Implementation of the hardwired AFDX NIC

Pusik Park, Hangyun Jung
KETI
#68 Yatap, Bundang, Seongnam, Gyeonggi, Korea
+82-31-789-{7318, 7319}
{parksik, junghg}@keti.kr

Daekyo Shin, Kitaeg Lim
KETI
#68 Yatap, Bundang, Seongnam, Gyeonggi, Korea
+82-31-789-{7316, 7312}
{dukeshin, limkt}@keti.kr

Jongho Yoon
Korea Aerospace University
100 Hanggongdae gil,Hwajun, Deogyang, Goyang, Gyeonggi, Korea
+82-2-789-7312
yoonch@kau.ac.kr

ABSTRACT
The ARINC 664-7 called as Avionics Full Duplex Switched Ethernet (AFDX) deployed in modern aircrafts offers reliability and higher bandwidth for aircraft data network (ADN). The AFDX network system adopted the IEEE 802.3 Ethernet technology and added some special features to compose deterministic and fault-tolerant network. Transmitting characteristics of two kinds of the AFDX implementation such as the software-based MAC and the hardware-supported MAC were evaluated. The hardware-supported AFDX NIC has better performance than the software-based one but doesn't meet our own transmitter's requirement, which is arrival within 5% duration of each BAG, as well. Finally, the new hardware proposal and further works are suggested to solve the transmitting problem.

Categories and Subject Descriptors
B.4.1 [**Data Communications Devices**]: Hardware implementation of the avionics network interface controller is more efficient than legacy software-based implementation.

General Terms
Performance, Design, Reliability, Experimentation, Verification.

Keywords
Avionics, AFDX, Ethernet, Redundancy, COTS, NetFPGA, NIC, Controller, AND, QoS, Virtual link, BAG, IEEE 802.3.

1. INTRODUCTION

As several digital computing devices had been deployed in aircraft, needs of inexpensive and reliable networking technology that interconnect these devices had been raised and then commercial well-verified digital data bus technology was adopted.

Because high reliability is required when data is transmitted in an aircraft, various fault tolerance technologies should be applied to minimize or remove loss of the data.

In the late 1990s, as Internet technologies like Ethernet, IP, TCP, and UDP had lead data communication field, development of next-generation Aircraft Data Network (ADN) with low cost and higher data rate, was started. As a result, the ARINC664 specification was defined [1], [2], [3], [4].

The ARINC664 specification adopted Full Duplex Switched Ethernet technology and supported up to 100Mb/s for data transmission [5], [6].

Part 7 of the ARINC664 specification called as AFDX is based on Ethernet switching technology with additional fault tolerance functionality using two independent physical links and scheduling mechanism to supply bandwidth-guaranteed service.

In this paper, first of all, we introduce the AFDX specification briefly and then, we propose faster hardwired AFDX network interface controller including block-diagram of the hardware and the software. In the section three, results of hardware and software implementation will be shown and finally the performance evaluation will be compared with the legacy fully software-based AFDX system and provide some concluding remarks.

2. OVERVIEW OF THE AFDX

The goal of next-generation ADN technology was taking advantage of commercial off-the-shelf (COTS) technology to minimize development time and cost, while ensuring compatibility with the need for reliable data transmission and higher data rate. As a result, ARINC664 was defined as the profiled IEEE 802.3 network using TCP/IP protocol suite including fault tolerant redundant channels.

The ARINC664 specification consists of several parts:

Part 1: System concepts and overview

Part 2: Ethernet physical and data link layer specifications

Part 3: Internet-based protocols and services

Part 4: Internet-based address structures and assigned numbers

Part 5: Network interconnection services and interconnection

Part 7: Avionics full duplex switched Ethernet network

Part 8: Upper-layer and user services

The AFDX technology composes the deterministic network that supports guaranteed bandwidth and Quality of Server (QoS) based on the IEEE 802.3 Ethernet technology.

The major aspects of AFDX are as follows:

Full duplex switched network: the network is wired with a star topology and the physical interconnect medium is twisted pair, with separate pairs for transmit and receive channels. Each

switch can connect up to 24 End System (ES)s. And the network operates at either 10 Mbps or 100 Mbps link speed.

Redundancy: dual networks provide a higher degree of reliability than a single network scheme provides. Each ES copies a data frame and transfers original data frame and copied data frame via two independent switched networks.

Virtual Link (VL): the network separates a physical link into several VLs and multiplexes transferred data frames. Each VL is identified with Virtual Link Identification (VLID).

Deterministic: the network guarantees configured bandwidth and maximum jitter per each VL.

