

An AFDX Switch Fabric Hardware Core for Avionic Network Prototyping and Characterization

Davide Trentin
Ecole Polytechnique de Montreal

Yvon Savaria and Guchuan Zhu
Ecole Polytechnique Montreal

Claude Thibeault
Ecole de Technologie Superieure

ABSTRACT

Avionic Full-Duplex Switched Ethernet (AFDX) is one of the most promising solutions developed in recent years for implementing high-bandwidth Avionic Data Networks (ADNs) that can support the increasingly high information flow required by modern avionic systems. Although AFDX commercial products are available, developing custom implementations using generic software and hardware, without depending on third-party products, allows identifying practical challenges and constraints, prototyping and characterizing new architectures, testing new algorithms, as well as performing validation and verification in the early stage of development. In this paper, we show how an AFDX switch fabric hardware core can be designed and implemented on an FPGA to facilitate the prototyping of a generic ADN in an early development stage. This implementation is compatible with a generic platform that provides connections with a PC offering multiple physical ports and supports the generation of custom traffics for system validation and characterization. More specifically, the developed switch fabric component is a soft core configurable at synthesis time to support from 2 to 24 ports, resulting in a core size that needs from 10k to 65k LUTs on a Spartan-6 FPGA. A 3-port switch fabric has been integrated on a XC6SLX45T FPGA, which will be used to validate the system behaviour in a real network. The Scheduler behaviour and the resulting average latency introduced do not depend on the switch size, but rather on the incoming traffic load. It is also demonstrated how a maximum sized switch fabric (24 ports) can guarantee that no input congestion occurs for traffic up to 25Mbit/s per port. In order to reduce high priority frames latency, low priority frames are stored in a separate buffer, thus making critical traffic processing mostly insensitive to their presence. The queues allocated to non-critical traffic can also be used to store critical frames whenever the main buffer fails due to hardware problems or when it is full, thus reducing the possibility of frame loss.

CITATION: Trentin, D., Savaria, Y., Zhu, G. and Thibeault, C., "An AFDX Switch Fabric Hardware Core for Avionic Network Prototyping and Characterization," *SAE Int. J. Aerosp.* 5(1):2012, doi:10.4271/2012-01-2123.

INTRODUCTION

The evolution of avionic systems, together with the adoption of Integrated Modular Avionic (IMA) architectures, brought a significant increase in the volume of digital data exchanged between devices across different application domains. In order to provide the bandwidth necessary to support all these communications, new technologies have been developed to replace or to coexist with traditional ones. One of the most promising solutions is the Avionic Full-Duplex Switched Ethernet (AFDX) that not only provides high bandwidth, but also allows significant reductions of the

complexity of wiring bundles. The AFDX technology also benefits from the exploitation of Commercial Off-The-Shelf (COTS) components originally developed for Ethernet, thus reducing development and maintenance costs.

As the AFDX has only been introduced recently in the avionic environment, it is still of great interest to analyze and characterize networks built on this technology, in order to identify its full potential and limitations. The objective of the present work aims at proposing modifications and improvements to the protocol as well as effective means of implementing system components compliant with the ARINC

664 Part 7 standard [1] to obtain better performance, especially in terms of reliability and end-to-end delay. The development of a prototype of an AFDX network is an important step required to validate and characterize the network behaviour, as well as the impact of any proposed modification.

Although there are existing commercial products that can be used to implement and configure a complete AFDX network, they usually do not allow important modifications on their structure. Consequently, these platforms are not suitable for prototyping new systems, where it is important to explore how to integrate new techniques and solutions, such as new scheduling algorithms or new methods to control the transmission jitter. This is necessary for validation and full characterization of the impact the new techniques and solutions have on performance and complexity. Extensive research is being done to model AFDX networks and to propose solutions to make them more deterministic while limiting the maximum end-to-end delay [2]. The present work aims at supporting the development of a hardware platform that can be used to prove that mathematical results obtained with models, simulation and analysis can be attained in some real networks as well. This is important when considering that models are usually limited in how accurately they reflect reality and un-modeled features may lead to miss some key specifications in the field.

