Adam ZIĘBIŃSKI[1], Rafał CUPEK[1], Piotr PIĘKOŚ[2], Łukasz HUCZAŁA[1]

Silesian University of Technology, Institute of Informatics (1), ASML B.V., Veldhoven, The Netherlands (2)

A frame filter IP core for RT-Ethernet monitoring

***Abstract**. The novelty of this paper is the concept for an RT-Ethernet Frame Filter that is dedicated for RT network analyzers. In classic network analyzers the hardware part of the analyzer is dedicated for a given network protocol or a set of protocols. The captured frames are selected by the software part of a classic network analyzer. In the proposed solution the authors describe the FPGA-based hardware network TAP which includes an internal filtering structure that may be adjusted during network analyses. This paper clarifies the hardware filtering module concept, together with its principle of operation and introduces a potential solution for RT-Ethernet traffic filtration that is embedded into the FPGA structure as a configurable IP core called an Ethernet Frame Filter.*

Streszczenie.** Artykuł przedstawia koncepcję sprzętowej filtracji ramek sieciowych dla dedykowanych urządzeń analizujących sieć Ethernet czasu rzeczywistego (RT-Ethernet). W klasycznych urządzeniach analizujących sieci, część urządzenia analizującego jest dedykowana dla danego protokołu sieci lub zbioru protokołów. Wychwycone ramki w dalszym etapie są przetwarzane przez klasyczne oprogramowanie urządzenia analizującego sieć. W proponowanym rozwiązaniu autorzy opisują rdzeń filtra ramek Ethernet „RT-Ethernet Frame Filter", dedykowany dla układów FPGA. Filtr ten pozwala na modyfikację swej wewnętrznej struktury w taki sposób, by mogła być przystosowana do wyspecyfikowanych przez użytkownika dowolnych testów, niezależnie od stosowanych protokołów sieciowych.* ***Koncepcja sprzętowej filtracji ramek sieciowych dla dedykowanych urządzeń analizujących sieć Ethernet czasu rzeczywistego

Keywords: FPGA, Frame Filter, IP core, Industrial Networks, network analyzer, network TAP, PROFINET I/O, RT-Ethernet.
Słowa kluczowe: FPGA, Filtr ramek, rdzeń IP, sieci przemysłowe, analizator sieci, sieciowy TAP, PROFINET I/O, RT-Ethernet.

doi:10.12915/pe.2014.10.53

Introduction

Due to the large number of packets that are cyclically transmitted by an industrial Real-Time Ethernet (RT-Ethernet) [1],[2],[3],[4],[5], network diagnosis using packet-analysis methods has become a task that requires a great deal of computational power and results in enormous memory consumption. One of the solutions that permits the amount of data that is received by dedicated network-analysis software to be reduced [6],[7],[8] is a filtering tap that is implemented as dedicated hardware [9], [10], [11], [12]. Filtering is also used in a hardware firewall for IP networks [13], [14].
This paper presents the concept of a hardware filtering module for RT-monitoring.
The main motivation of the work is to reduce the netload for the analyzing device by removing unwanted frames. The presented solution is programming by the softcore processor and allows the filter conditions for each bit in the frame to be set. Thus, we can filter not only standard field network frames (such as IP address, Protocol Version or MAC address field) but also other specific data such as a frame alarm or measurement data that exceed the set threshold. The filter mask is programmable in terms of Ethernet fields and allows the frame fields from an incoming packet to be compared and the frame fields to be stored in the configuration module. The specified frames are stored in the internal memory. Furthermore, compared to the commercially available solutions, the presented filter may be extended and additional functions that are related to the on-the-fly frame processing can be added to the FPGA structure. The authors believe that the proposed solution can be used not only for statistical analyses of Ethernet network traffic but that it can also be easily extended to the analysis of on-line information content.

This paper clarifies the hardware filtering module concept (Fig. 1) together with its principle of operation and introduces a potential solution for RT-Ethernet traffic filtration that is embedded into the FPGA structure as a configurable IP core called an RT-Ethernet Frame Filter (RTEFF). The following assumptions drive the design of RTEFF IP core: The RTEFF IP-core is a configurable filtering module that is designed to operate in a 10/100 Mb/s Ethernet environment. The RTEFF is compatible with the Altera SOPC builder tool and fully integrates with the Altera Avalon bus.

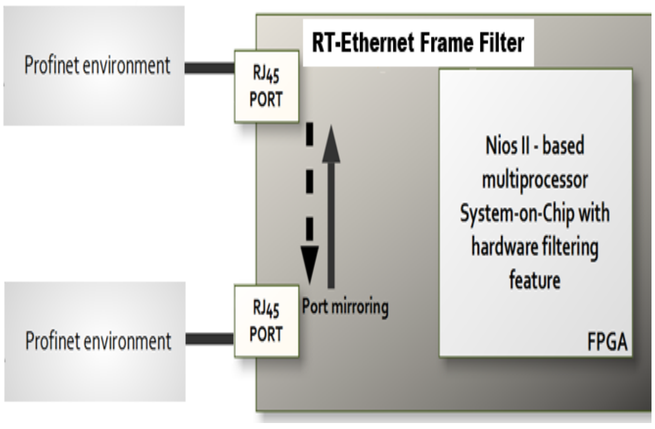


Fig.1. The RT-Ethernet Frame Filter concept

The project should fulfill the following assumptions:
- The RTEFF module is capable of filtering incoming Ethernet packets with respect to a predefined configuration. Uploaded filtering schema may vary from simple logical conditions to complex logical sentences.
- The RTEFF IP-core works as both a datagram sink from the Media Access Controller and as a source of filtered packets to any external buffer that is compatible with Altera's Avalon Bus. Information about the availability of a packet on the output of the RTEFF is provided by an interrupt that is asserted by the module.
- Filtering process duration – understood as the time that is measured from the moment of the arrival of a packet into the buffer up to time when the information about the validity of the packet is provided to the software part of the network analyzer – is constant and negligible from the point of view of the timing constraints that are defined by the 100Mb/s RTE. Thus, the performance of the system can be estimated regardless of the internal structure of the RTEFF Core module.
- The RTEFF IP core operates together with a software driver operating on the Hardware Abstraction Layer of a Nios II embedded processor [15],[16],[17],[18],[19],[20].

