A Highly Configurable Packet Sniffer Based on FPGA for Network Security Applications

Marco Grossi , Fabrizio Alfonsi , Marco Prandini , Alessandro Gabrielli *

Posted Date: 13 September 2023

doi: 10.20944/preprints202309.0884.v1

Keywords: network security; packet sniffer; packet classification; FPGA; embedded systems.

Article

A Highly Configurable Packet Sniffer Based on FPGA for Network Security Applications

Marco Grossi [1,4], Fabrizio Alfonsi [1,3], Marco Prandini [2] and Alessandro Gabrielli [1,3] *

[1] Department of Physics and Astronomy "Augusto Righi" (DIFA), Alma Mater Studiorum—Università di Bologna, 40126 Bologna, Italy; marco.grossi8@unibo.it (M.G.); fabrizio.alfonsi@bo.infn.it (F.A.)
[2] Department of Computer Science and Engineering - Università di Bologna, 40126 Bologna, Italy ; marco.prandini@unibo.it (M.P.)
[3] INFN Bologna, 40127 Bologna, Italy
[4] INFN CNAF, 40127 Bologna, Italy
* Correspondence: alessandro.gabrielli@unibo.it; Tel.: +39-051-209-5052

Abstract: In recent years web applications and on-line business transactions have grown many folds. Consequently, also cyberattacks have increased and represent a serious threat to the pervasive digital services upon which our society relies. To mitigate cyberattacks, many countermeasures are deployed on computing nodes (e.g., anti-malware software) as well as on network devices to detect and possibly block malicious packets in transit; these monitoring devices broadly go under the name of firewalls. Firewalls are designed according to two main architectural approaches: software running on a standard or embedded computer, or purposedly designed hardware, e.g., ASICs. Software-based solutions have the advantage of high flexibility and can be ported on easily upgradable hardware. However, hardware implementation represents the only viable solution for high data rates. On the market, very fast devices of the latter kind are available, but their cost is typically very high, especially considering that their ultra-optimized design makes updating them very difficult, with the consequence of a rather short lifespan. As a more balanced alternative, we wanted to investigate the use of an FPGA architecture, , which is significantly easier to update than custom-built chips, and features low-latency and high-throughput characteristics concurrently, making it preferrable to other programmable systems based on GPUs or microcontrollers. In this paper a packet sniffer that has been designed on FPGA with a 1 Gbit/s data transfer rate is presented. The system is implemented on the FPGA development board KC705 by Xilinx, can analyze Ethernet frames, checking the frame fields against a set of rules defined by the user and calculates statistics of the received Ethernet frames over time. The designed packet sniffer has been successfully tested both with Ethernet frames ad hoc generated using a packets generator, and with real web traffic by connecting the packet sniffer to the internet.

Keywords: network security; packet sniffer; packet classification; FPGA; embedded systems

1. Introduction

The interconnection of embedded technologies in the paradigm of the Internet of Things (IoT) and the increasing complexity of networks due to the continuous increase of web applications and business transactions, make networks more vulnerable to cyberattacks that can result in denial of service, unauthorized accesses and sensitive data theft or alteration [1]. Cyberattacks represent a threat in different fields of application, such as Industry 4.0 [2–4] and healthcare [5–7].

In the context of computer networks, cyberattacks are often mitigated using network security systems, such as firewalls and packet sniffers. Both kinds of devices try to detect malicious traffic, either by finding it unrelatable with known legitimate packets (anomaly-based approach) or by comparing packets with samples of known attacks (signature-based approach). In this work, we designed our prototype according to the signature-based approach, not excluding the possibility of

improving the design to encompass the anomaly-based one. The difference between a firewall (or Intrusion Prevention System, IPS) and a packet sniffer (or Intrusion Detection System, IDS) is that an IPS has -- either physically or logically -- two ports. When network data enter one port, these are transparently transferred to the output of the other port if no threats are detected or blocked otherwise [8]. An IDS, on the other hand, is a system that works in passive mode, reading all data received at its input port, and sending alarm messages to a remote server when potential threats are detected [9,10]. It can be placed in-line, receiving traffic on an entry port but always forwarding packets to the exit port, or on the side of the network, by configuring a switch to mirror all traffic from monitored ports to the IDS itself, which in this configuration does not even need an exit port. Firewall and packet sniffers on home or business networks, i.e. running at speeds that rarely exceed 1Gbps, can be effectively implemented in software running on a server. Nivedita and Kumar in 2017 proposed an innovative approach for a firewall using a hybrid frame of Netfilter for Linux web server [11]. Nivethan and Papa in 2016 presented a novel methodology that extends existing Linux-based firewalls for use in systems that use DNP3 protocol, in particular to protect US smart grid [12]. Tirumala et al. in 2022 investigated the hardware capability of network interfaces of Raspberry Pi to handle high volumes of incoming traffic in the protection of small-to-medium enterprises (SMEs) and smart homes [13]. Phalguni and Krishna in 2015 presented a software firewall for the application layer running on a ARM9 based single board computer based on the Iptables/Netfilter frame in Linux [14]. Oluwabukola et al. in 2013 proposed Psniffer, a packet sniffer software application for network security in Java [15]. Phang et al. in 2008 presented V6SNIFF, an efficient packet sniffer capable to analyze Ipv6 packets [16]. Goyal et al. in 2017 made a comparative study between the two most popular packet sniffing software (Tcpdump and Wireshark) [17].

Firewalls and packet sniffers based on software running on a standard computer work reliably in most situations. However, when the amount of transferred data and data transfer speed increase over a certain threshold, these systems can lose effectiveness. In these situations, a hardware implementation is preferred since this can guarantee real time operations and much higher data throughput. Niemiec et al. in 2019 presented a survey about the open research challenges that must be addressed for the adoption of FPGAs in the acceleration of virtualized network functions [18]. Wicaksana and Sasongko in 2011 presented a prototype of a hardware stateless firewall designed using Cyclone II FPGA working at 91 MHz [19]. According to the authors, however, only the packet classification was implemented, and the lack of an efficient FIFO buffer prevents high speed data transfer. Fiessler et al. in 2016 proposed HyPaFilter, a hybrid packet classification approach where simple operations are handled in hardware designed with FPGA while complex operations are handled by a Linux based software firewall [20]. Ezzati et al. in 2010 proposed a packet classification engine based on artificial neural networks that achieves a 97% accuracy in the classification of TCP/IP packets [21]. According to the authors, the proposed method is appropriate for FPGA implementation, Matlab simulation were carried out and the VHDL code was simulated and synthetized, but an operative system was not developed.

In this paper a hardware packet sniffer implemented in FPGA is presented. The system is designed using Verilog with a commercial development board: the Xilinx KC705. The system is highly configurable, that is the rules to discriminate safe and potentially dangerous Ethernet frames can be set by the user with an ad-hoc software designed in LabVIEW (National Instruments). The system also provides a set of statistics on the analyzed network traffic. In Section 2 the format of an Ethernet frame and the most important fields for the determination of potential threats are presented. In Section 3 the developed packet sniffer is presented with details of the involved signals and simulations of the system behavior. In Section 4 the results of several tests carried out with the designed packet sniffer are presented and compared with the results obtained with Wireshark, a popular packet sniffing software. In Section 5 a short discussion is presented to the extension of this project to higher data rates. Finally, conclusions are presented in Section 6.

2. Ethernet frame format

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

Ethernet is a networking technology widely used in local area networks (LANs), metropolitan area networks (MANs) and wide area networks (WANs). It is also used in other fields of application, ranging from industry to avionics, telecommunication, and multimedia. It has been introduced in 1980 and the first standardization was in 1983 (IEEE 802.3). Ethernet data transfer rate evolved from the initial 2.94 Mbps up to 100 Gbps [22].

In Ethernet networking, data are described according to different levels of abstraction of the OSI model [23]. At level 2 (data link layer) of the OSI model data are described in the form of frames. The format of an Ethernet II frame is shown in Figure 1. It always starts with a preamble and a start frame delimiter (SFD) and ends with a frame checksum (FCS), a four bytes CRC, that allows to detect corrupted data in the frame. The header of the Ethernet frame consists of the following fields: destination and source media access control (MAC) address, VLAN tag (an optional field) and the protocol type of level 3 (network layer), while the payload represents the data of the network layer. Two of the most important protocols of the network layer are ARP and IP, whose packet format is presented in Figures 2 and 3, respectively. The IP header also contains the protocol of the level 4 layer (transport layer) whose data are present in the IP payload. In the case of the transport layer, three different protocols have been considered (TCP, UDP and ICMP) whose packet format is presented in Figures 4, 5 and 6, respectively.