Profiled network: parameters for several ES's are defined in configuration tables at switches and ES's. Each ES and switch loads these configuration tables at startup and reset time to operate appropriately.

In the AFDX network, 24 ES's can be connected on a switch with a star topology. The AFDX network also offers link redundancy functionality using two independent physical links, guaranteed bandwidth, and QoS functionality using traffic shaping and policing.

The AFDX network has two independent data paths between two end systems. One end system generates two same frames and these frames should be delivered in the same time or within a tiny delay through each data path. These same frames are carried simultaneously via two separated links like the figure 1.

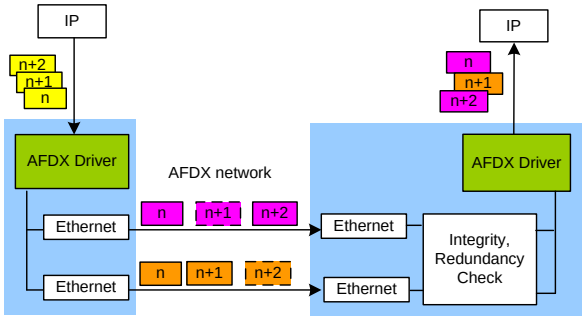

Figure 1. Redundancy support.

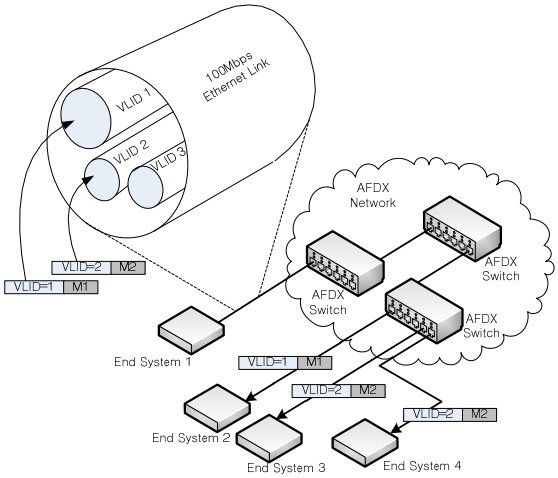

Figure 2. Virtual link.

Each ES which uses 100 Mb/s physical links supports multiple virtual links like the figure 2. It is possible that several devices share a physical link by identifying multiple logical links with 2 octet identification which can be configurable up to 216 virtual links in a physical link. The virtual link (VL) is a unidirectional logical link and one source node can have several destination nodes. Unlike legacy Ethernet switch, the AFDX switch multicasts received frames to multiple destination nodes that have same VL identification (VLID).

To allocate bandwidth of each VL, following parameters are used:

Bandwidth Allocation Gap (BAG): The BAG is the interval between two adjacent transmitted AFDX frames like the figure 3. The BAG value should be in range 1 ms to 128 ms.

These values should satisfy the following formula:

BAG=2^k [in ms](k integer in range 0 to 7). (1)

L_{MAX}: L_{MAX} is maximum byte number that can be transmitted. The maximum allowed bandwidth for a given VL is defined in equation (2).

$$BW_{VL} = \frac{L_{MAX} \times 8}{BAG} \quad (2)$$

where:

BW_{VL} is the maximum allowed bandwidth for the VL in b/s.

L_{MAX} is the maximum allowed frame size for the VL in bytes.

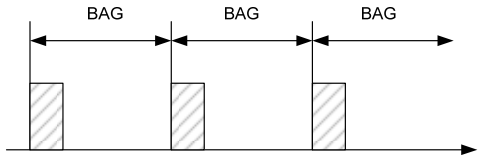

Figure 3. Bandwidth Allocation Gap (BAG).

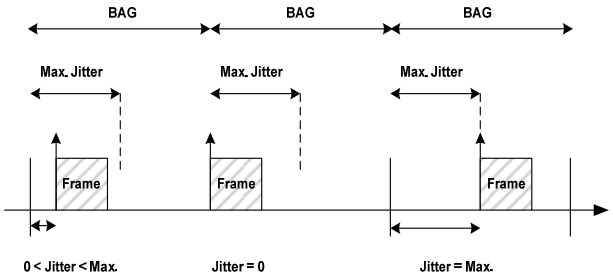

Figure 4. Jitter definition.

The jitter is variation of interval between two continuously received frames. The jitter can occur due to network environment. As shown in the figure 4, the transferred frame experiences delay that is the interval between BAG start time and transmitting time of first bit of the frame.