This driver creates an accessible and flexible interface between the IP core and the high level programming structures.

This paper is organized as follows: Section 2 presents the IP Core operation principle, Validity Function and 3-stage filtering. Section 3 shows the Structure of the Ethernet Frame Filter IP Core. Section 4 presents the prototype of the Filtering System and tests. Section 5 concludes the paper.

IP Core operation principle

A rough idea of hardware filtering consists of two main stages. The first stage is at the software level. The filtering schema is defined by the user and interpreted by the software. The driver mask is translated into the proper low-level register configuration of the IP core. This will be referred to later as the „Configuration Phase" of the Ethernet Frame Filter (EFF). The second stage begins after the configuration is uploaded, data capturing begins. The Media Access Controller inserts the packets directly into the internal buffer of the EFF. This is done via the DMA channel. If the packet has arrived to the buffer successfully, the MAC asserts an interrupt thereby informing the CPU about its arrival. This causes an update (software controlled) of the EFF status registers and starts the filtering process. An interrupt is asserted depending on the filtering result informing that the frame matched the filter's current configuration and can be sent to the external buffer (IP stack, SDRAM etc.). If the filtering result has been positive, the packet is copied. This concludes the data capturing phase, which repeats itself in the loop until the filtering process is terminated. Depending on the network's payload and the reference clock driving IP core, another frame might be copied into the RTEFF buffer while the previous one is being sent. In order to provide fast and „transparent" filtering, the buffer space is paged – the internal controller alternatively inserts frames into the upper or lower part of the buffer. Since dual-port memory is implemented, parallel reading from the lower part is allowed while the upper part is being overwritten by the next incoming frame and vice versa.

Case Study

Let us consider a simplified industrial network structure that is composed of two PROFINET I/O [21] industrial network switches. Each of the switches services two PROFINET I/O compatible devices: a PLC and IO module. In this case, the RTEFF module is a part of a System on Programmable Chip (SOPC) that transparently captures all of the information that appears on the medium connecting both switches. The captured PROFINET I/O packets can be further sent to external diagnostic software using the standard 100Mb/s Ethernet. Depending on the filtration scenario, all, none or some of the frames can be transmitted outside the PROFINET I/O environment. In the presented case, the primary objective is a reduction of data so that only packets that are important from the point of view of diagnostics are passed outside of the PROFINET network. Let's assume that from the point of view of diagnostics, major stress is put on three crucial data streams:

- The PROFINET I/O frames (Ethertype 8892) that are transmitted from the first PLC to the PROFINET IO module that is plugged into a different switch than the first PLC.
- All of the data are sent from the first to the second PLC (connection establishment, control protocols etc.) except for the PROFINET frames. This means that normal, real-time communication will be cut-off.
- All alarms may occur between the second PLC and the field device that is plugged into a different switch than the PLC. One would like to monitor the alarms that are coming from both devices.

The scenario described above can be synthesized into a logical sentence. The different elements of the packet can be abbreviated by certain keywords. By introducing a simplified notation that includes elementary logical operators and parentheses, one can define a script language that allows a complete definition of a complex logical sentence – the filtering schema. Keywords and the syntax of such a script language are similar to the one that is used in the common packet analysis software – Wireshark. Since it is widely recognized among industrial network technicians, the decision to follow this existing nomenclature is straightforward. The example could be rewritten as a short script that defines the filtering schema (Fig. 2).

```
(eth.dst=PN_DEVICE_B&&eth.src==PLC_A&&eth.type==0x8892)||
(eth.dst= PLC_C&&eth.src==PLC_A&&!(eth.type==0x8892))||
((eth.dst= PN_DEVICE_A&&eth.src==PLC_C)||
(eth.dst== PLC_C&&eth.src==PN_DEVICE_A 0x8892))||
&&alarms_all)
```

Fig.2. The filtering schema.

Validity Function

Subsequently, one can distinguish several (seven in this example) conditions inside the filtering schema as presented in Fig. 2. Each assignment created in this way possesses information about the elements of the filtration schema; however, it is atomized and irrelevant to the structure of the logical expression. Using the assignments shown in Fig. 3, one can construct an abbreviated form of the filtration schema. Such an expression can be rewritten as a validity function Q.

$$\begin{aligned}
&a \leftarrow \text{eth.dst} == \text{PN\_DEVICE\_B}\\
&b \leftarrow \text{eth.src} == \text{PLC\_A}\\
&c \leftarrow \text{eth.type} == \text{0x8892}\\
&d \leftarrow \text{eth.dst} == \text{PLC\_C}\\
&e \leftarrow \text{eth.src} == \text{PLC\_C}\\
&f \leftarrow \text{eth.src} == \text{PN\_DEVICE\_A}\\
&g \leftarrow \text{alarms\_all}
\end{aligned}$$

Fig.3. The logical expressions.

Function Q, which is presented in Fig. 4, is a Boolean function that takes a set of Boolean conditions a…f as an argument.

$$Q = abc + db\bar{c} + (de + df)g = abc + db\bar{c} + ged + gdf$$

Fig.4. The Function Q.