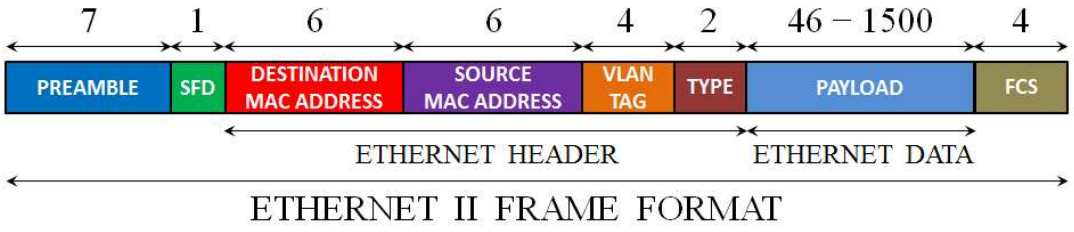


Figure 1. The format of an Ethernet II frame. The length of each field of the frame is reported as number of bytes.

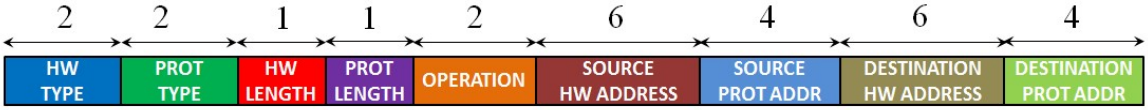


Figure 2. The format of an ARP packet. The length of each field of the frame is reported as number of bytes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

Figure 3. The format of an IP packet. The different fields of the packet are presented in rows of 32 bits.

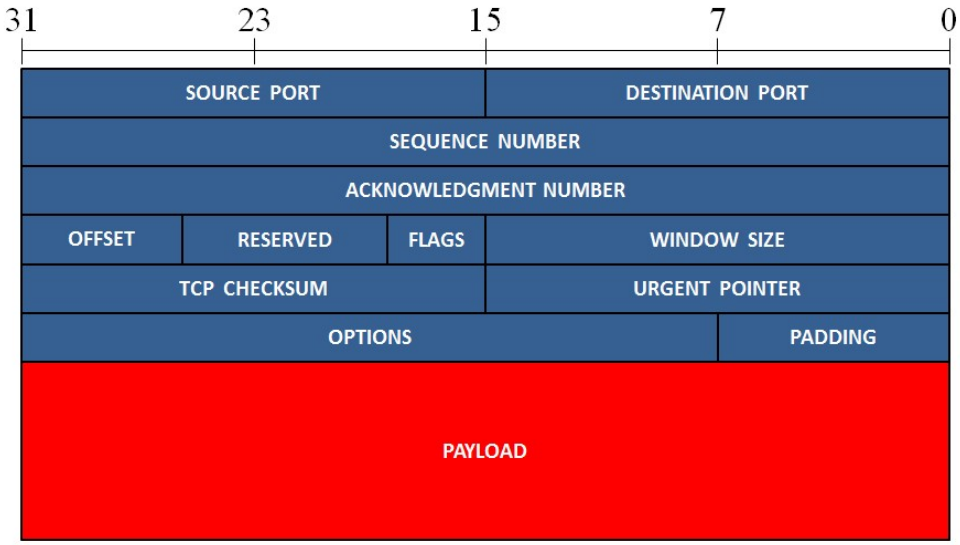


Figure 4. The format of a TCP packet. The different fields of the packet are presented in rows of 32 bits.

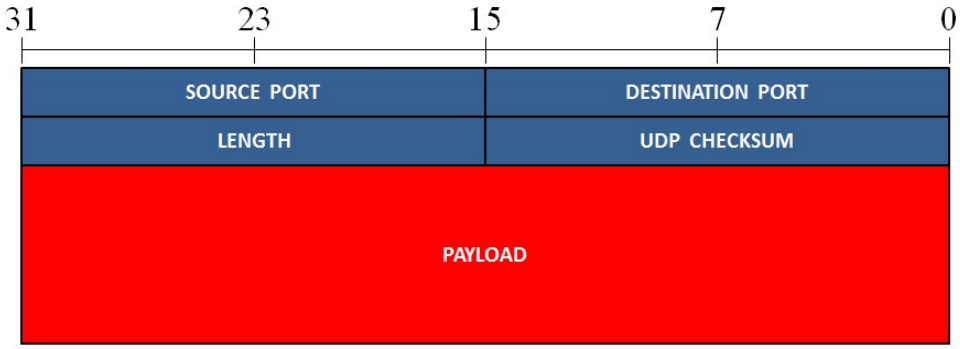


Figure 5. The format of an UDP packet. The different fields of the packet are presented in rows of 32 bits.

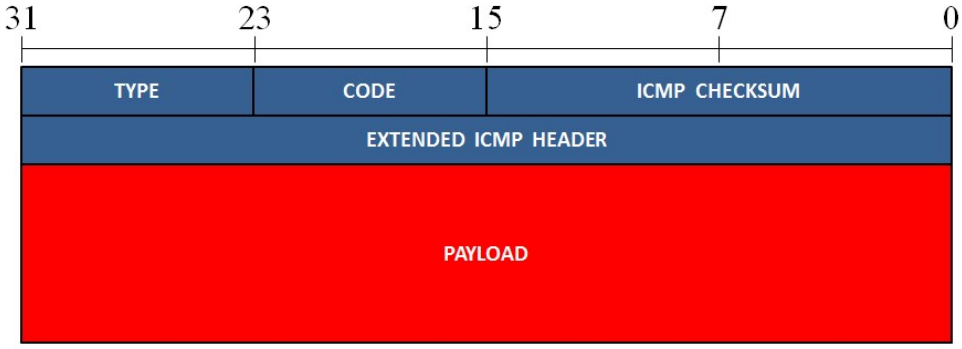


Figure 6. The format of an ICMP packet. The different fields of the packet are presented in rows of 32 bits.

In the case of TCP-IP and UDP-IP packets, the most important fields from the cybersecurity point of view are IP source and destination addresses as well as the source and destination ports. IP address is used, along with MAC address, to identify a device inside a network. IP address is used as a global address while MAC address is used as local address. Data on internet are directed using the IP address and, when the local area network (LAN) is reached, IP address is translated to MAC address

 doi:10.20944/preprints202309.0884.v1

and data are delivered to the correct device. Similarly, source and destination port numbers are used to define the application or service involved, so that the operating system can deliver the packet to the appropriate process. The relevance of these parameters for cybersecurity is twofold. Being able to read them enables a potential attacker to infer who is communicating with whom (in terms of network hosts) and what kind of dialogue is happening, even if the contents (payload) are encrypted. Obscuring IPs and ports is possible only with modified protocols, such as IPSec [24], or by encapsulation performed by specific applications (Virtual Private Networks, VPN); moreover, it would interfere with the functionality of a firewall or benign packet sniffer. For these reasons, we do not consider the scenario of encrypted IP and port numbers. An attacker that can alter these parameters can hide their own identity by spoofing the source address, usually with the goal of bypassing firewall rules that would block packets bearing their real address.

3. Experimental design

The experimental setup designed to validate the developed packet sniffer is composed of two different parts: a packets generator, to generate Ethernet frames with variable data rate selectable by the user, and the packet sniffer that analyses the Ethernet traffic. The analysis includes Ethernet data statistics based on the fields of the analysed Ethernet frames.

Both the packets generator and the packet sniffer are designed using the FPGA development board KC705 by Xilinx [25]. The KC705 development board provides a hardware environment for developing and evaluating designs targeting the Kintex-7 XC7K325T-2FFG900C FPGA. It integrates, among other things, 1GB of DDR3 RAM, 128 MB of flash memory, a 200 MHz LVDS oscillator, PCI express connectivity, USB ports for FPGA programming and UART data communication, a 10/100/1000 tri-speed Ethernet PHY (Marvell M88E1111-BAB1C000), pushbuttons, switches, and LEDs for user interaction.

The physical layer of the Ethernet protocol is managed by the tri-speed Ethernet PHY chip integrated on the KC705 board, while the data link layer is managed by the Tri-Mode Ethernet Media Access Controller (TEMAC), an intellectual property module provided by Xilinx [26]. Ethernet traffic generation and analysis have been implemented for a data rate of 1 Gbit/s. Ethernet frame analysis is carried out for the following protocols: ARP, IPv4, UDP, TCP, ICMP. The maximum Ethernet frame size is set to 1518 bytes for normal frames and 1522 bytes for VLAN frames.

The implementation of the packets generator and packet sniffer is discussed in detail in Section 3.1 and 3.2, respectively.

