FPGA-based Implementation of an Ethernet Switch for Real-Time Applications*

Rui Santos, Ricardo Marau, Arnaldo Oliveira, Paulo Pedreiras, Luis Almeida
DETI/IEETA, Universidade de Aveiro, Portugal
{*rsantos, marau, arnaldo.oliveira, pbrp, lda*}*@ua.pt*

Abstract

The use of switched Ethernet for precise and safe real-time communication still suffers from undesired phenomena, such as blocking caused by long non-preemptive frames, lack of protection against errors in the time domain, couplings across virtual LANs and priority levels via internal switch shared resources. Recently, a few solutions were proposed to cope with such phenomena. One such solution is based on an enhanced switch following the Flexible Time-Triggered paradigm, which enforces strict service differentiation, blocking-free forwarding and timing errors confinement. In this paper we propose a new architecture following an hardware-software co-design approach that facilitates the development of the enhanced switch features by separating the traffic scheduling from the common management activities associated to switching.

1. Introduction

Nowadays, Ethernet and particularly using switches, presents attractive advantages to provide safe, predictable and deterministic communications. It has a large bandwidth, cheap network controllers, high availability, easy integration with Internet and positive future prospectives.

However, Ethernet switches are not typically designed to support the requirements of hard real-time networks. Certain undesired phenomena hinder this use in applications that require precise and safe real-time communications, such as the control of high speed servos, target tracking in military systems or even control of electrical protections in substations. Such phenomena range from blocking caused by long non-preemptive frames, lack of protection against errors in time domain, couplings across virtual LANs and even priority levels via internal switch shared resources.

Supporting real-time communication with switched Ethernet has been a topic of intense research for several years. Several techniques were proposed to overcome its limitations which can be divided in two groups, namely the solutions based on COTS Ethernet switches and the solutions that use modified switches. The first group includes many different techniques, but most of them requiring software modifications in the end nodes, for instance: the shaping of the traffic submitted to the switch [8] and limiting that traffic by application design [3] [7]; and master-slave protocols [10] [6] [2] that provide more efficient scheduling policies, QoS management and admission control features. The second group includes solutions [13] [12] [15] that allow to obtain gains in terms of performance, timeliness guarantees, combined with low level of intrusion because it is possible to use network nodes with standard hardware and software stacks, possibly with specific layers just for accessing the real-time services.

This paper presents a solution belonging to the second group. Particularly, the paper describes the FPGA-based architecture of a modified switch that provides real-time communication services, based on the Flexible Time-Triggered paradigm. The current architecture is an evolution of a preliminary prototype [9] [13] that facilitates the development of the enhanced switch features by separating the traffic scheduling from the common management activities associated to switching.

2. Related work and Contribution

The fast evolution of programmable logic devices, in particular FPGAs, allows building customizable devices with specific properties for different application domains. They present a set of interesting advantages, such as, very large logic capacity, flexibility of use and low NRE (Non-recurring engineering) costs. Moreover, they exhibit a fast design cycle, easily upgradable and fast prototyping due to high level modeling languages and synthesis tools. The integration of standard IP cores with custom functional blocks further increases those advantages.

On the other hand, the development of protocols and ethernet devices that supporting real-time communication can take advantage of FPGA technology. Industrial and embedded device vendors are increasingly interested in providing ethernet devices that support real-time services. According to [1], there is a demand for the integration of Modbus/TCP, Profinet IO and Ethernet/IP into a single device. The FPGAs are probably the best solution, given the possibility to integrate soft-core processors and IP cores.

In the case of the real-time protocols, one possible solution is to enhance switches with tighter timing control and traffic classification capabilities. There are currently two protocols that use modified switches, TTEthernet and Profinet IRT. TTEthernet (Time Triggered Ethernet) [15] [14] [11] is a new scalable real-time Ethernet platform that was created by the Real-Time System Group in Vienna University of Technology. It provides safe and real-time

*Work partially supported by the FCT project PTDC/EEA-ACR/73307/2006 (HaRTES - Hard Real-Time Ethernet Switching)

Ethernet communication among applications. The communication can be performed using three different traffic classes, Time-Triggered (TT), Rate-Constrained (RC) and Best-Effort (BE) messages. The messages of the first group are in the top priority, not suffering interference of other types of traffic and they are transmitted in predefined instants. The RC messages are used for applications with less stringent determinism and real-time requirements than time-triggered applications. They guarantee that bandwidth is predefined for each application and the delays as well as temporal deviations have defined limits. The BE messages have no time guarantees because their transmission is performed in the remaining bandwidth after the transmission of the TT and RC messages. This protocol uses a modified switch that includes a dedicated TTEthernet controller and a high performance switching module. According to [15] this switch is based on an Altera Stratix II GX FPGA board.