Fig. 4 shows function Q after substitution and – on the right hand side – its unfolded variant, so that function Q is defined as the sum of products (SOP). A condition that occurs two or more times within one product section of the SOP is simplified into a single condition. The occurrence of a condition and its complement within one product section nullifies this part of the sum. If function Q results in a Boolean true (logical one) value, the current Ethernet frame that is under investigation matches the filtration schema. The sum of the minterms that form interchangeably defines the structure of the filtration schema.

3-stage filtering
In order to provide a scalable (independent of the complexity of the logical expression defining the filtration schema) method of accessing the structures and conditions, those two factors are divided and resolved separately during the filtering process. This approach results in the concept of three-stage filtering, in which each packet undergoes three processing levels – Virtual Lan Tag removal, a comparison of the values present in the frame with the ones defined by the filtering schema and eventually the resolution of the comparison results in a structural context. Figure 5 illustrates the concept of 3-stage filtering.
The diagram represents the conceptual approach to the IP core of an EFF operation. The frame input is presented on the left-hand side. The first stage is the VLAN Tag Detector module, which is responsible for the removal of VLAN tags (if they are present). This is done to preserve the offsets of the elements inside the packet. After this preliminary operation, stage 2 begins. Every significant element of the packet is dissected and passed to the appropriate comparison section that is located inside the Comparison Module. In the Comparison Module, a valid signal occurs when the source from which the frame was emitted matches the source address that was uploaded to the comparator during the configuration phase.

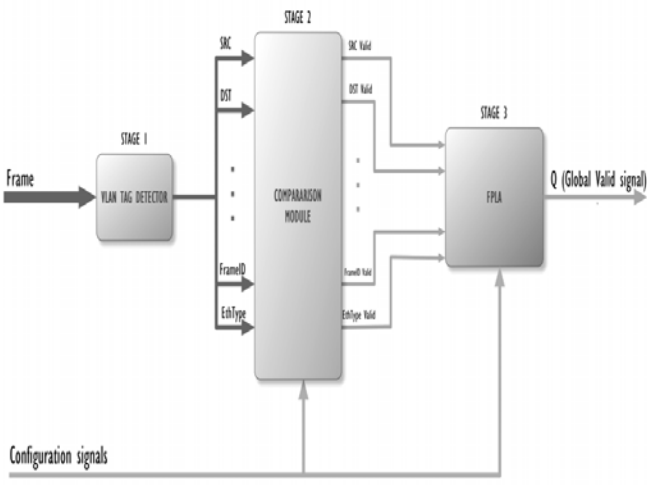


Fig.5. The concept of 3-stage filtering.

A valid signal is then passed to the FPLA module where it is resolved in a structural context. The final third stage takes place in the FPLA module. The valid signals that are produced by the Comparison Module are input into the FPLA module during the filtering operation. The FPLA module resembles a Programmable Logic Array when it comes to its operation principle. This allows the FPLA module to be preconfigured (during the configuration phase) so that it implements an arbitrarily given logical function in an SOP form; therefore, it is used to incorporate the Conditions from the Comparison Module into the structure of the Validity Function. The output of the Validity Function is the information about whether the packet matches the filtration schema that was defined during the configuration phase. Taking into consideration the sum of minterms form of validity function Q that was presented in the example, the internal process of structural resolution can be viewed as a resolution of the combinational function (Fig. 6)
The black dots indicate logical connections to each of the multi-input logical AND gates. They are labeled T0, T1, T2, T3 and T4, respectively. The output of Terms is connected to the multi-input OR gate that implements the summation of minterms. The output of the validity function is the main feedback to the EFF controller, which informs it whether a particular packet matches the filtration schema.

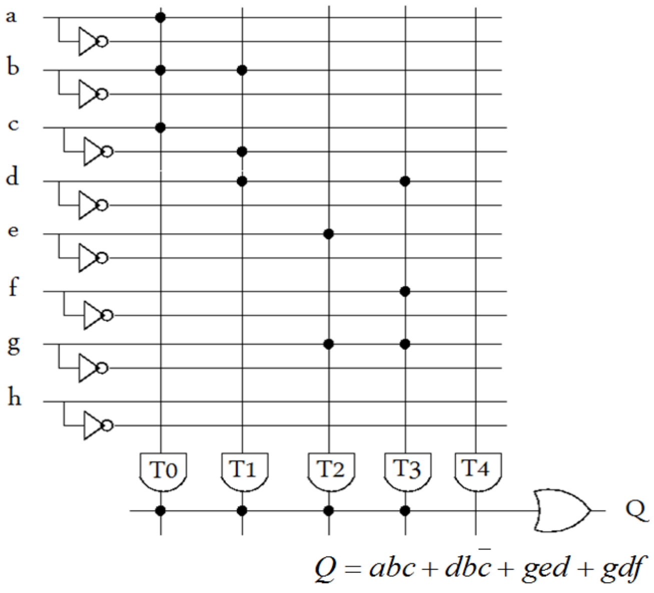


Fig.6. The structure of the comparison Module.

Structure of the Ethernet Frame Filter IP Core
The internal structure of an Ethernet Frame Filter IP core (Fig. 7) consists of three top-level sub-modules: the Frame Buffer, the Filter Controller and the Filter Core. The Frame Buffer is responsible for storing the incoming data (regardless of the filtering result). The Filter Controller provides control signals to the remaining modules. The Filter Core module performs filtering operations and VLAN Tag cut-off, if necessary.