3.1. Packets generator

A KC705 development board is used to generate Ethernet frames that are sent to the packet sniffer to test its functionality. The main functional blocks that compose the packets generator are presented in Figure 7, where all the interconnections among the blocks are also shown.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

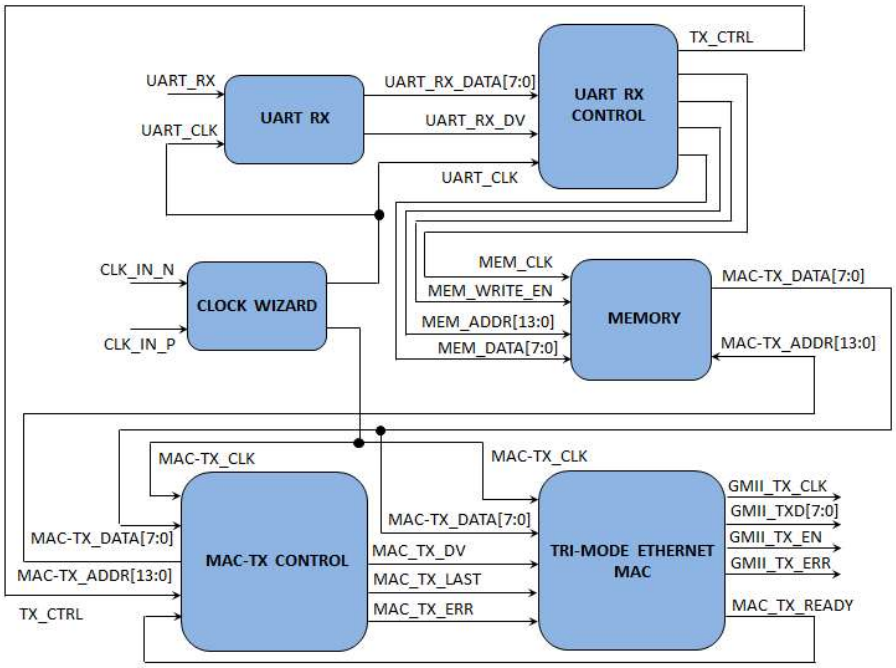


Figure 7. Schematic of the packets generator and its main functional blocks.

The packets generator features a 200 MHz differential clock available on the KC705 board (CLK_IN_N and CLK_IN_P) and the serial UART receiving port (UART_RX). The 200 MHz differential clock is used to generate two different clock signals: UART_CLK (10 MHz, used to provide the clock signal for the UART_RX module and the UART_RX_CONTROL module) and MAC_TX_CLK (125 MHz, used to provide the clock signal for the MAC-TX_CONTROL module and the TRI-MODE_ETHERNET_MAC module). The UART_RX module is used to read the serial input UART_RX and generate a 8-bit output byte (UART_RX_DATA) that is sampled when the signal UART_RX_DV is active (i.e. logic value 1). The UART_RX_CONTROL module reads the received 8-bit data from UART and controls the operation of the system. In Figure 8 the waveforms of the signals are presented in the case of frames data loading in the memory in the case of a set of two Ethernet frames (each of length 60 bytes), one encapsulated with UDP data and the other with TCP data. The operation begins when the UART_RX_CONTROL module receives the control byte 0x14 (decimal value 20), followed by two bytes indicating the number of frames that are loaded (0x0002), the total number of bytes that follows (0x007c, decimal number 124) and, for each frame the data preceded by two bytes indicating the frame length (0x003c, decimal value 60). When received, data are stored in a 16kbytes distributed RAM (MEMORY module) with synchronous write and asynchronous read. Figure 9 shows the waveform signals at the end of frames data set loading.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

Figure 8. Signal waveforms in the case of data loading in memory using UART (start of the operation).

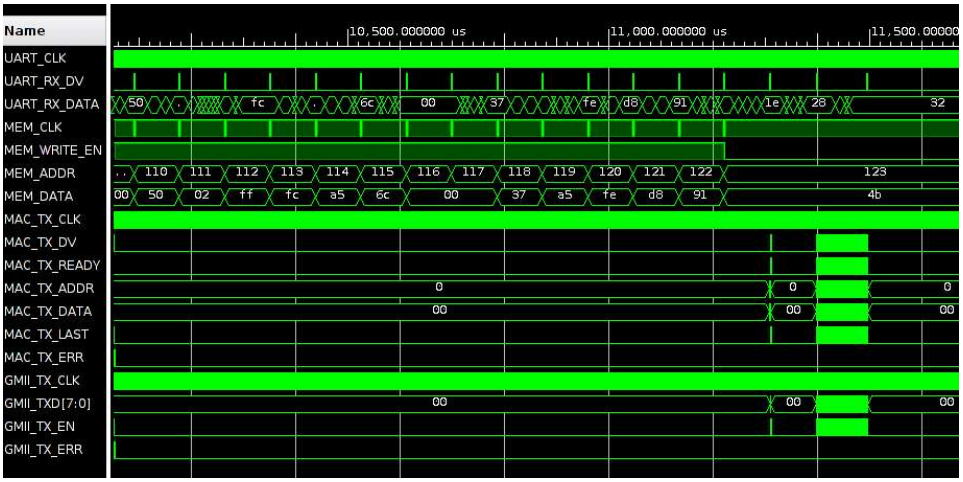


Figure 9. Signal waveforms in the case of data loading in memory using UART (end of the operation).

Once the frames dataset is loaded in memory, different operations can be performed by sending the correct control byte to the UART_RX_CONTROL module: start of a single write operation (control byte 0x1e, decimal value 30, to send out on the Ethernet port the frames in the data set), start of a continuous write operation (control byte 0x28, decimal value 40, to send out on the Ethernet port the frames in the data set in looping mode), stop of the continuous write operation (control byte 0x32, decimal value 50). The system can be programmed to insert a defined delay between the end of a frame and the beginning of the next one, to provide a controllable data rate for Ethernet transmission.

Figure 10 shows the waveform signals in the case of a single write operation, where the two Ethernet frames in the data set are read from memory by the MAC-TX_CONTROL module. Then the data are sent to the input of the TEMAC module that generates the output data for the Ethernet PHY chip on the KC705 board. Figures 11 and 12 show the signal waveforms of first frame transmission. The data valid signal (MAC_TX_DV) is enabled during the transmission operation and data read from memory are sampled when the signal MAC_RX_READY is active. Initially, the length of the frame in bytes is read from memory (0x003c, 60 bytes), then MAC_TX_DV is enabled, and the frame data are transmitted. During the transmission of the last byte of the frame the signal MAC_TX_LAST is enabled, then MAC_TX_DV is disabled, and, after a delay, the second frame is transmitted. At the output of the TEMAC module, the frame data (GMII_TXD) are sent to the Ethernet PHY chip preceded by the frame preamble (0x55555555555555d5) and followed by the frame checksum (0x8a68866a). During the transmission the signal GMII_TX_EN is enabled.

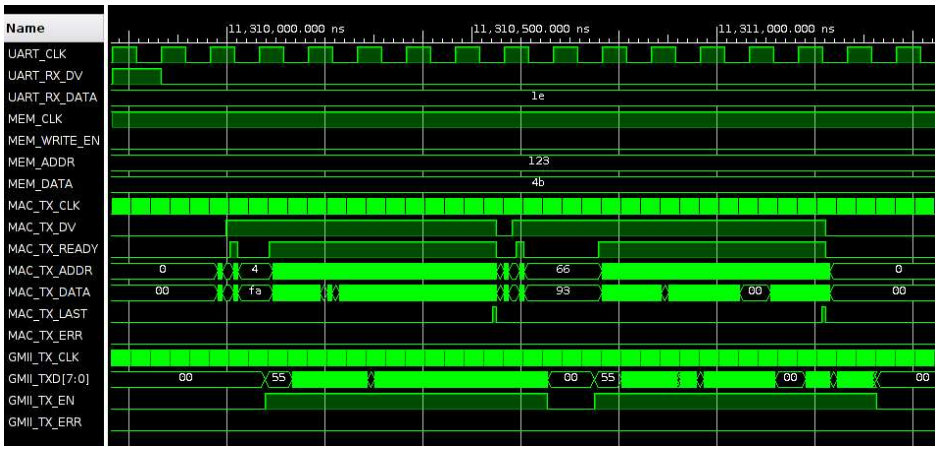


Figure 10. Signal waveforms in the case of the generation of two Ethernet frames in sequence.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

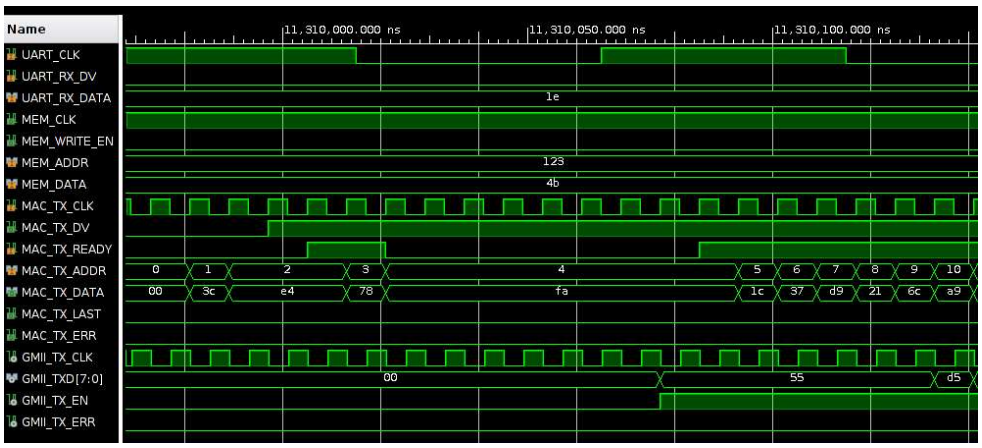


Figure 11. Signal waveforms in the case of the generation of an Ethernet frame (start of the operation).