The other protocol is Profinet IRT [12] [5] [4]. It was developed by the Profibus International Consortium and is also based on switched Ethernet technology. This protocol is implemented with a modified switch that offers determinism in the transmission through explicit bandwidth reservation for the real-time data. The scheduling parameters are configured during the setup phase and they are obtained through the previous execution of a scheduling algorithm. On the other hand, this switch also allows the integration of Ethernet standard devices whose traffic is confined to dedicated time windows.

Despite the advantages presented by the above two protocols, they exhibit some limitations, e.g., they require a static pre-defined configuration of the time-triggered traffic and of the slot structure of an underlying TDMA round. Therefore, we propose an alternative based on the Flexible Time-Triggered (FTT) paradigm that is fully reconfigurable online. We propose an FTT-enabled switch providing support for arbitrary traffic scheduling policies, as well as online admission control, policing mechanism and dynamic quality of service management.

3. FTT-Enabled Switch

3.1. Rationale

The FTT-enabled switch is based on the Flexible Time-Triggered paradigm, being the FTT master included in switch (Figure 1). Therefore, it uses the master/multi-slave transmission control technique, according to which a master addresses several slaves with a single poll, considerably alleviating the protocol overhead with regard to the conventional master-slave techniques. The master uses transmission control to provide improved timeliness and enforce system integrity, for example, it completely avoids memory overflows inside the switch. This is achieved organizing the communication in fixed duration slots called Elementary Cycles (ECs), with one master message broadcast per cycle called Trigger Message (TM), which contains the periodic schedule for that EC. The periodic messages are

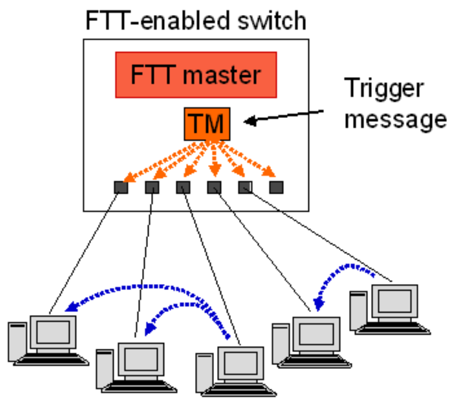

Figure 1. FTT-Enabled Switch.

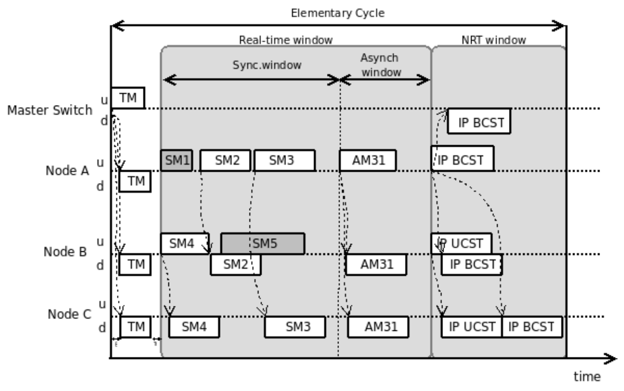

Figure 2. Traffic scheduling in FTT-Enabled Switch.

referred to as synchronous since their transmission is synchronized with the periodic traffic scheduler. This switch also supports aperiodic traffic, called asynchronous, which is managed in the background, in the time left within the EC, after the periodic traffic (Figure 2).

This solution with the FTT master inside of the switch enables preserving all FTT attributes, but at the same time, it permits obtaining important gains in the following key aspects:

- A noticeable reduction in the switching latency jitter found in common in Ethernet switches;
- An important performance boost of the asynchronous traffic, which in this case is autonomously triggered by the nodes instead of being polled by the master node;
- An increase in the system integrity since unauthorized transmissions can be readily blocked at the switch input ports, thus not interfering with the rest of the system.
- Seamless integration of standard non FTT compliant nodes without jeopardizing the real-time services.

3.2. Architecture

Figure 3 shows the functional architecture of the FTT-enabled Ethernet switch. The FTT master functionality, is