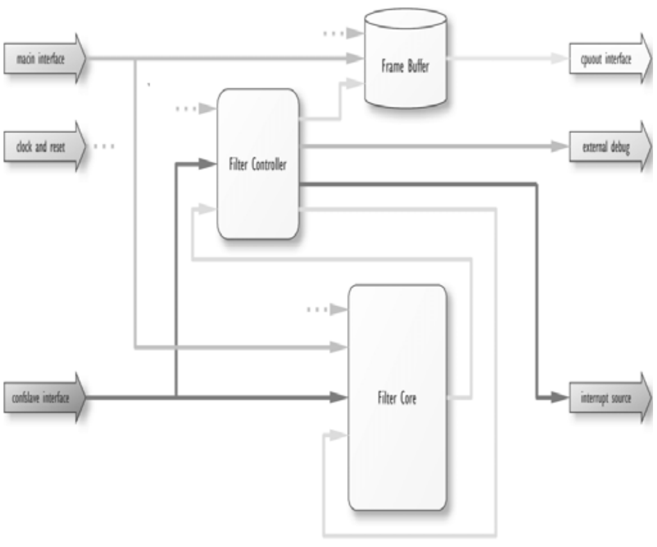


Fig.7. The Ethernet Frame Filter Modules.

Filter Core
The Filter Core was prepared in accordance with the concept of 3-stage filtering (described in Section 2 3-stage filtering). Additionally, it was fitted into the configuration module. The Filter Core (Fig. 5) processes incoming packets and depending on the predefined EFF configuration tags them as matched or mismatched. The configuration registers from which the information that was sent from the software during the configuration phase are located here.

Before the Filter Core can provide output information about the validity of a packet, it must be activated by setting up an enable signal. The Filter Controller asserts this signal after a packet has arrived to the internal buffers of EFF. Obviously, this assumes that the EFF was properly configured prior to the start of the filtering. After enabling the Filter Core, the packet is processed in a consecutive sequence, where one sub-module activates the next one after finishing its operation.

Firstly, the VLAN Tag detector is enabled. After four clock cycles, an activation signal is passed to the Comparison Module, which eventually (after a one clock cycle delay) enables the FPLA sub-module. After resolution of the filtering schema sequence (taking two clock cycles), an output, which informs the controller about whether the actual packet matched the filtering schema or not, is generated. The Frame Memory Submodule is the storage buffer for the first 256-bits of each packet that is received by the Ethernet Frame Filter IP core. This module stores only the first 32 bytes of each incoming frame. The rest of the packet is neglected. The first 32 bytes of every Ethernet frame (not counting the preamble) comprises the destination and source addresses, the Ethernet type and the frame ID.

The VLAN Tag Detector sub-module searches for a specific Tag Protocol Identifier (TPID) that is located inside the packet (value 0x8100 identifies VLAN tagged frame) and corrects the offsets of the Ether Type and the FrameID fields by removing the VLAN tag from its original position. The frame with the VLAN Tag removed is then available in the sub-module's output after four clock cycles. This time delay is needed in order to stabilize the information on the output due to the width of the bus.

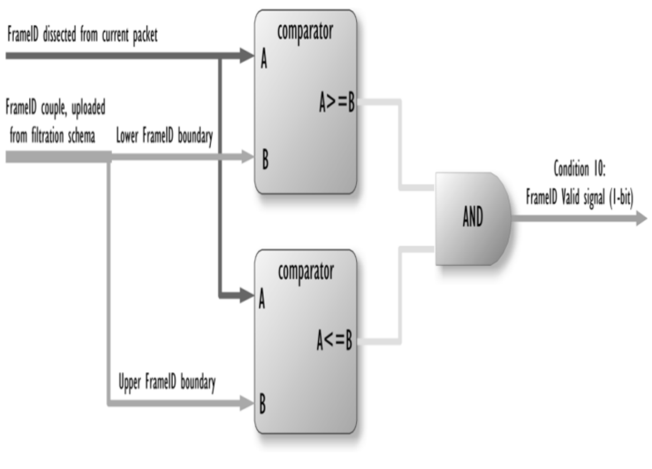

FrameID dissected from current packet
FrameID couple, uploaded from filtration schema
Lower FrameID boundary
Upper FrameID boundary
comparator
A
B
A>=B
comparator
A
B
A<=B
AND
Condition 10: FrameID Valid signal (1-bit)

Fig.8. The topology of the comparison module.

The Configuration Submodule is a custom tailed single input, multiple output memory made up of the register's matrix. It is used to store the information about the filtering conditions that were uploaded during the configuration phase. Instead of the typical output, every condition that has been uploaded to the module can be accessed in parallel.

The Comparison module (Fig. 8) consists of a number of comparators. Their outputs are the Conditions that are streamed to the FPLA module. The comparators used for the analysis of the FrameID are coupled together with an AND gate. Each comparator sets its output high if the relation between the FrameID field from an incoming packet and the FrameID that is stored in the Configuration module is „greater or equal" or „lower or equal", respectively, for the lower and upper FrameID interval boundary.

Figure 9 depicts the FPLA Control Submodule that incorporates the third filtering stage that deciphers the structural context of the Conditions when supplied with the set of Conditions from filtering stage two..

As was mentioned in the Filtering Example section, its principle of operation is based on its ability to flexibly create any combinational logical function in the sum of minterms form. The Conditions (comparison results) that are provided are taken as the function's arguments. During the configuration phase, the structure of the logical function needs to be uploaded to the module's internal registers. Inside the module, the validity function is constructed of a single 8-input OR gate whose inputs are supplied by the outputs of the eight Minterm AND gates.