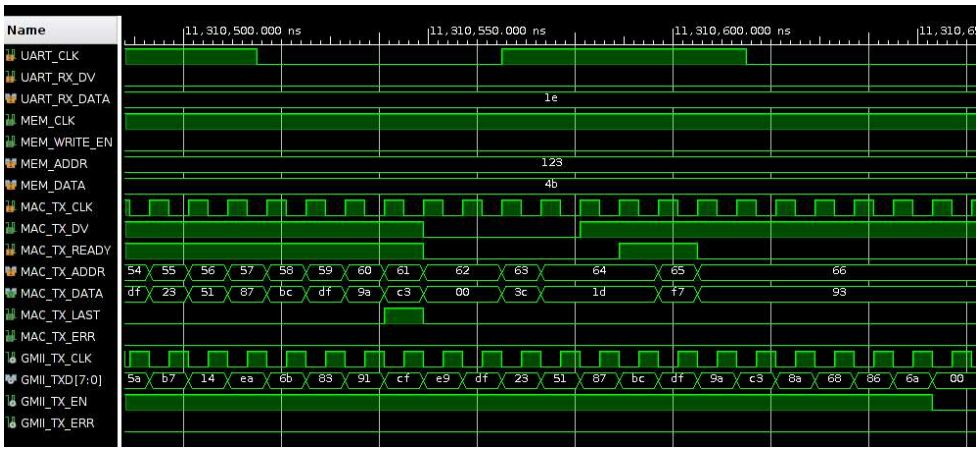


Figure 12. Signal waveforms in the case of the generation of an Ethernet frame (end of the operation).

Programs have been written in LabVIEW to control the packets generator. Test have been carried out by connecting the packets generator Ethernet port to a PC Ethernet port and generating different sets of Ethernet frames. The network traffic on the PC has been analysed by Wireshark to check the correct operation of the packet generator.

3.2. Packet sniffer

Figure 13 shows the main functional blocks that compose the packet sniffer, where all the interconnections among the blocks are also shown.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

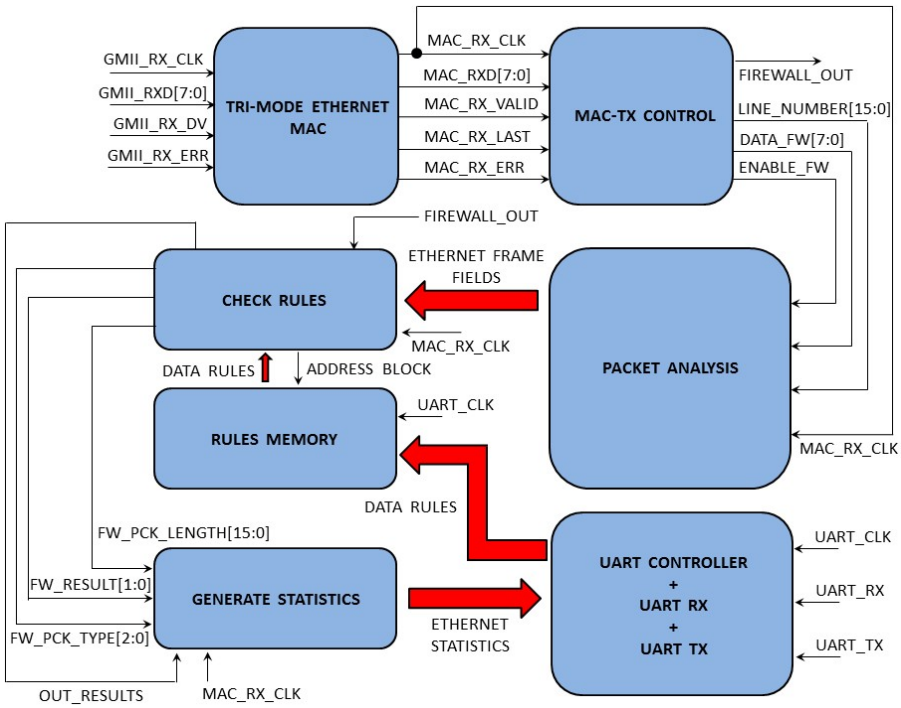


Figure 13. Schematic of the packet sniffer and its main functional blocks.

Two different clocks are used in the design: MAC_RX_CLK (125 MHz), used for synchronization of the packet reception and analysis and UART_CLK (10 MHz), used for synchronization of the data transmission/reception between the packet sniffer and a PC using UART at a baud rate of 9600.

The Ethernet frames are received by the Marvell M88E1111-BAB1C000 Ethernet PHY chip available on the KC705 development board and transmitted in parallel form (1 byte) to the TEMAC IP by Xilinx that removes the frame preamble and checks the correctness of the frame checking sequence (FCS). Then the frame data are presented to the MAC_TX_CONTROL module that feeds the data to the PACKET_ANALYSIS module that calculates the frame fields (as discussed in Section 2). Figure 14 shows the signal waveforms for the frame reception and analysis in the case of four different Ethernet frames (UDP, TCP, ARP and ICMP) that are continuously received.

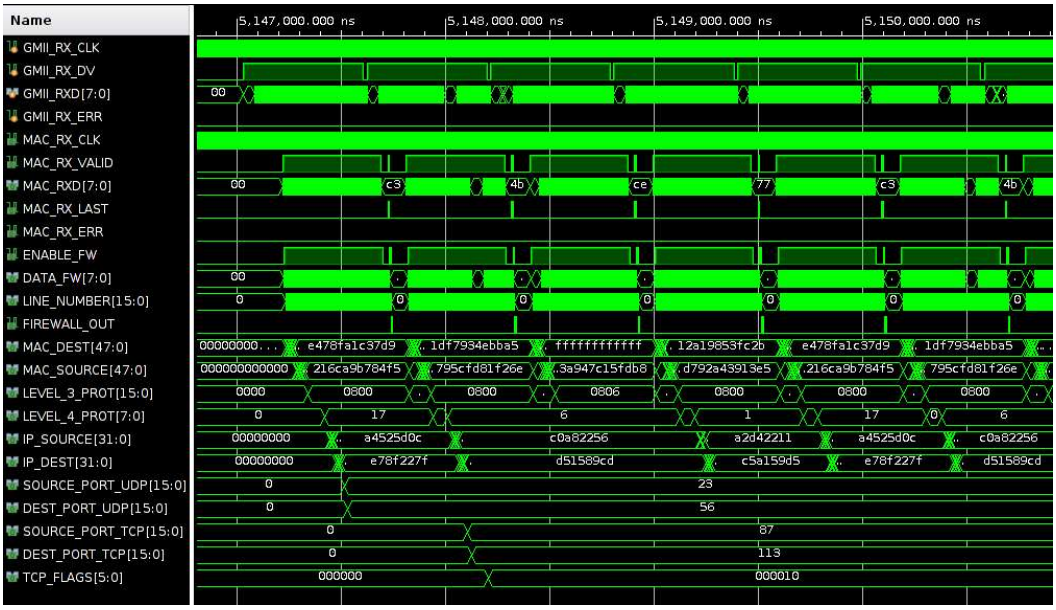


Figure 14. Signal waveforms for the reception and analysis of Ethernet frames.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

When the packet is analysed, the signal ENABLE_FW is activated and when the analysis is completed a pulse is generated on the signal FIREWALL_OUT. The frame analysis calculates different parameters of the frame such as: MAC source and destination addresses (MAC_SOURCE and MAC_DEST), the protocol of level 3 (LEV_3_PROT) and, in the case of LEV_3_PROT = 0x0800 (IP packet), the protocol of level 4 (LEVEL_4_PROT), the IP source and destination addresses (IP_SOURCE and IP_DEST), the source and destination ports in the case of UDP and TCP packets and the flags in the case of TCP packets. When the frame analysis is completed and the signal FIREWALL_OUT is pulsed, the CHECK_RULES module decides if the packet matches the firewall rules. Firewall rules are stored in the RULES_MEMORY module and are loaded from the PC using UART. The firewall rules are defined for an IP packet according to a whitelist strategy based on the range of IP source and destination addresses, the protocol of level 4 and the range of source and destination ports. Based on these rules, the CHECK_RULES module outputs the type of packet (FW_PCK_TYPE), the length in bytes of the packet (FW_PCK_LENGTH), and the results of the analysis (FW_RESULT, 0 packet with errors, 1 packet with rules violation, 3 allowed packet). When the rules check is completed, the signal OUT_RESULTS is pulsed and the module GENERATE_STATISTICS calculates a statistic (packet type, packet length, packet allowed or rejected, total data transferred, etc.) that can be downloaded to the PC using UART.