While some custom implementations of AFDX End Systems have recently been presented [3], the issues related to determining how the ARINC 664 specification affects the design of switches required for implementing AFDX networks has not been addressed in the open literature. In this paper, we will report on the development of a custom hardware switch fabric and on its integration in a generic platform used for validating and characterizing the prototype. By developing this system component from scratch as a hardware core targeting an FPGA physical implementation, it is possible to have full control of its features, which offers the flexibility required in a prototyping stage and allows identifying the challenges related to its implementation. This work also allows an exploration of the internal structure of the switch, providing important information about the latency that can be expected on each processed frame and potential bottlenecks of the system.

AFDX SWITCH FABRIC SPECIFICATION

AFDX networks employ a switched topology comparable to that found in Ethernet networks. Switches in networks are needed to route frames towards their respective destinations. As in Ethernet switches, this functionality is provided by a component called switch fabric. The main roles of a switch fabric are to analyze the frames, to determine their destination, and to forward them towards the corresponding output ports. A detailed description on functionalities that the switch fabric must provide is given in Chapter 4 of ARINC

664 Part 7 specification [1]. An AFDX switch is composed of different modules that are respectively responsible for traffic policing, frame filtering, switching, handling of routing tables, as well as error monitoring and supervision. As this work focuses on the design of switch fabrics, only the characteristics relating to this component are described in this section.

The switching functionality of an AFDX switch is very similar to that found in any Ethernet switch. The header of each incoming frame must be analyzed to extract the destination MAC address, which is used to determine, based on the routing tables, towards which output ports the frame must be transmitted. The most common solutions used in Ethernet switch fabrics to bring the incoming packets to the desired destinations are space-division and time-division switching architectures [4]. In the former, frames received on different input ports are processed in parallel, and a scheduler solves possible output contentions and determines which frame can be forwarded to each destination at a given time. In time-division switches, on the other hand, all frames are sent to a single bus or memory and are sequentially processed. This requires less hardware resources but higher operating frequencies to achieve the same total throughput.

As multiple frames, coming from different input ports, may require to be transmitted towards the same output port, it is important that the switch fabric includes a mechanism to store the frames that lose output contention. One solution consists in providing each output port with N buffers, where N is the number of inputs, so that each packet can be immediately sent to the required outputs. This allows scheduling to be performed locally. This approach is generally called output buffering. A second solution allowing a better utilization of memory resources is input buffering, which requires that each input module stores received packets in its own buffer. The incoming frames then wait for the global Scheduler to forward them to the required output port.

One problem of input buffering is that the first frame in the input buffer, also called the head-of-line (HOL) packet, may block frames received afterwards whenever it loses contention multiple times. This problem is also known as HOL blocking and it causes a reduction of the maximum throughput of a switch fabric. This problem can be mitigated by adopting a combined input-output buffering technique or using virtual queues (VQs) in input buffers [5].

Apart from the previously described common features for Ethernet and AFDX switches, the ARINC 664 specification defines additional functionalities that need to be provided by an AFDX switch fabric. The most important ones are the need for frame filtering and for traffic policing. They become fundamental in critical, real-time networks, where it is crucial to identify and eliminate erroneous frames that may otherwise increase the latency of valid frames. Frame filtering ensures that any frame with erroneous CRC, structure, or size will be discarded, while traffic policing guarantees that the bandwidth allocated to each virtual link (VL) will be respected.

VLs are the most important feature introduced by AFDX: they are virtual communication channels coexisting on the same network. In order to limit interference between virtual links, a bandwidth allocation gap (BAG) is assigned to each of them. The BAG determines the minimum time interval separating two consecutive transmissions on the same VL. Therefore, it limits the bandwidth allocated to each virtual channel. To ensure that the received frames are conforming with the BAG allocated to their VL, the switch fabric must implement a token-bucket algorithm, as specified in [1].