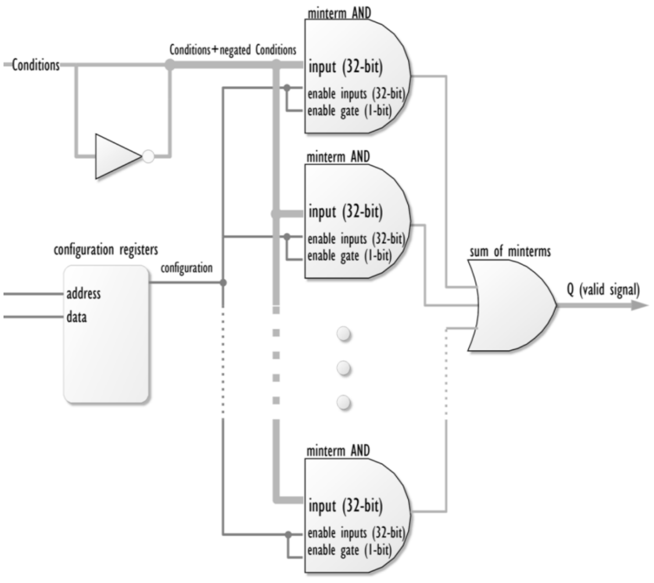


Fig.9. The topology of the FPLA Submodule

Frame Buffer

The Frame Buffer module is a primary packet storage element in the EFF design. It is implemented as a dual-port RAM that is able to store two frames of a maximum length that is defined by the IEEE 802.3 norm. The Frame Buffer interface is a standard dual-port RAM set of inputs and outputs. The module is synchronized with a common reference clock signal. The memory space of the Frame Buffer is organized into two logical parts: the *Lower Memory Area (LMA)* and the *Upper Memory Area (UMA)*. A finite state machine sends the incoming packets to the LMA and UMA consecutively. If the FSM inserts the packet into the LMA, the previously written packet can be optionally read from the UMA. Similarly, if the packet is written into the UMA, only the LMA is available for reading.

Filter Controller

The Filter Controller is a discrete control module that governs the filtering process together with the external CPU. The Filter Controller monitors and affects the operational conditions of the EFF. It is composed of two parts:

- A set of registers that is accessible from outside the module (Interrupt servicing, status registers). This part also provides external debug information about the most important signals of the system.
- A finite State machine (Fig. 10) that directly controls the activation of the Filter Core and the Frame Buffer,

depending on the main feedback signal that is provided by the Filter core.

Several other operational phases can be distinguished during the Filtering Phase and the Configuration Phase. After powering up, the assertion of the reset signal or termination of packet capturing, the IP core puts itself into the Idle Phase. In between successive Filtering Phases, packets are written into the Frame Buffer – this step is denoted as the Writing Phase. Eventually, in the event of the occurrence of an interrupt, the packet needs to be sent further into the external memory storage or buffer, whilst the next packet comes in parallel from the Media Access Controller – this happens during the Sending/Writing Phase. As a result of paging, the Filtering Phase, Writing Phase and Sending/Writing Phase each appear as two different types. Each of the two refers to a different memory area (Lower Memory Area or Upper Memory Area).

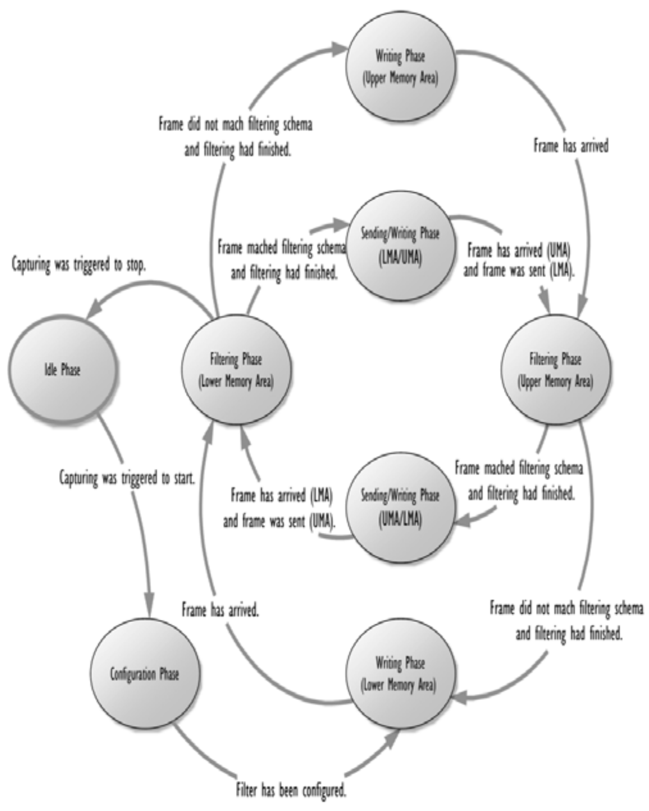

Writing Phase (Upper Memory Area)
Frame did not mach filtering schema and filtering had finished.
Frame has arrived
Sending/Writing Phase (LMA/UMA)
Frame mached filtering schema and filtering had finished.
Frame has arrived (UMA) and frame was sent (LMA).
Capturing was triggered to stop.
Idle Phase
Filtering Phase (Lower Memory Area)
Filtering Phase (Upper Memory Area)
Capturing was triggered to start.
Frame has arrived (LMA) and frame was sent (UMA).
Sending/Writing Phase (UMA/LMA)
Frame mached filtering schema and filtering had finished.
Frame has arrived.
Frame did not mach filtering schema and filtering had finished.
Configuration Phase
Writing Phase (Lower Memory Area)
Filter has been configured.

Fig.10. Filtering phases of EFF

Filtering System

An Ethernet Frame Filter IP core was used in prototype Filtering System (FS) (Fig. 11). It incorporates a mirroring feature that allows for transparent „infiltration" of a Profinet environment. Two central processing units control the two Ethernet Frame Filter IP cores in order to handle the full duplex communication on the wire. The Cyclone III FPGA (40k logical elements) on the DBC3C40 board [15] was used as a reference development board for this project. An on-board reference clock generator provided a 50 MHz signal. It was later divided by a Phase-locked loop module in order to generate a 75 MHz reference clock for all of the peripherals that were included in the System on a Programmable Chip design. The prototype system had proven to be a successful implementation of a minimalistic approach to hardware packet filtering. The entire system occupies 14869 logic elements after fitting into a 3C40 Cyclone III FPGA (38%).