Figure 15 shows the signal waveforms for the frame analysis and statistics generation in the case of four different Ethernet frames (UDP, TCP, ARP and ICMP) that are continuously received. Initially, the rules have been defined to accept only the IP source address of the TCP packet. It is planned to extend the set of rules in the near futures according to the applications. Currently we plan to test and use this firewall hardware based on FPGA to monitor the I/O internet traffic crossing our department at the University of Bologna.

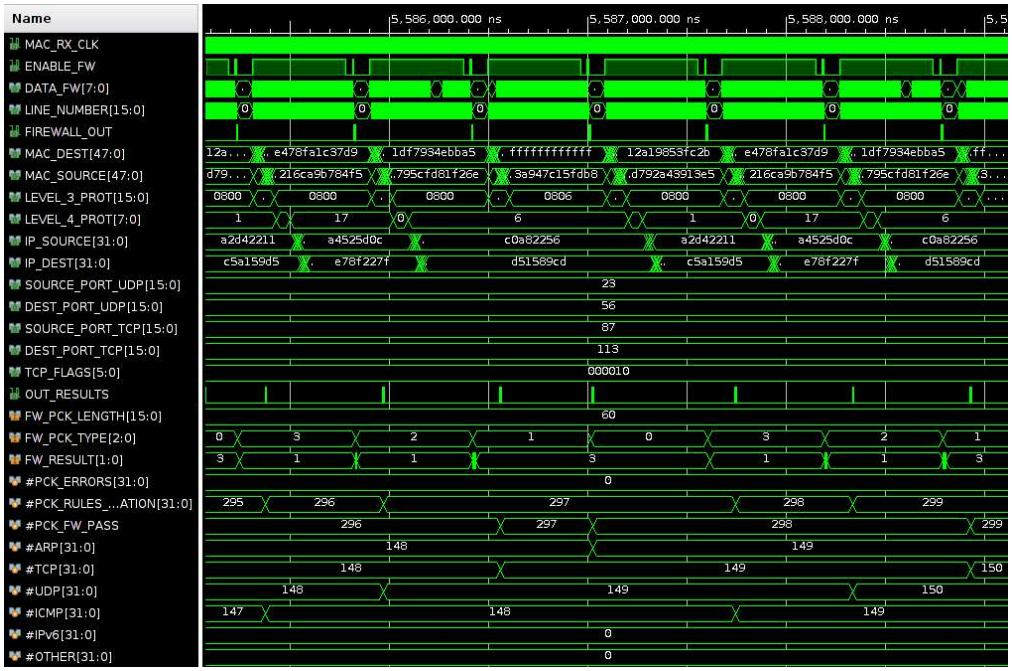


Figure 15. Signal waveforms for the frame analysis and statistics generation of Ethernet frames.

As can be seen in Figure 15, among the four types of packets that are generated (UDP, TCP, ARP and ICMP), TCP (FW_PCK_TYPE = 1) and ARP (FW_PCK_TYPE = 0) are tagged as allowed (FW_RESULT = 3). In fact, in the TCP packet the IP source address matches the firewall rules and ARP is not an IP packet (LEV_3_PROT = 0x0806). On the contrary the IP packets (LEV_3_PROT = 0x0800) with a level 4 UDP protocol (LEV_4_PROT = 17) and ICMP protocol (LEV_4_PROT = 1) are tagged as rejected since the IP source address does not match the firewall rules.

4. Results

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

The packet sniffer presented in Section 3.2 has been tested to evaluate its performance. Initially, it has been tested by generating the Ethernet frames using the packets generator described in Section 3.1. These results are presented in Section 4.1. Next, the packet sniffer has been connected to the internet to test its reliability and performance under real Ethernet traffic and the corresponding results are presented in Section 4.2.

4.1. Tests with the packet generator

The packet sniffer has been tested by generating Ethernet frames with the packet generator. The measurement setup is depicted in Figure 16.

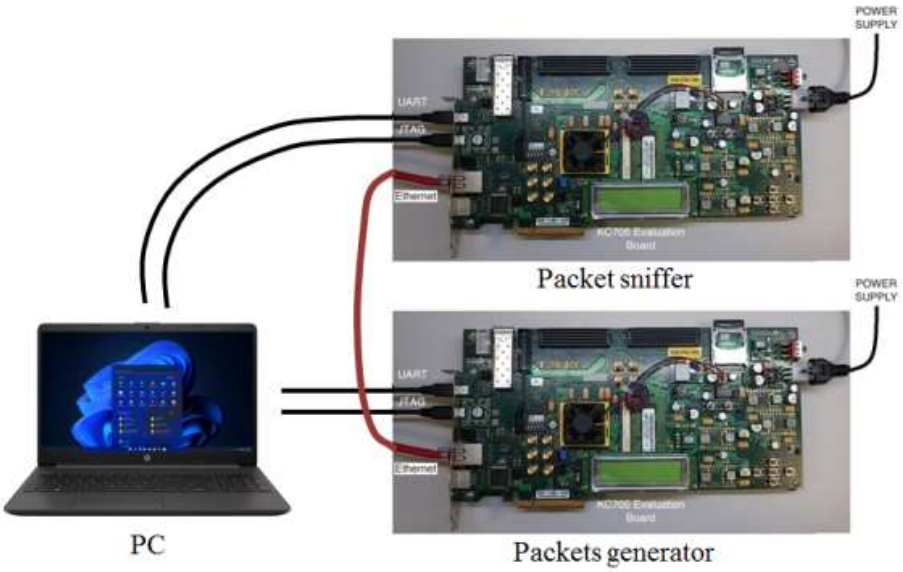


Figure 16. Experimental setup where the Ethernet cables of the packets generator and the packet sniffer are connected together.

Both the packet generator and the packet sniffer are interfaced to the PC UART port. The PC loads the selected Ethernet frames on the packet generator, sets the delay between frames and starts a continuous transfer to the packet sniffer. Once per second, the packet sniffer transfers the traffic statistics to the PC using the UART port. Ethernet frames of different lengths and different inter-frames delays have been tested. The results have shown that the packet sniffer operates correctly and the Ethernet traffic in terms of frame types and speed of data transfer is what expected.

Considering that the packet sniffer reads a single byte in a clock cycle of duration 8ns (T_{CLK}), corresponding to a 125 MHz clock, the number of clock cycles N_{CLK} needed to analyse an Ethernet frame of byte length N_{ETH} is given by:

$$N_{\mathrm{CLK}} = N_{\mathrm{PREAMBLE}} + N_{\mathrm{SFD}} + N_{\mathrm{ETH}} + N_{\mathrm{FCS}} + N_{\mathrm{MAC}} + N_{\mathrm{DELAY}} \quad (1)$$

where $N_{PREAMBLE}$ is the number of clock cycles needed to read the Ethernet frame preamble (7), N_{SFD} is the number of clock cycles needed to read the Ethernet start frame delimiter (1), N_{FCS} is the number of clock cycles needed to read the Ethernet frame checksum (4), N_{MAC} is the number of clock cycles of delay introduced by the TEMAC module, and N_{DELAY} is the number of clock cycles of delay set in the packets generator using the UART command. This results in :

$$N_{\mathrm{CLK}} = N_{\mathrm{ETH}} + N_{\mathrm{MAC}} + N_{\mathrm{DELAY}} + 12 \quad (2)$$

Thus, the data rate (DR) expressed in bit/s can be calculated as:

$$\mathrm{DR} = \frac{8 \times N_{\mathrm{ETH}}}{N_{\mathrm{CLK}} \times T_{\mathrm{CLK}}} = 10^9 \frac{N_{\mathrm{ETH}}}{N_{\mathrm{ETH}} + N_{\mathrm{MAC}} + N_{\mathrm{DELAY}} + 12} \quad (3)$$

From equation (3) it is evident that, even for $N_{DELAY} = 0$, the data rate cannot reach the maximum value of 1 Gbit/s, but this can be reached asymptotically for $N_{ETH} \rightarrow +\infty$. Different Ethernet frames of different byte lengths and different protocols have been tested with the packet generator working in continuous mode: ARP (42 bytes), TCP (54 bytes), TCP (74 bytes), UDP (112 bytes), UDP (256 bytes),

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

TCP (506 bytes), TCP (983 bytes), TCP (1414 bytes). For each Ethernet frame, different inter-frame delays have been tested: 0, 10, 50, 100, 500, 1000, 5000, 10000 and 50000 clock cycles delay. The results of such test are reported in Figure 17.