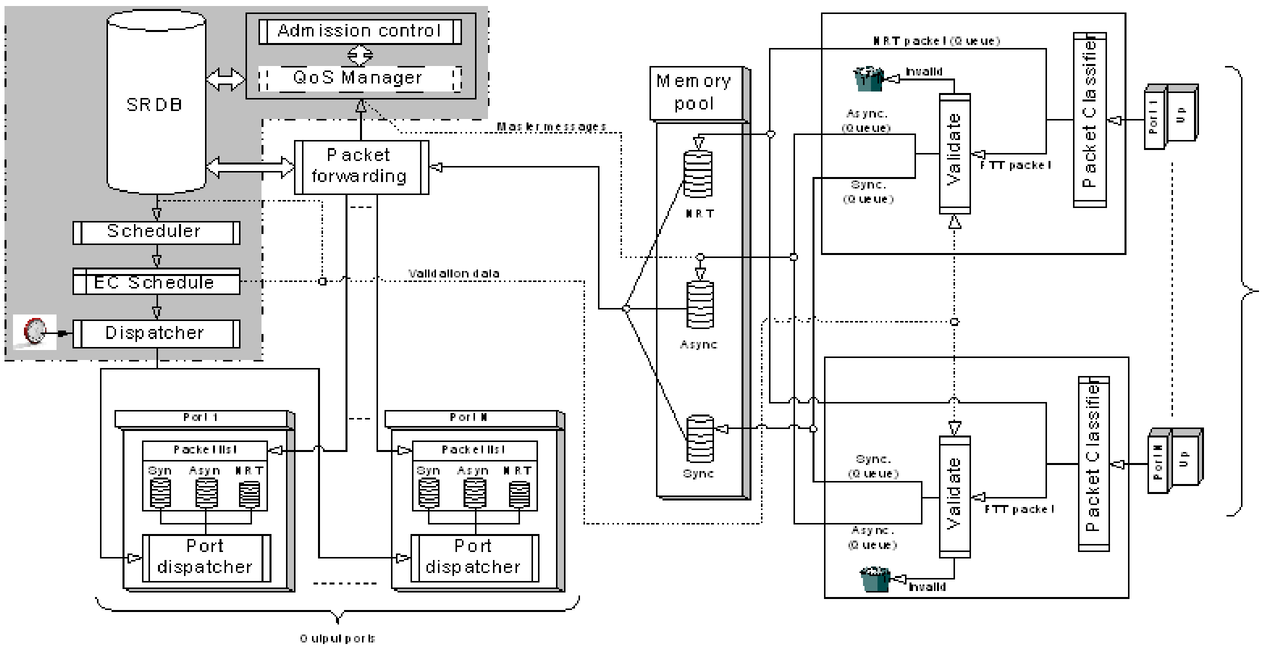

Figure 3. FTT-enabled switch functional architecture.

represented by the shaded area. It includes admission control, Quality of Service (QoS) manager and scheduler functionality. Moreover, it implements a System Requirements Data Base that is a central repository for all information related to the traffic management.

On the right side are the input blocks. Here, the ingress traffic is classified and placed in appropriate memory blocks. Furthermore, real-time traffic is verified in the time domain. Unscheduled periodic traffic or sporadic traffic that violates the minimum inter-arrival time period are trashed. Non real-time traffic is accepted only if the corresponding memory buffer has enough free space.

The global memory pool keeping the messages of each class in independent subdivisions allows avoiding memory exhaustion for the real-time messages, a situation that standard switches do not guarantee. On the other hand, the real-time traffic is subject to an explicit registration. During the registration process the procedures use the flow properties to compute and pre-allocate the amount of memory that guarantees enough resources for all admitted messages.

On the bottom left side are the output blocks. Each port as three pointer queues, one for each traffic class. At the beginning of each Elementary Cycle (EC) the Trigger Message (TM) is generated, afterwards the port dispatcher transmits the submitted packets from each queue, according to the EC schedule provided by the FTT master. This procedure enables to enforce the temporal isolation between the different traffic classes.

This switch is transparent to the nodes that not produce real-time messages. They can use any standard Ethernet driver. The transmission of this class, inside the switch, is according to the normal procedures of standard Ethernet switches, based on the MAC address. The only limitation arises the confinement of this traffic to the NRT phase in the EC. However, the nodes that produce real-time messages require the FTT network driver to be updated to include different queues for the three traffic classes, react to the Trigger Message and avoid blocking of the synchronous traffic in the uplinks.

4. FTT-enabled Switch Implementation

The hardware architecture of the FTT-enabled switch using FPGA technology is shown in Figure 4.

The FTT master, represented by the Master Unit, executes a complex set of operations, namely the admission control, QoS manager and scheduler. It also implements a System Requirements Database to store the information related to the traffic management. Given the algorithmic complexity these functionalities they are implemented in software.

On the other hand, there are several functionalities in the FTT-enabled switch that need predictability, determinism and speed in their execution thus being preferable to execute them in hardware. This group includes the reception process, switching and transmission process. This way, all blocks except the Master Unit are implemented in hardware (figure 4). We will call this set of blocks the Switching Module.

The integration of these two parts, Master Unit and Switching Module, can be performed in different ways, for instance:

- The Master Unit runs in an independent CPU and the communication with the FPGA is carried out by a conventional communication means available in the development board (e.g Ethernet, USB, PCI, ...);
- Utilization of an FPGA embedded processor to runs the Master Unit, either synthetizable or hardwired (e.g MicroBlaze, PowerPC).