The idea was to capture the packets that appear on the TX and RX channels separately using two Ethernet Frame Filter modules. In order to provide sufficient performance for this operation, the triplet CPU-EFF-MAC was doubled so that each maintained one of the two channels. The configuration was uploaded through an Ethernet Port that was located on the board.

This eliminated the rather inconvenient procedure of sending the configuration either through Profinet environment or removing the existing connection and plugging in a PC with software to generate the configuration for the time of setting up the FS.

The settings for the filter are passed to the system by sending a special Ethernet frame to an address of the device. The ether type must be equal to 0x9999 and after that data block must begin with 9999 again, which informs the device that the frame contains data that is to be set in the filter. After this preamble any logical expression for the filter set in the form of a string can be included. The values for the source and destination MAC address should be separated by " : ", ex. (dst == 45:16:A4:64:31:91). The values for the ether type should be preceded by " 0x ", e.g., (ethtype == 0x8892). The Frame ID should be given only in the form of the keywords that were predefined in the syntax file, ex. (frameid == Alarm_High). A logical command should be expressed as " || " for OR, " && " for AND. A logical condition should be expressed as " == " for EQUAL," != " for UNEQULAL.

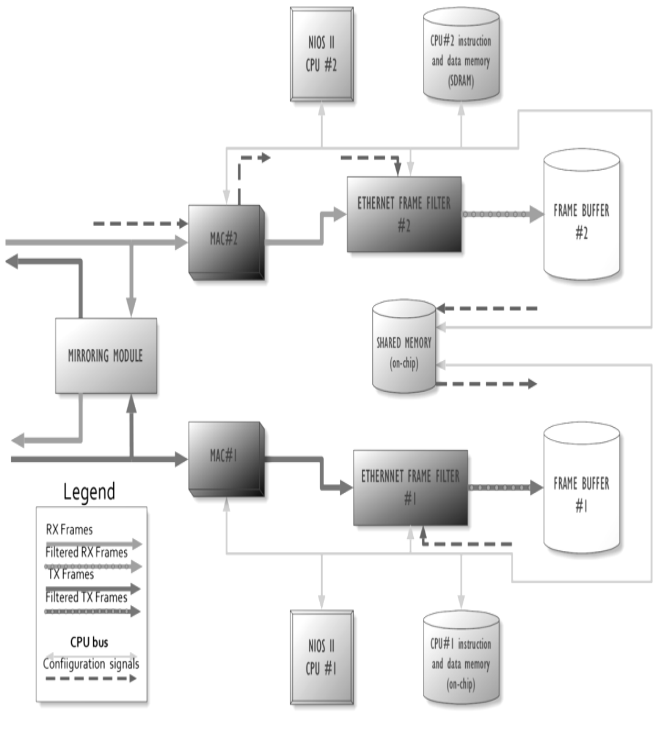


Fig.11. Internal Structure of the Filtering System.

After the FS has been properly configured, it can be safely inserted back into the Profinet environment. The most significant alteration of the system is the addition of a Mirroring Module, which allows transparency of an FS that is connected between any two Profinet devices.

Filtered packets are stored in internal frame buffers. Frame Buffer #1 and Frame Buffer #2 are both on-chip memories of a very limited volume; therefore, they are organized as circular buffers. After they are filled up, the new arriving frames overwrite the ones that arrived at the beginning of the filtration. The configuration procedure differs slightly due to the fact that CPU #1 cannot be accessed directly. Dashed grey arrows indicate how the configuration is uploaded to EFF #1 via the shared on-chip memory (a connection point between two CPU's). The mirroring module is a part of the Filtering System that is located inside the FPGA; however, it is located outside the NIOS II SOPC system. It possesses two features – mirroring and conversion from Reduced Media Independent Interface (RMII) [22] to Media Independent Interface (MII) [23]. The Mirroring Module actually consists of two RMII-MII conversion modules, where data signals from Instance 1 of the converter supply „MAC outputs" interfaces with Instance 2 and vice versa.

Each converter generates 120ns of delay – in total a 240ns delay is introduced inside the FPGA for the packets that pass through the FS. Although the PHY delay cannot be precisely determined, empirical tests have proven that the total delay of both physical and logical layer of the Ethernet Frame Filter System is low enough to fulfill Isochronous Real Time [8] communication requirements.

The progress of filtration and its results can be viewed through the NIOS II IDE [16] debug perspective or on the seven segment display that is located on the top of the board. In order to read the filtered frame, it should be sent the special Ethernet frame at the address of the device. The ether type must be equal to 0x9999 and after that the data block must begin its destination MAC address and next block 9998 for Frame Buffer #1 or 9997 for Frame Buffer #2.

Test Environment

The system structure was tested using three Profinet IO devices and a Siemens PLC (S300 Series) with built-in Profined I/O Controller functionality. All of the devices were connected to an 8-port Siemens switch (Fig. 12). The above configuration used about 15-20% of the maximum Ethernet Bandwidth. The properly prepared RT-Ethernet configurations that are used in industry do not exceed 15-20% (including other traffic). The authors assumed that errors due to network overload could be easily detected by the standard tools that are used for Ethernet communication. All PROFINET I/O frames used VLAN tags.