The maximum allowed jitter for a given ES is defined as follows.

$$\text{Max. Jitter} \leq 40\mu s + \frac{\sum_{i\in\{\text{set of VLs}\}} (20\text{byte} + L_{MAX}) \times 8\text{bits/bytes}}{N_{BW}} \quad (3)$$

where:

Maximum jitter is in μs.

N_{BW} is the medium bandwidth in b/s.

L_{MAX} is the maximum allowed frame size for the VL in bytes.

3. IMPLEMENTATION

3.1 Environment

Because the NetFPGA solution is based on the Linux, which is exactly not the real-time operating system, precise verification and performance evaluation for the AFDX transceiver function didn't fully perform. However fundamental implementation and evaluation are possible on the NetFPGA environment.

The NetFPGA provides four 1GbE ports and PCI interconnection and thus is able to support very efficient environment for the AFDX NIC development, because basically two or more Ethernet channels are required for redundant communication and current commercial AFDX NIC solution has PCI interconnection with the PMC form-factor. Additionally, the redundancy can easily extend by using spare two Ethernet channels and link speed can improve as well.

3.2 Software-based implementation

For comparison against the AFDX hardware implementation, fully software-based AFDX function was implemented by using software timers and two general Ethernet transceivers.

The figure 5 and figure 6 show the software block diagram of the fully software-based AFDX receiver and transmitter. Whole functions such as BAG timer, integrity checkers, sequence number checkers, VLID filtering, and redundant management was implemented in the software stack.

3.3 Hardware-based implementation

The several functions were implemented in the hardware. The most important functions such as precise BAG timers, integrity checkers, and redundant management were accelerated in the FPGA with direct connecting two Ethernet transceivers.

The Xilinx ISE tools configuration is as follows:

Target device	: xc2vp50-7ff1152
Product version	: ISE 10.1.03
Design goal	: Balanced
Design strategy	: Xilinx Default

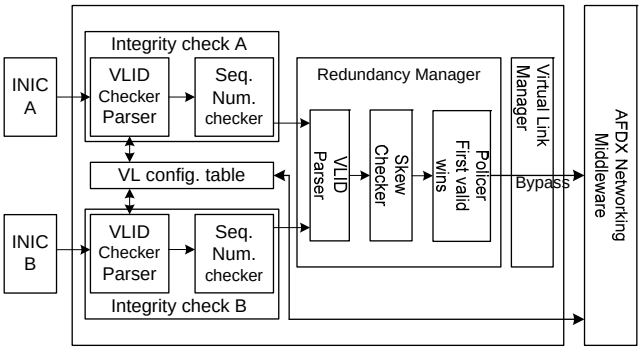

Figure 5. AFDX Rx software block diagram.

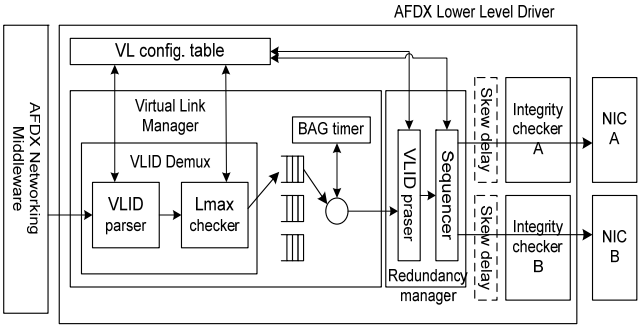

Figure 6. AFDX Tx software block diagram.

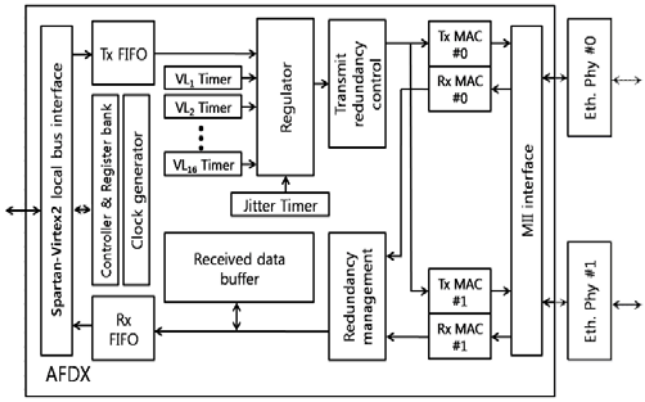

Figure 7. AFDX NIC hardware block diagram.

