is important to note that in these results the implementation uses a COTS module for MAC layer. Its use is not optimized and covers wider usage domain as the one really needed for AFDX for Space. Figure 8 shows the impact of the number of VL on the footprint. Note that the two curves on the left (light blue and orange) side show the fast increasing of the footprint with the amount of the queuing VL since each VL needs a FIFO.

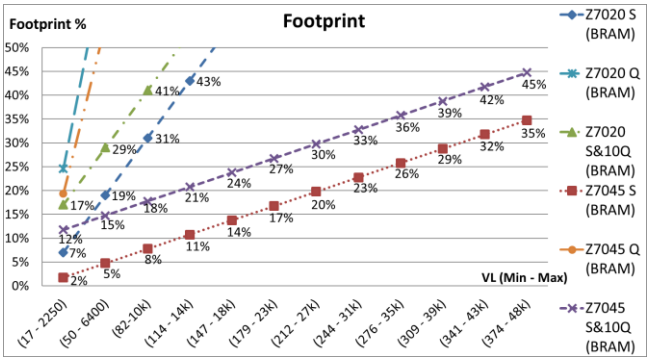

Fig. 8. Impact of the VL number on the footprint.

V. CONCLUSIONS

After all the work, we can arrive to the conclusion that the demonstrator of the AFDX End System implementation over FPGA is possible. On the one hand, the design of the system gives a dynamic possibility depending on the deterministic network result. On the other hand, with respect to the footprint measurements, the rise depends on sampling and queuing VL increasing number, being sampling method the most efficient in FPGA due to the shorter FIFOs needed. The benefit of FPGA as communication based in satellites brings a step further to the satellite technologies. The maintenance and/or upgrades at distance is an option that can reduce increasing the obsolete number of satellites in the space. Moreover, it is a simple solution to add robustness against SEU when implementing directly in hardware a typically communication layer integrated mostly in software.

Both benefits together allow the option to add higher number of redundant channels after the device is already on the space and allows the option to be adapted to the environmental exposition reducing the risk of satellite designs.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] N. E.-D. Safwat, A. Zekry, and M. Abouelatta "Avionics full-duplex switched Ethernet (AFDX): Modeling and simulation", in *Proc. of 2015 32nd National Radio Science Conference (NRSC)*, 2015. DOI: 10.1109/nrsc.2015.7117841.

[2] *AFDX®/ARINC 664 Tutorial*, TechSAT GmbH, Poing (Germany), 29 Aug., 2008.

[3] R. Gomez, P. Corral, S. M. Froes, A. C. D. Lima, I. De Barros, and G. De Scals, "An emulation model for embedded networks used in avionics", in *Proc. of 2019 Brazilian Symposium on Computing Systems Eng. (SBESC)*, 2019. DOI: 10.5753/sbesc_estendido.2019.8635.

[4] J. Yao, W. Shaojun, M. Ning, and P. Yu, "A SEU test and simulation method for Zynq BRAM and flip-flops", in *Proc. of 2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, 2018. DOI: 10.1109/ICEMI.2017.8265693.

[5] *Zynq-7000 All Programmable SoC*, Technical Reference Manual (v1.4), San José, US, 2012.

[6] *Xilinx Manual*, LogiCORE IP AXI Ethernet (v3.00a), San José, US, 2011.

[7] *Tttech manual*, TTTech. [Online]. Available: https://www.tttech.com/fr/products/aerospace/development-test-vv/development-switches/tte-development-switch-1-gbits-12-ports/

[8] Q. Guo, R. Feng, Y. Wu, and N. Yu, "Measurement of the AFDX switch latency based on FPGA", in *Proc. of 2016 IEEE International Conference on Aircraft Utility Systems (AUS)*, 2016. DOI: 10.1109/AUS.2016.7748018.