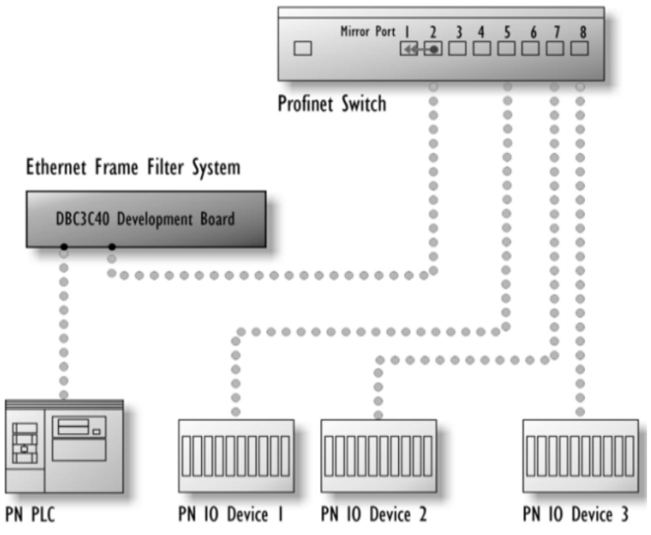

Mirror Port 1 2 3 4 5 6 7 8
Profinet Switch
Ethernet Frame Filter System
DBC3C40 Development Board
PN PLC
PN IO Device 1
PN IO Device 2
PN IO Device 3

Fig.12. Testing environment

Various patterns of the filtration were used to study the behavior of the system:

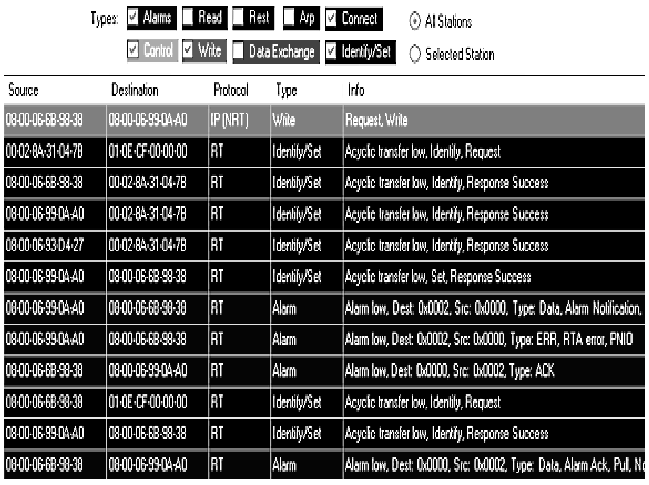
- bidirectional and unidirectional communication between one of the IO devices and PLC
- alarm frames which were passed from I/O device to the PLC
- multicast and broadcast frames
- real time traffic

More sophisticated expressions utilized all of the features of the script language, for example:

(src==08:00:06:99:0A:A0&&||src==08:00:06:6B:98:38)&& (dst==08:00:06:6B:98:38||dst==08:00:06:99:0A:A0)&& frameid==All_Alarms

Two types of frames can be seen in Fig. 13. Alarm frames are marked in gray and have a higher priority than the other frames. Each alarm frame is confirmed by a recipient. IdentifySet frames have also been confirmed but since they do not belong to any time critical communication channel, they have a lower priority than alarm frames.

A combination of different filtration schemas was uploaded to the EFF and positive results were obtained in every set of circumstances that was investigated.



Types: [✓ Alarms] Read Rest Arp [✓ Connect] [✓ All Stations]
[✓ Control] [✓ Write] Data Exchange [✓ Identify/Set] Selected Station

Source	Destination	Protocol	Type	Info
08-00-06-6B-98-38	08-00-06-99-0A-A0	IP (NRT)	Write	Request, Write
00-02-8A-31-04-7B	01-0E-CF-00-00-00	RT	Identify/Set	Acyclic transfer low, Identify, Request
08-00-06-6B-98-38	00-02-8A-31-04-7B	RT	Identify/Set	Acyclic transfer low, Identify, Response Success
08-00-06-99-0A-A0	00-02-8A-31-04-7B	RT	Identify/Set	Acyclic transfer low, Identify, Response Success
08-00-06-93-D4-27	00-02-8A-31-04-7B	RT	Identify/Set	Acyclic transfer low, Identify, Response Success
08-00-06-99-0A-A0	08-00-06-6B-98-38	RT	Identify/Set	Acyclic transfer low, Set, Response Success
08-00-06-99-0A-A0	08-00-06-6B-98-38	RT	Alarm	Alarm low, Dest: 0x0002, Src: 0x0000, Type: Data, Alarm Notification,
08-00-06-99-0A-A0	08-00-06-6B-98-38	RT	Alarm	Alarm low, Dest: 0x0002, Src: 0x0000, Type: ERR, RTA error, PNIO
08-00-06-6B-98-38	08-00-06-99-0A-A0	RT	Alarm	Alarm low, Dest: 0x0000, Src: 0x0002, Type: ACK
08-00-06-6B-98-38	01-0E-CF-00-00-00	RT	Identify/Set	Acyclic transfer low, Identify, Request
08-00-06-99-0A-A0	08-00-06-6B-98-38	RT	Identify/Set	Acyclic transfer low, Identify, Response Success
08-00-06-6B-98-38	08-00-06-99-0A-A0	RT	Alarm	Alarm low, Dest: 0x0000, Src: 0x0002, Type: Data, Alarm Ack, Pull, N

Fig.13. Results of the test filtering frames.

Conclusions

The concept for an RTEFF IP core that is dedicated for RT network analyzers was presented. The authors have proven that the solution that is described may be used as the probe part of an RT-Ethernet network analyzer. A prototype IP Core was developed and verified in order to define the simplest possible setup with an Ethernet Frame Filter IP core. The solution that is presented was desired for its implementation on a Cyclone III FPGA using only the peripherals located on a DBC3C40 development board [15] with two functional Ethernet ports. This allowed for the execution of solution for parallel data filtration incoming from separate channels in a full-duplex environment. Many tests were performed simultaneously by using the parallel filtering structure. The test scenarios were changed by the software part of the network analyzer through which a new underlying FPGA-based hardware structure was generated. The prototype was used to perform tests on a PROFINET I/O RT-Ethernet based industrial network. The RTEFF IP Core for RT-Ethernet Monitoring that is presented allows for a reduction in the software effort and for a more efficient data analysis. Although the solution that is presented was implemented and tested in an Altera SOPC environment, it may also be applied in other FPGA environments.