The logic utilization summary is as follows:

Number of Slice Flip Flops	: 14,960 of 47,232 (31%)
Number of 4 input LUTs	: 18,957 of 47,232 (40%)
Number of RAMB16s	: 106 of 232 (45%)
Number of BUFGMUXs	: 8 of 16 (50%)
Number of DCMs	: 6 of 8 (75%)

4. RESULTS

4.1 Traffic shaper's jitter

If there is no switched network and two end systems connects directly, accuracy of the BAG timer and processing delay during preparation of transmission are essential for traffic shaper's jitter evaluation. The Ethernet analyzer is used to capture frames and calculates the arrival interval between adjacent frames. The table 1 and table 2 show the jitter results according to various length of a frame and several BAG values for each implementation.

The table 1 and the figure 8 show the shaping jitter of the transmitter of the software implementation. Most of all results meet our transmitter's requirement such as a period within 5% of the BAG duration. However, there is an undesirable case when the BAG is 1ms. The maximum jitter exceeds the requirement and doesn't meet it. That means that software-based implementation is not sufficient to support the transmitter's requirement.

The table 2 and figure 9 show the shaping jitter of the transmitter of the hardware implementation. Even though there is a little performance improvement, this traffic shaper's jitter characteristics doesn't meet our transmitter's requirement. When the BAG is 1ms, the maximum jitter exceeds the requirement as well.

This problem was caused by the hardware specification. The hardware was implemented like first-BAG timer expiration and context delivery. The timing to transfer a frame into the hardware was not stable because of the non-realtime Linux's interrupt service routine.

Table 1. Result of traffic shaper's jitter (SW).

BAG [ms]	Lmax [bytes]	Max. jitter [%]	Avg. jitter [%]
1	64	9.22	0.84
	700	9.12	0.99
	1518	9.52	0.9
64	64	1.68	0.07
	700	1.63	0.08
	1518	1.64	0.08
128	64	0.88	0.04
	700	0.81	0.04
	1518	0.81	0.05

Table 2. Result of traffic shaper's jitter.

BAG [ms]	Lmax [bytes]	Max. jitter [%]	Avg. jitter [%]
1	64	8.88	0.73
	700	8.98	0.84
	1518	9.18	0.81
64	64	0.47	0.07
	700	0.24	0.07
	1518	0.48	0.07
128	64	0.83	0.03
	700	0.12	0.03
	1518	0.12	0.03

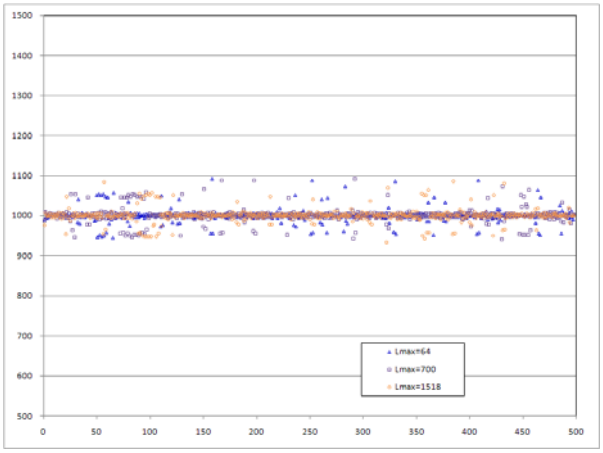

Figure 8. Graph of traffic shaper's jitter, BAG = 1.

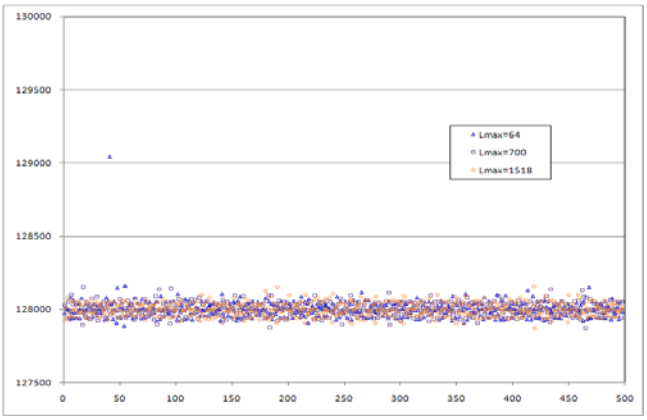

Figure 9. Graph of traffic shaper's jitter, BAG = 128.

4.2 Transmission performance

For evaluating transmission performance, the result, maximum allowable jitter according to the maximum frame length and the number of virtual links should be measured.

Table 3. Result of transmission performance.