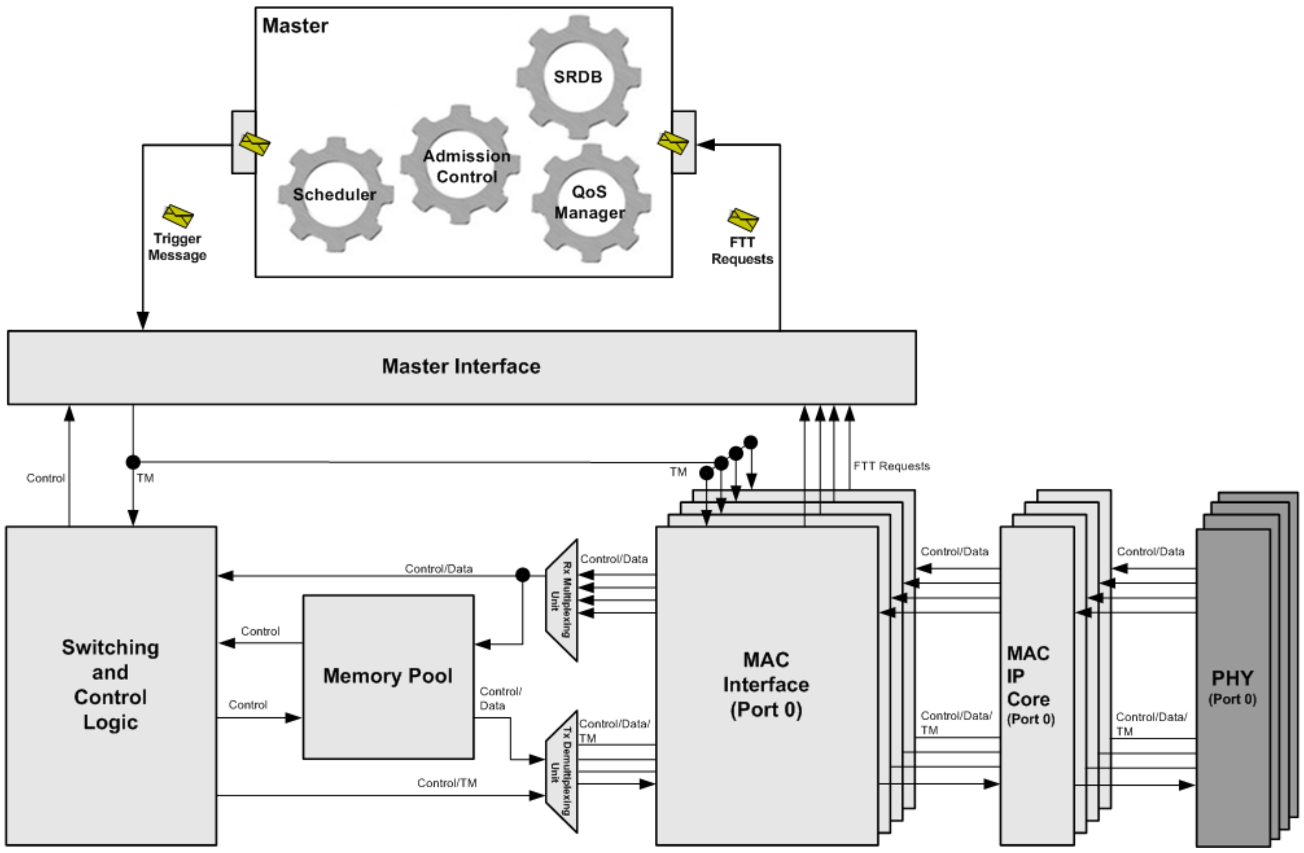

Figure 4. FTT-enabled switch - hardware implementation (top level).

4.1. Hardware Implementation - Top Level

The implementation of the Switching Module is based on FPGA technology, because it allows an easy and flexible integration of practically all components. The only devices implemented outside the FPGA are the Ethernet PHYs due to their electrical characteristics, timing requirements and wide availability of pre-built modules.

Each port has one associated Ethernet Phy directly linked to one Xilinx Tri-Mode Ethernet MAC soft core fully compliant with IEEE 802.3 standard, which can operate at 10/100/1000 Mbits/sec. This MAC core can be implemented on the programable logic resources of Xilinx FPGAs and it is highly configurable and provides the following interfaces:

- Reception Control and data ports;
- Transmission Control and data ports;
- Core management with control, status, address and data ports;
- Ethernet PHY Media Independent Interface.

The MAC Interface Unit, specific for each switch port, implements all logic that allows to configure the MAC IP Core. In the reception, it receives, classifies, validates and handles the data, writing it write in memory. In the transmission, this unit reads the data from the memory, handles it and manages the interface with the MAC IP Core for the transmission.