Further studies will be performed using a development system equipped with three Ethernet connectors. The entire system will work in almost the same way. The third Ethernet plug will be connected at the monitoring station and the system will send all of the filtered frame to it immediately.

This will allow for the continuous filtration and analysis of the filtered frames. In the next phases, the authors plan to develop new features that will support the process of filter structure definition and will allow for a content aware analysis of RT- communication protocols.

Acknowledgment
This work was supported by the European Union from the FP7-PEOPLE-2013-IAPP AutoUniMo project "Automotive Production Engineering Unified Perspective based on Data Mining Methods and Virtual Factory Model" (grant agreement no: 612207) and by the European Social Fund (grant agreement no: UDA-POKL.04.01.01-00-106/09).

REFERENCES

[1]. Y. Ning, Z. Guo, S. Shen, B.Peng, Design of Data Acquisition and Storage System Based on the FPGA, Procedia Engineering, 29 (2012) pp. 2927-2930
[2]. Z. Luan, W. Zhang, Y. Zhang, Y. Lu, A new high-speed FPGA and Ethernet Based Embedded Data Acquisition System, IERI Procedia, 2 (2012), pp. 444-449
[3]. D. Palme Feasibility Study - PROFINET IO Device Integration into a FPGA. Haar, Germany : s.n., July 5, (2007).
[4]. X. Wang, Y. Lu, L. Zhang, Design and implementation of high-speed real-time data acquisition system based on FPGA, The Journal of China Universities of Posts and Telecommunications, 13 (2006), n.4, 61-66
[5]. P. Gaj, J. Jasperneite, and M. Felser. "Computer communication within industrial distributed environment - a survey," Industrial Informatics, IEEE Transactions on, vol. 9, no. 1, pp. 182–189, Feb 2013.
[6]. R. Cupek, M. Bregulla, and L. Huczala, PROFINET I/O network analyzer, Computer Networks in Springer-Verlag Precedings: Communications in Computer and Information Science, Vol 39, (2009) pp. 242-251,
[7]. F. Iwanitz, Net diagnosis for Profinet IO, Embedded Systems - Oldenbourg Industrieverlag Munchen, pp. 73-80, (2008)
[8]. G. Prytz, A Performance Analysis of EtherCAT and PROFINET IRT, Billingstadt, IEEE International Conference on Emerging Technologies and Factory Autiomation, (2008), pp. 408-415,
[9]. C. R. Clark, C. D. Ulmer, D. E. Schimmel, An FPGA-based network intrusion detection system with on-chip network interfaces, International Journal of Electronics, Vol. 93, No. 6, (2006) pp. 403–420,
[10]. Y. Wang, Implementation of digital filter by using FPGA, Karawara, Western Australia, Curtin University of Technology, (2005).
[11]. Song, Haoyu, et al. "Snort offloader: A reconfigurable hardware NIDS filter." Field Programmable Logic and Applications, 2005. International Conference on. IEEE, 2005.
[12]. netANALYZER http://www.hilscher.com/products_details_hardware.html?p_id=P_52ceb4af60a9b&bs=14
[13]. Prajapati, Gouri Shankar, and Nilay Khare. "A Framework of a Internet Firewall for IPv6 using FPGA." International Journal of Computer Applications 50 (2012).
[14]. Wicaksana, Arief, and Arif Sasongko. "Fast and reconfigurable packet classification engine in FPGA-based firewall." Electrical Engineering and Informatics (ICEEI), 2011 International Conference on. IEEE, 2011.
[15]. Devboards GmbH, DBC3C40 – Cyclone III Development Board, Lohmar: Devboards GmbH, (2008)RMII Consortium.
[16]. Altera Corporation, Nios II Processor Reference Handbook, San Jose: Altera Corporation, (2008)
[17]. Z. Navabi, Embedded Core design with FPGAs, McGraw-Hill, (2007)
[18]. R. Munden, Richard. AISIC and FPGA Verification: A Guide to Component Modeling. San Francisco : Morgan Kaufmann Publishers, (2005). ISBN: 0-12-510581-9.
[19]. D. Pellerin, St. Thibault, Practical FPGA Programming in C. s.l.: Prentice Hall PTR, (2005). ISBN: 0-13-154318-0.
[20]. R. Bodenner, S. Thibault Application Note - Using DMA for Data Communications on Altera Nios II Platforms. Kirkland, Washington, United States of America : Impulse Accelerated Technologies, Inc., October 25, (2004).
[21]. P. Gaj, B. Kwiecień. 'Useful efficiency in cyclic transactions of Profinet IO' Studia Informatica, Gliwice 2010, PL ISSN 0208-7286
[22]. RMII Specification. s.l.: RMII Consortium, 1998.
[23]. H.M. Frazier, Media independent interface: concepts and guidelines. Jr. San Francisco, CA, USA: Sun Microsyst. Comput. Co., 1995. ISBN: 0-7803-2636-9

***Authors**: dr inż. Rafał Cupek, Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, E-mail: rafal.cupelk@polsl.pl; mgr inż. Łukasz Huczała, Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, E-mail: lukasz.huczala@polsl.pl; mgr inż. Piotr Piękoś ASML B.V., Veldhoven, The Netherlands E-mail: piotr.piekos@asml.com; dr inż. Adam Ziębiński, Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, E-mail: adam.ziebinski@polsl.pl;*