Lmax [byte]	64	700	1518
Num. of VLs	16	7	3
Max. allowable jitter [us]	147.52	443.2	409.12

Max. jitter [us]	133	691	385
Avg. jitter [us]	113.77	349.67	359.37
Success rate of jitter performance [%]	100	99.8	100

Table 3 shows the result of transmission performance. Most of all results are delivered successfully and the figure 10 shows the transmitting jitters and most of jitters are smaller than the maximum allowable jitter represented by the red line. However, there are 0.2% packets which don't be arrived at the NIC within the maximum allowable jitter 500us.

This is why the interrupt service routine's timing is not precise as well and this inexact timing should be caused by our software architecture. The file isn't copied from the hardware buffer to the system memory in the kernel's interrupt service routine but in the application receiving of the kernel event. This software architecture causes these kinds of jitter evaluation due to context switching delay.

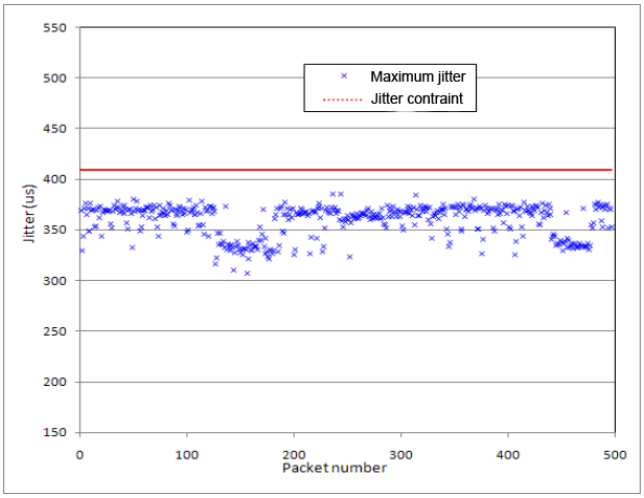

Figure 10. Graph of transmission performance.

5. CONCLUSION AND FURTHER WORKS

Two kinds of AFDX implementation were evaluated. The results of two models were not enough to verify the AFDX transmitter's requirement. There are two problems. One is the test environment. The test platform is the Linux and it's not the real-time operating system. The other is the software architecture that is the place to transfer a frame. Current system is waiting for BAG interrupts according to the virtual link IDs and the CPU should begin to copy a frame into the AFDX NIC after BAG timer expiration. Because these works didn't perform in the kernel routine but in the application, there were context switching delays every file transmission. Finally, these context switching delays caused transmission timing to become inaccurate and thus the jitter characteristics were not stable.

To fix these problems, the design of the NIC and the software architecture should be modified. Contexts delivery should perform as soon as new context is created. That means the CPU should copy a frame into the AFDX hardware memory on receiving a frame from the applications and then the hardware should transmit received frames already, according to each BAG expiration time. For better performance, more hardware logics should be necessary.

6. REFERENCES

[1] ARINC 664 Standard, 2002, Aircraft Data Network.

[2] ARINC 664 Part 1 Standard, 2002, Aircraft Data Network Part 1 : System Concepts and Overview.

[3] ARINC 664 Part 2 Standard, 2002, Aircraft Data Network Part 2 : Ethernet Physical and Data Link Layer Specification.

[4] ARINC 664 Part 7 Standard, 2005, Aircraft Data Network Part 7 : Avionics Full Duplex Switched Ethernet (AFDX) Network.

[5] IEEE 802.3 Standard, 2005, IEEE Standard for Local and Metropolitan Area Network, Part 3 : Carrier sense multiple access with Collision Detection (CSMA/CD) access method and physical layer specification.

[6] IEEE 802.1D, Local and Metropolitan Area Network : Media Access Control Level Bridging.

[7] NetFPGA project, http://netfpga.org

[8] Charara, H., Fraboul, C., 2005, Modelling and Simulation of an Avionics Full Duplex Switched Ethernet, In the Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop, (Lisbon, Portugal, July 17-22, 2005), AICT/SAPIR/ELETE'05, 207-212.

[9] Anand, M., Vestal, S., Dajani-Brown, S., Lee, I., 2006, Formal Modeling and Analysis of the AFDX Frame Management Design, In Proceedings of the Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (Gyeongju, Korea, April 24-26, 2006), ISORC'06, 393-399.

[10] Chen, X., Yin, H., Zhou, Y., Wan, J., 2008, An Effective Framework for Delay Control in Hard Real-Time Switched Networks, In the Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications, (Sep. 25-27, 2008), 736-743.

[11] Chen, X., Xiang, X., Wan, J., 2009, A Software Implementation of AFDX End System, In the Proceedings of the International Conference on New Trends in Information and Service Science, (Beijing, China, June 30-July 02, 2009), 558-563.