Another top level unit is the Memory Pool that implements a dual port static Synchronous Random Access Memory - SRAM with separate clocks, control, address and data buses. One port (write only) is shared among all input ports and the other (read only) used by all output ports. The memory is segmented in blocks that allow to store packets with maximum size, and each block can store one packet, only. This mechanism is inefficient with short packets but simpler to manage and enough to prove the concept.

The Control and Switching Logic Unit plays a central role within the switch, performing the reception, switching and transmission management based on the control and status signals provided by MAC Interface Unit. Moreover, it synchronizes all the switch functioning and protocol operations.

The Rx Multiplexing Unit is shared among all switch ports. It is basically a multiplexer (or TDMA wheel) that allows all ports uplinks to write the received messages data into the switch main memory. Contrarily, the TX Demultiplexing Unit is also shared among all switch ports and it is basically a demultiplexer (or TDMA wheel) that allows all ports to read data from the Memory Pool and write it on the corresponding downlink MAC.

The Master Interface Unit implements the interface with the Master Unit, which executes on a suitable hardware platform.

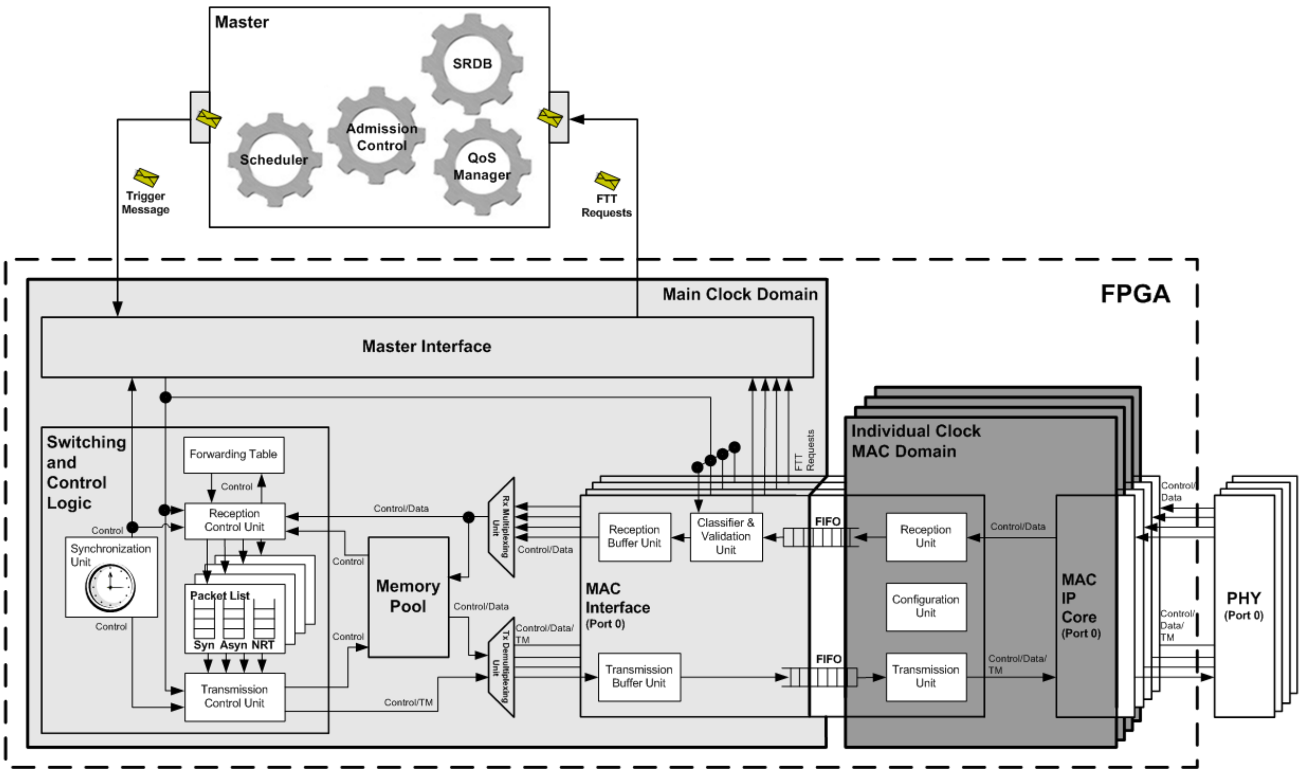

Figure 5. FTT-enabled switch - complete view.

4.2. Hardware Implementation - Full View

Figure 5 represents the complete view of the FTT-enabled switch. This subsection focuses, in particular, on the problem of synchronization inside the switch and the structure of two main units, the MAC Interface Unit and the Switching and Control Logic Unit.