Another important feature of AFDX switches is latency reduction. Although standard Ethernet switches are designed to provide a good average quality of service, it is still crucial in AFDX networks to further limit the maximum latency added to every incoming frame and to guarantee that an upper bound is never exceeded. Consequently, the system must be designed to ensure that frames will not wait indefinitely in the internal buffers for transmission and that, if this happens, excessively old frames, which the Scheduler was not able to forward due to excessive congestion, will be eliminated. Note that AFDX supports two priority levels, high and low, and that high priority frames must always be transmitted before low priority ones. Finally, the presence of a redundant network in AFDX does not influence the design of the switch fabric. When there are two networks, a common scenario, they are in fact completely separate and independent. Consequently, switches are not aware of the presence of the redundant network.

SWITCH FABRIC HARDWARE STRUCTURE

The system reported here has been designed in VHDL as a platform independent hardware core. Because of the similarity between Ethernet and AFDX switches, a space-division switching architecture, originally intended for an Ethernet switch fabric, was used as starting point. Packet switching has been preferred over cell switching because it does not require segmentation and reassembly (SAR), but uses simpler scheduling algorithms and allows a reduction of the necessary hardware resources [6]. Furthermore packet-mode scheduling can provide better throughput than cell switching [7]. A parallel structure has been preferred over a time-division mechanism, because the relatively low maximum clock frequencies that can be used on FPGAs would have otherwise excessively limited the number of ports that the system could support. As shown in Fig. 1, each input has its own processing module and buffer. The input processing module is called Filter to underline the main difference from a similar module present in Ethernet switch fabrics: while the input module of Ethernet switches only needs to save incoming frames in the input queue while extracting their header, in AFDX switch fabrics, this module is also responsible for frame filtering. By detecting and eliminating erroneous packets as soon as they are received, we can avoid wasting resources to process them, both in the

switch and in the rest of the network. To eliminate all the frames that are not compliant with the AFDX specification the Filter must ensure that:

- the CRC is valid;
- the frame structure complies to the AFDX standard;
- the frame size is within the bounds assigned to the relevant Virtual Link.

As all these tests are executed while an incoming packet is being saved in the input queue, the Filter contributes no additional delay. By including these filtering features in the input modules, concurrently received frames can be processed in parallel.

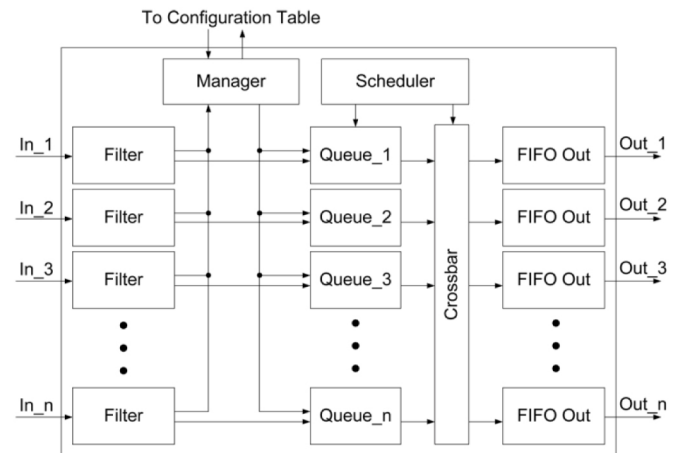


Figure 1. Switch fabric hardware architecture

The Manager is the only module that has access to the routing and configuration tables; therefore it is responsible for determining packets' destinations and providing relative information to the input storage buffers. Configuration tables can be accessed using the MAC destination address, included in the header that has been extracted by the Filter, since it determines the VL the frame is transmitted on. The Manager also performs a second filtering stage, called Traffic Policing, to ensure that frames that do not respect the allocated bandwidth are discarded. This was not included in the input Filters in order to keep the system complexity reasonable. Indeed, the algorithm to be implemented in this filtering stage is rather complex, and it would take too much resource on the FPGA if implemented as part of each input port. In addition, the Manager is the only module that has direct access to the configuration tables, from which information about BAG and jitter assigned to each VL can be retrieved. If the frame respects the allocated bandwidth, the Manager sends information about its destinations and priority to the queue where it is being saved. Information about the minimum and maximum acceptable frame size is provided by the Manager to the input Filter to complete the aforementioned controls. To ensure that the limit on transmission rate is respected, a token-bucket algorithm has been implemented in the