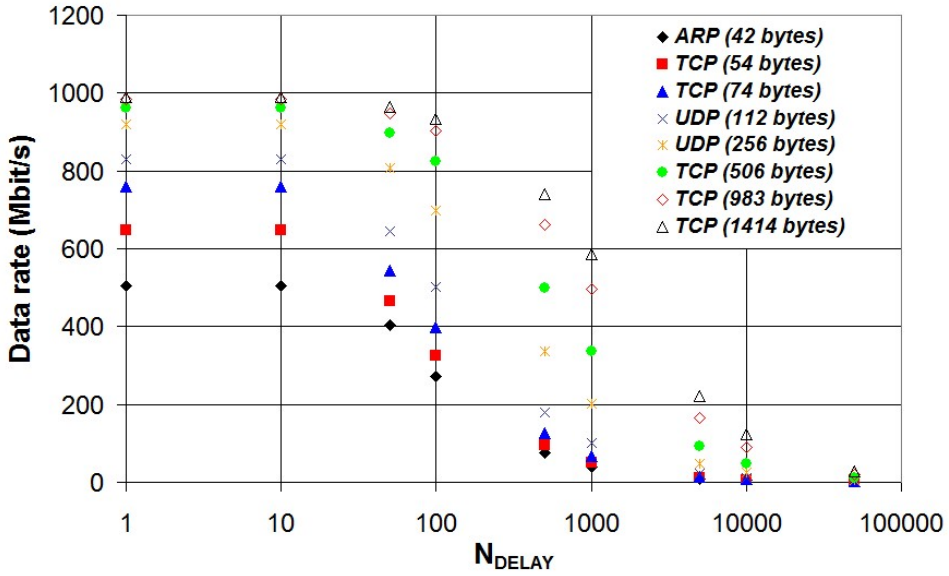


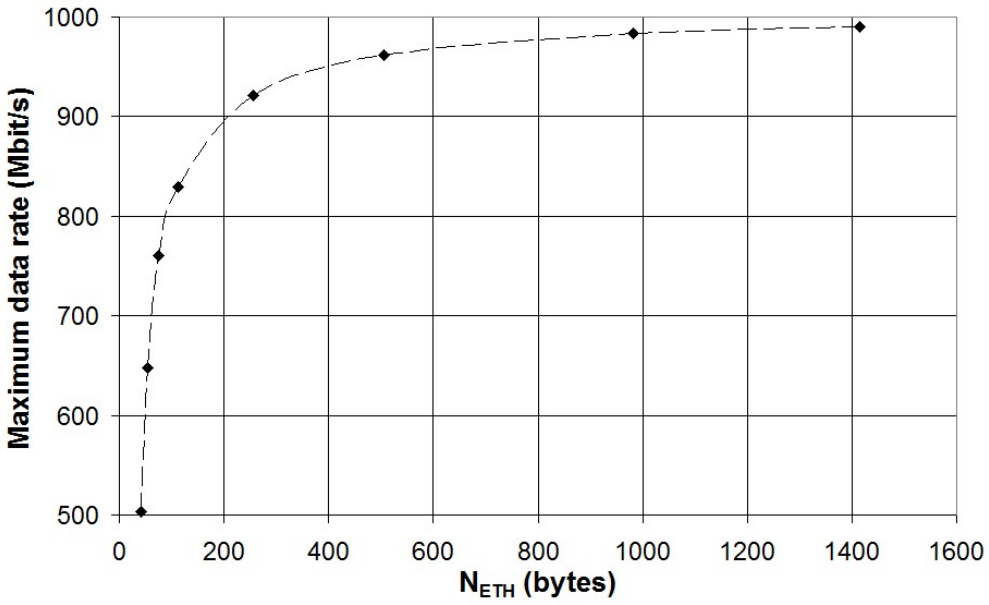
Figure 17. Data rate as function of inter-frame delay for Ethernet frames of different byte lengths and different protocols.

As expected, the data rate decreases with the inter-frame delay. Data reported in Figure 17 have been fitted to equation (3) using the software CurveExpert Professional v2.4.0 (Hyams Development) and the results are characterized by an high correlation with a coefficient of determination (R^2) ranging from 0.989 to 0.999.

The maximum value of the data rate (DR_{MAX}) is obtained in the case of no inter-frame delay ($N_{DELAY} = 0$) and can be expressed as:

$$\mathrm{DR_{MAX}} = 10^9 \frac{N_{\mathrm{ETH}}}{N_{\mathrm{ETH}}+N_{\mathrm{MAC}}+12} \tag{4}$$

In Figure 18 the maximum data rate is plotted vs the frame length in bytes (N_{ETH}). Measured data have been fitted to equation (4) and this resulted in high correlation ($R^2 > 0.98$) and an estimated value of N_{MAC} of about 16 clock cycles.



Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

Figure 18. Maximum data rate as function of the Ethernet frame length (N_{ETH}).

Then, the performance of the proposed packet sniffer has been evaluated and compared with a PC running Wireshark. Figure 19 shows the experimental setup. A switch Summit X450e-48p by Extreme Networks has been used to provide the same Ethernet traffic for the packet sniffer and the PC running Wireshark. The Ethernet port of the PC is connected to port 2 of the switch and the traffic on port 2 is mirrored to port 10 where the Ethernet port of the packet sniffer is connected. The packets generator is connected to port 6 of the switch and generates an Ethernet frame addressed to the PC. Such Ethernet frame is continuously transmitted with different values of the inter-frame delay. The results of such test are presented in Figure 20 where the frame rate is plotted vs time for both the packet sniffer and the PC running Wireshark. The produced plots correlate as expected but the response of the PC running Wireshark is delayed of about 8-11 seconds if compared to the packet sniffer. Moreover, when the frames rate is higher than 48000 frames/s, the PC running Wireshark begins to lose data and it eventually crashes for frame rates higher than 100000 frames/s. This shows how the proposed packet sniffer can monitor Ethernet traffic with data rate up to 1 Gbit/s with accurate results, while this is not the case of a PC running Wireshark that cannot work properly if the data rate is too high.

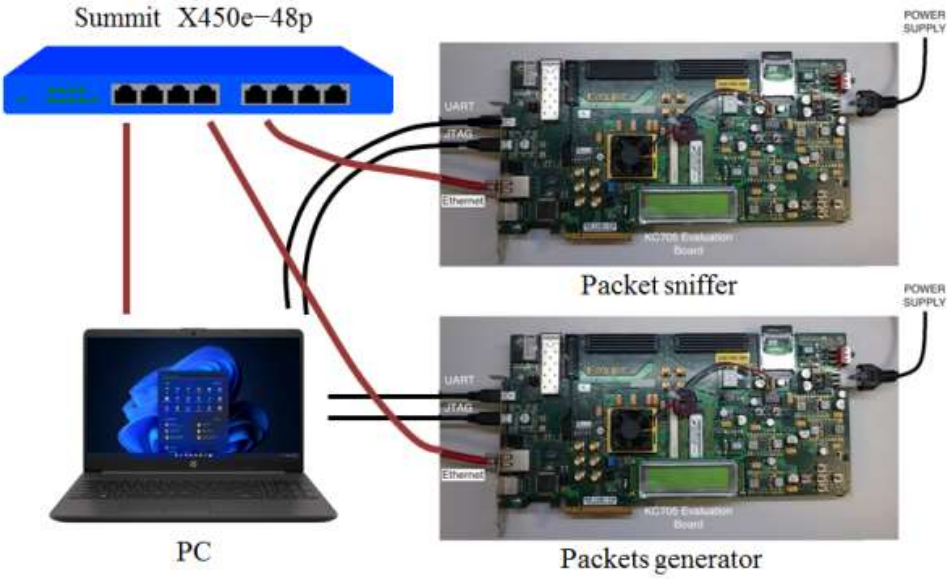


Figure 19. Experimental setup where the PC Ethernet traffic is mirrored on the packet sniffer and the packets generator is used to generate the traffic.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

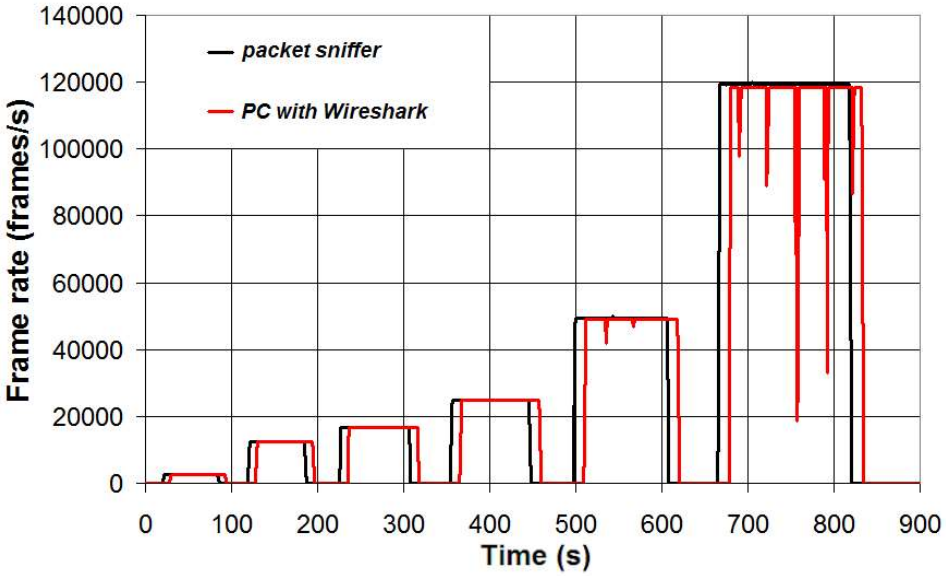


Figure 20. Measured frame rate vs time in the case of the proposed packet sniffer and a PC running Wireshark.