The MAC IP Core provides the clocks that enable the reception and the transmission of data. On the other hand, one independent main clock manages the switch core, composed by three main blocks (Memory Pool, Switching and Control Logic and Master Interface). The union of these different clock domains was possible using FIFOs (included inside the MAC Interface Unit) with independent clocks for the write and read sides. This way, for each port, the information received from the MAC IP Core is written in the corresponding FIFO with the reception clock provided by the same MAC. The reading of the information from the FIFO is performed with the main clock. The transmission of the data is processed in a similar way. The data ready for transmission in a specific port is written in the transmission FIFO with the main clock and read by the MAC with its transmission clock.

In the MAC Interface Unit, the Reception Unit implements the reception interface of Ethernet frames coming from the MAC IP Core. The received data are inserted in the FIFO that separates the clock domains. The next unit in the chain of reception is the Classifier and Validate Unit that performs the classification and the validation of the received packets. For instance, the non-FTT packets are directly delivered to the Reception Buffer Unit. The FTT control packets, that comprise commands to the Master Unit, are delivered to the Master Interface. Finally, the remaining packets still pass through a validation process. If these packets are consistent with the EC-Schedule provided in the Trigger Message by the Master Unit, they are delivered to the Reception Buffer Unit, otherwise, they are trashed. Thus, the integrity of the switch and network is guaranteed. The Reception Buffer Unit accumulates the incoming bytes to form a word N-bytes wide, where N is the number of ports. This way, the writing in the Memory Unit is carried out, per port, with N bytes at a time and at a rate N times slower than the arrival rate of the respective byte stream. After multiplexing all N ports, the wriring frequency at the Memory Unit is equal to the bytes arrival rate at the individual ports, avoiding the need for higher frequency clocks. This technique is also used in the transmission chain by the Transmission Buffer Unit, which receives words N bytes wide from the Memory Unit and forwards the data to the respective FIFO one byte at a time. The reading frequency from the Memory Unit is again equal to the byte transmission rate in the port FIFO. The Transmission Unit reads the data from the FIFO, which allows the separation of clock domains, and manages the sending of this data to the MAC IP Core. Finally, in a different chain, the Configuration Unit implements the logic used to configure the MAC IP Core at startup.

The Switching and Control Logic is a complex unit, where the Reception Control Unit manages the forwarding of the received packets to the output ports. This unit receives, for each Elementary Cycle, the corresponding Trigger Message, which includes the identification of all data packets that will be received from each input port and sent by each output port (EC-Schedule).

Thus, the FTT data packets are forwarded based on the contents of the Trigger Message, only. On the other hand, the non-FTT data packets are forwarded based on the Forwarding Table that is updated dynamically as in common switches. The actual forwarding is carried out by delivering the pointers of the respective packets to the Packet List Unit attached to the corresponding output port, and inserting them in the correct queue. This unit contains three queues, one for each traffic class (Sync RT, Async RT, and NRT), which contain the pointers to the packets that have to be sent in this EC. The Transmission Control Unit controls the transmission of the packets in those queues respecting the corresponding phases in the EC, thus enforcing appropriate trafic confinement of the different traffic classes. Moreover, the Transmission Control Unit only transmits messages from the asynchronous or NRT queues if the time left within the respective windows is enough, thus preventing blocking of the Trigger Message and synchronous traffic. To start the transmission of a packet, this unit asks the Memory Unit to send its data to the MAC Interface Unit of the respective output port. One particular case is the transmission of the Trigger Message, which is sent directly to all MAC Interface Units (broadcast), at the beginning of each EC, as determined by the Synchronization Unit in a blocking-free fashion thus with high precision. The Synchronization Unit controls the EC timing and requests EC-Schedules (Trigger Messages) from the Master Unit.

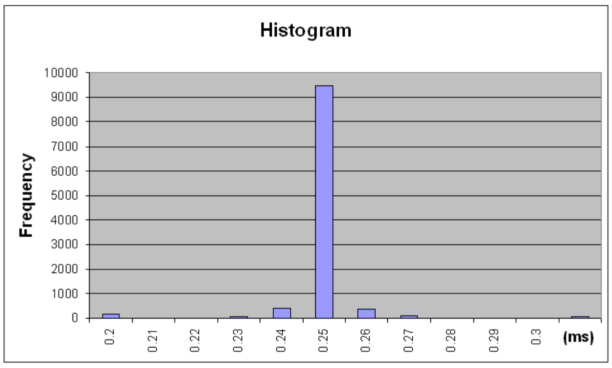

Figure 6. Inter-arrival time of NRT packets at the switch ingress.