4.2. Tests with real Ethernet traffic

After the designed packet sniffer has been tested with controlled Ethernet traffic generated with the packets generator, the system has been tested with real Ethernet traffic, using the experimental setup shown in Figure 21.

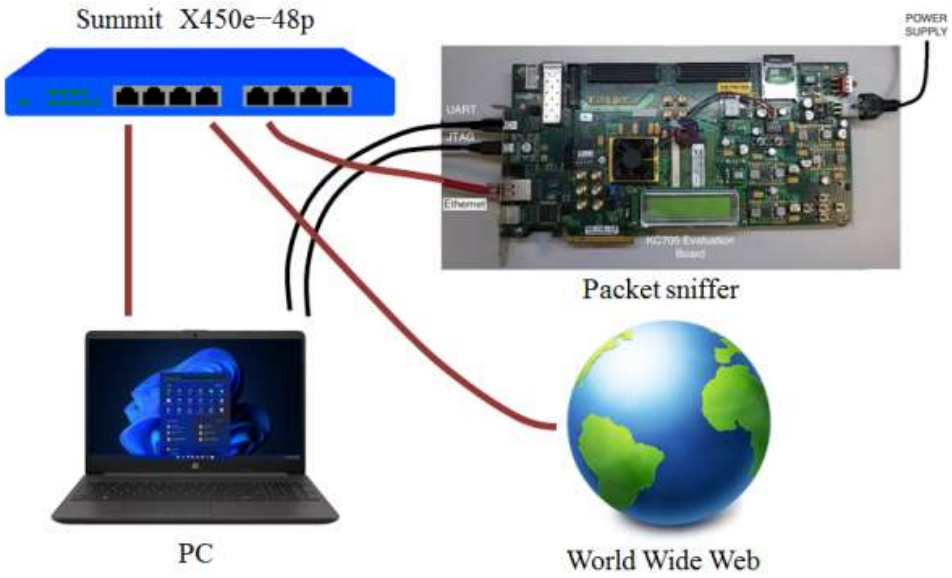


Figure 21. Experimental setup where the PC Ethernet traffic is mirrored on the packet sniffer and real Ethernet traffic is generated by the PC.

In this configuration a switch Summit X450e-48p by Extreme Networks has been used to provide the same internet traffic for the packet sniffer and the PC running Wireshark. The Ethernet port of the PC is connected to port 2 of the switch and the traffic on port 2 is mirrored to port 10 where the Ethernet port of the packet sniffer is connected. A router is connected to port 6 of the switch to allow internet connection of the PC.

In a first test a ICMP flood attack towards the PC connected to port 2 of the switch was emulated. This was carried out with the software UDP Flooder using another PC connected on the same LAN to generate the attack. Figure 22 shows five different attacks which were generated for different time

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

periods. The results for the data measured using the packet sniffer and the PC running Wireshark are also shown. A cross-correlation analysis was carried out to investigate the correlation between packet sniffer data and Wireshark data using Real Statistics add-on for Excel. The results have shown that Wireshark data are delayed about 6 seconds and the maximum correlation factor is estimated as $r^2 = 0.99$.

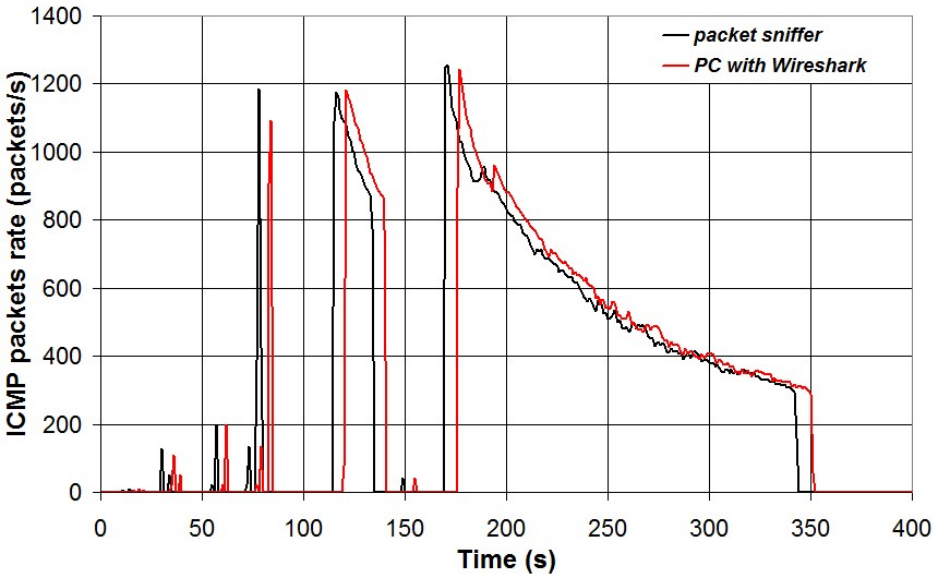


Figure 22. ICMP packets rate under different ICMP flood attacks detected with the packet sniffer and the PC running Wireshark.

A second test was carried out by generating UDP Ethernet traffic using an on-line video streaming service. A Youtube video of duration of about 2 minutes was viewed on the PC using different video resolutions (1080p, 720p and 480p) and the corresponding UDP traffic monitored with the packet sniffer and Wireshark. The number of UDP packets in 1 second detected by the packet sniffer is presented in Figure 23 for the video resolution of 1080p, in Figure 24 for the video resolution of 720p and in Figure 25 for the video resolution of 480p.

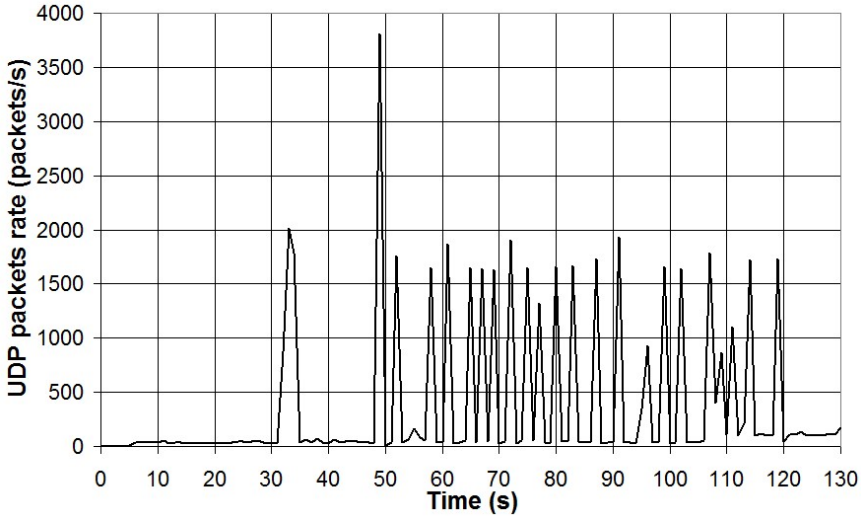


Figure 23. UDP packets rate in the case of video resolution 1080p.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

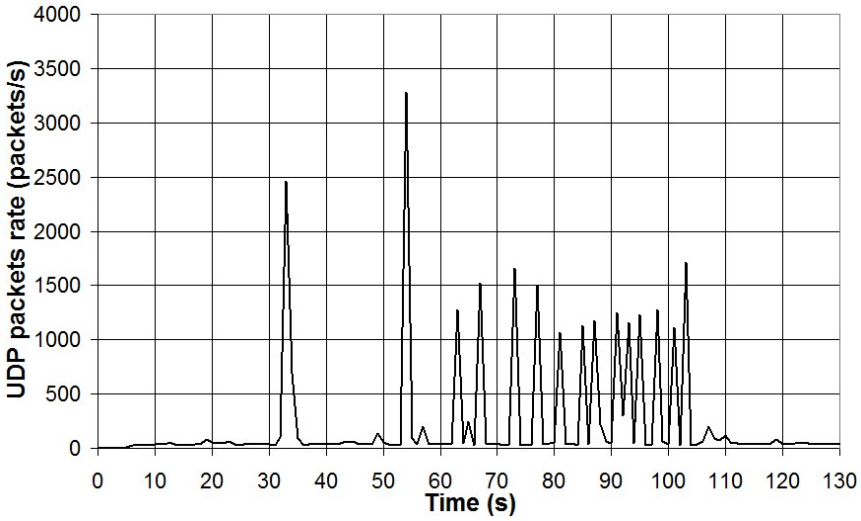


Figure 24. UDP packets rate in the case of video resolution 720p.