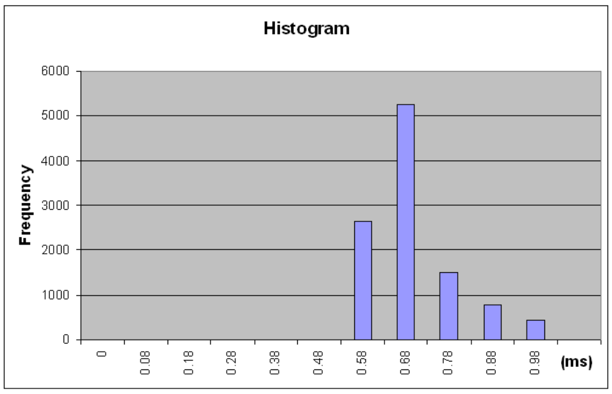

Figure 7. Transmission times of NRT packets at the switch egress with respect to the TM (difference between the transmission timestamps of the packet and the previous TM.

4.3. Hardware Implementation - Experimental Results

In order to validate the main features of the FTT-enabled switch we developed an initial prototype capable of enforcing traffic classification, confinement and temporal validation, as well as fast forwarding and blocking-free transmission [13]. That prototype basically corresponded to the Switching Module that we now propose in the current architecture, since all traffic scheduling features were still missing. Therefore, the performance figures therein referred also hold for our current switching module, particularly the traffic confinement capability (Figures 6 and 7) and the blocking-free transmission of the Trigger Message even in the presence of strong load. In the former case, the figures show the histograms obtained at the switch ingress (inter-arrival times) and egress (relative delay inside the EC) of 10000 NRT packets with 1000B payload each, using approximately 30% of the links bandwidth. In the latter case, the TM was transmitted with a period of 1.000ms with a standard deviation of 138ns and maximum deviation below +/- 200ns.

4.4. Software Implementation

The Master Unit implements a set of complex operations, including the SRDB, the QoS Manager, the Admission Control and the Scheduler. From an implementation point of view, these operations are algorithmically complex and make extensive use of dynamic lists, which are more efficiently implemented in software. Moreover, these operations basically correspond to the functionality of the Master in the FTT-SE protocol [10], which is a fully software implemented version of the protocol that works over COTS Ethernet switches.

Therefore, in order to re-use the FTT-SE Master as the Master Unit, the Master Interface was designed to provide the necessary standard interfaces that FTT-SE uses, namely one Ethernet port. This port, however, is dedicated to the communication with the Master Unit and does not integrate the communication ports managed by the Switching Module. This allows incorporating the FTT-SE Master practically as is, with substantial gains in development time. The FTT-SE Master is then an autonomous component that can be connected to a COTS switch or to the Switching Module herein proposed, providing two different levels of service concerning traffic filtering, confinement, policing, temporal protection and compatibility with non-FTT nodes. This is a highly flexible and efficient solution.

When comparing with the other possibility of using an embedded microprocessor inside the FPGA to run the Master code, the proposed solution is more expensive given the use of an external computer, and possibly less reliable given the larger number of components and connections. However, it has the advantage of maximizing the FPGA resources available for communication ports and it allows sharing the FTT Master between the full software FTT-SE version and the one based on the enhanced switch.

One aspect that had to be worked out was the source of the EC clocking. While in the full software version of FTT-SE it is the Master that controls the EC timings directly, in this case, using the enhanced switch herein proposed the EC timing is controlled by the switch Synchronization Unit. This unit sends requests to the Master Unit that triggers the scheduling activity leading to the generation of the Trigger Messages containing the respective EC-Schedules that are then used by the Switching Module.

This aspect is also relevant when considering multi-switch topologies. In such case, only one Master is used and the source of the EC clocking must be unique, too. Therefore, when using the enhanced switch, it must be the only one of such kind, with all others being COTS switches. For larger networks it is advisable to plan different synchronization domains, each with its own enhanced switch, connected by means of gateways that allow decoupling the different synchronizations.

Concerning the internal components of the Master Unit, the System Requirements Database (SRDB) deserves a special reference. It is a central repository for all the information related to traffic management, namely the messages attributes for both synchronous and asynchronous traffic, e.g., period/minimum inter-arrival time, length in bytes, priority if applicable, deadline and offset if applicable, plus information about the resources allocated to each traffic class, e.g., phase durations and maximum amount of buffer memory, and global configuration information, e.g., elementary cycle duration and bit rate.

The attributes in the SRDB can be updated on-line via FTT requests that are identified by the Switching Module and forwarded to the Master Unit. These requests can ask for addition or removal of messages to/from the SRDB, for example, in the course of on-line reconfiguration procedures, or even changing attributes of existing messages, for example, in the course of dynamic QoS management. In both cases, the FTT requests arriving at the Master Unit are submitted to an Admission Control that guarantees that there are resources enough to enforce the timeliness of the real-time messages. When active, the Qos Manager adapts, upon request, the attributes of a set of messages in order to maximize a given figure of merit.

Finally, the Scheduler scans the SRDB on-line, every EC, and builds the list of real-time messages that must be produced in the following EC (EC-Schedule). This list is incorporated into the Trigger Message and sent to the Switching Module where it is buffered. The right transmission instant is defined by the Synchronization Unit, ensuring a precise timing.

5. Conclusions and Future Work

The advent of switched Ethernet has opened new perspectives for real-time communication over Ethernet. However, a few problems subsist related with queue management policies, queue overflows and limited priority support. While several techniques were proposed to overcome such difficulties, the use of standard Ethernet switches constraints the level of performance that may be achieved. On the other hand, the growing availability of powerful programmable hardware devices and associated tools as well as IP cores of communication components opens the way to build customizable devices with properties that are better tuned to specific application domains.

Therefore, following such trend we proposed recently an FPGA-based enhanced Ethernet switch relying on the Flexible Time-Triggered paradigm that enforces strict temporal isolation of three traffic classes, provides seamless integration of non-FTT nodes without causing any interference on the periodic real-time traffic, also provides filtering of unauthorized transmissions at the switch ingress and generates a high precision time mark.

In this paper we presented an architecture for such enhanced switch that exploits the separation between the packet switching activity and the FTT Master functionality, the former being implemented in hardware (Switching Module) and the latter in software (Master Unit). This also allowed reusing the Master of the fully software implemented FTT-SE protocol with minor adjustments and great benefits in modularity and development costs. The paper also described the hardware implementation of the Switching Module, with a focus on the synchronization issues among the asynchronous units inside the switch.

On-going work addresses the full characterization of the overheads incurred by the proposed architecture concerning the interconnection between the Switching Module and Master Unit to derive performance limits. Two other issues will also be addressed, namely the adaptation of the enhanced switch to allow the coexistance of several units in the same synchronization domain and the replication of the Master. The design of specific gateways for connecting different synchronization domains will also be addressed.

6. Acknowledgments

The authors would like to thank Xilinx Inc. for the donation of the Tri-mode Ethernet MAC soft IP core, as well as ISE and ChipScope Pro FPGA design tools.

References

[1] Automation.com magazine. http://www.automation.com/content/softing-uses-altera-fpga-for-real-time-ethernet.

[2] Ethernet powerlink - online information. http://www.ethernet-powerlink.org/.

[3] Open DeviceNet Vendors Association. Ethernet/ip. http://www.odva.org/.

[4] J. Feld. Profinet - scalable factory communication for all applications. In *2004 IEEE International Workshop on Factory Communication Systems*, pages 33–38, 2004.

[5] P. Ferrari, A. Flammini, D.Marioli, and A. Taroni. Experimental evaluation of profinet performance. In *2004 IEEE In-*

ternational Workshop on Factory Communication Systems, pages 331–334, 2004.

[6] EtherCAT Technology Group. Ethercat - ethernet for control automation technology. http://www.ethercat.org, December 2007.

[7] H. Hoang, M. Jonsson, U. Hagstrom, and A. Kallerdahl. Switched real-time ethernet with earliest deadline first scheduling - protocols and traffic handling. In *WPDRTS'02 - The 10th International Workshop on Parallel and Distributed Real-Time Systems*, page 308, Florida - USA, April 2002. IEEE Computer Society.

[8] J. Loeser and H. Haertig. Using switched ethernet for hard real-time communication. In *PARELEC'04 - International Conference on Parallel Computing in Electrical Engineering*, pages 349–353, Dresden - Germany, September 2004. IEEE Computer Society.

[9] R. Marau, P. Pedreiras, and L. Almeida. Enhanced ethernet switching for flexible hard real-time communication. http://www.csem.ch/events/RTN06/RTN06.html, jul 2006. RTN 2006, 5th Workshop on Real-Time Networks, Dresden, Germany.

[10] R. Marau, P. Pedreiras, and L. Almeida. Enhancing real-time communication over cots ethernet switches. In *WFCS 06 - The 6th IEEE Workshop on Factory Communication Systems*, Turin - Italy, June 2006. IEEE Computer Society.

[11] M. Plankensteiner. Ttethernet enabes the use of ethernet networks in all applications. *Embedded Control Europe*, pages 12–14, 2008.

[12] PROFInet. Real-time profinet irt. http://www.profibus.com/pn, December 2007.

[13] R. Santos, R. Marau, A. Oliveira, P. Pedreiras, and L. Almeida. Designing a costumized ethernet switch for safe hard real-time communication. In *2008 IEEE International Workshop on Factory Communication Systems*, pages 169 – 177. IEEE Computer Society, May 2008.

[14] K. Steinhammer, P. Grillinger, A. Ademaj, and H. Kopetz. A time-triggered ethernet (tte) switch. In *DATE'06 - Design Automation and Test in Europe*, pages 794–799, Munich - Germany, March 2006. ACM.

[15] TTTech. Ttethernet. http://www.tttech.com/solutions/ttethernet/, November 2008.