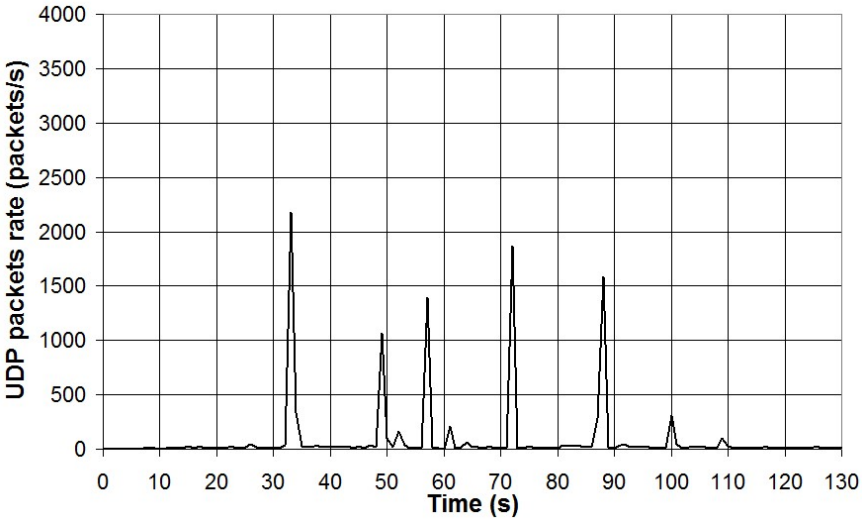


Figure 25. UDP packets rate in the case of video resolution 480p.

As can be seen, in the case of online video streaming, traffic is mainly characterized by spiked data traffic of Ethernet frames tagged UDP. The frequency of UDP data transfer is function of the video resolution (i.e. the higher the resolution, the higher the data traffic). A cross-correlation analysis was carried out between data from the packet sniffer and Wireshark. The results have shown that the correlation coefficient r^2 is 0.78 in the case of video resolution 1080p, 0.92 in the case of 720p and 0.57 in the case of 480p. The lower values of the correlation coefficient can be explained by the spiked characteristic of the data transfer and because the packet sniffer and the PC running Wireshark are not synchronized. Thus, the spiked data transfers can be read for the packet sniffer and the PC running Wireshark with a variable delay and this affects the correlation between the two sets of data. Total UDP data transfer has been measured for both the packet sniffer and the PC running Wireshark and the results are reported in Table 1.

Table 1. Total UDP data transferred in the case of Youtube video with resolution 1080p, 720p and 480p for the packet sniffer and the PC running Wireshark.

Device	Video resolution		
	1080p	720p	480p
packet sniffer	56.72 MB	33.26 MB	20.44 MB

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

PC running Wireshark	56.51 MB	33.14 MB	20.27 MB

A third test was carried out by downloading a file (Arduino IDE, file size 197 MB) using the Linux command "wget". In this case, data transfer is achieved using the TCP protocol and the data rate (TCP data per second) is presented in Figure 26.

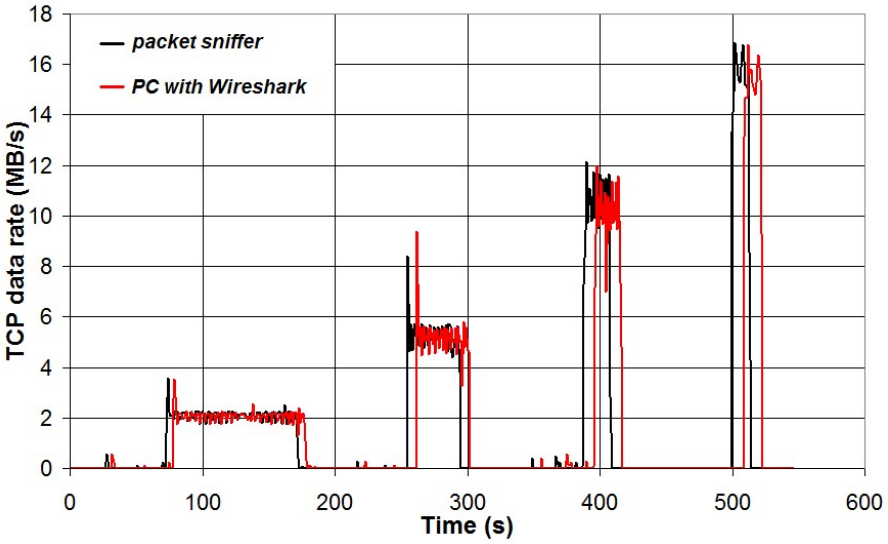


Figure 26. TCP data rate in MB/s in the case of file download using the Linux command "wget" for the packet sniffer and the PC running Wireshark.

The file was downloaded four times using different limits for the download bandwidth: 2 MB/s, 5 MB/s, 10 MB/s and 15 MB/s. A cross-correlation test was carried out and the results have shown that data from Wireshark have a delay of about 9 seconds if compared to the packet sniffer data and the measured correlation coefficient is high (0.97).

Finally, a test was carried out to check if the packet sniffer can properly discriminate data according to the rules defined in Section 3.2. Two different files were downloaded using the Linux command "wget" from two different IP addresses (82.197.215.15 and 104.18.12.241). The rules were defined to allow data transfer only from the IP address 104.18.12.241. The results of such test are reported in Figure 27. As can be seen, TCP data downloaded from 104.18.12.241 are marked as allowed data, while data downloaded from 82.197.215.15 are marked as data with rules violation.

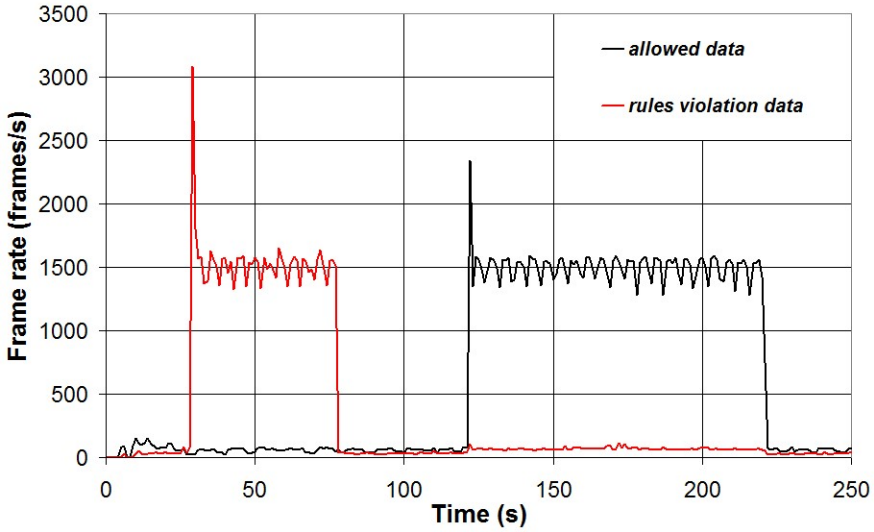


Figure 27. TCP packets rate in the case of file download from two different IP addresses (104.18.12.241 and 82.197.215.15) using the Linux command "wget" for the packet sniffer.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 13 September 2023 doi:10.20944/preprints202309.0884.v1

5. Ethernet data analysis for higher data rate

A packet sniffer has been designed in hardware on FPGA that can analyze Ethernet traffic with a maximum data rate of 1 Gbit/s. However, more recent standards have been defined for Ethernet that allow higher data speeds (such as 10 Gbit/s and 100 Gbit/s).

Preliminary investigations have been carried out to extend the designed packet sniffer to work at the higher data rate of 10 Gbit/s. The KC705 development board has been used to create a packet generator working at 10 Gbit/s and the transmitted Ethernet frames have been looped back on the receive channel of the same board. In this case, to support the higher data rate, the Small Form-Factor Pluggable (SFP) port on the development board has been used instead of the J45 port and data transfer has been achieved using optical fiber. The physical layer of Ethernet transmission is handled in this case by a PHY module instantiated inside the FPGA instead of the external PHY chip (Marvell M88E1111-BAB1C000) that can handle a maximum data speed of 1 Gbit/s. The IP module 10G Ethernet Subsystem v 3.1 provided by Xilinx has been used to handle the layers 1 and 2 of the OSI protocol.

Figure 28 shows the simulations of a single frame data transfer (UDP packet of 60 bytes length). While in the case of 1 Gbit/s data rate, data are acquired with data width 1 byte using a clock of 125 MHz, in the case of 10 Gbit/s data rate the data width is increased to 8 bytes using a clock of 156.25 MHz. In summary, while in the case of 1 Gbit/s, a 60 bytes length frame needs 60 clock cycles for data transfer, in the case of 10 Gbit/s the number of clock cycles is reduced to 8. Moreover, in the case of 100 Gbit/s, data are acquired with data width of 64 bytes, thus a 60 bytes length frame needs a single clock cycle for data transfer.

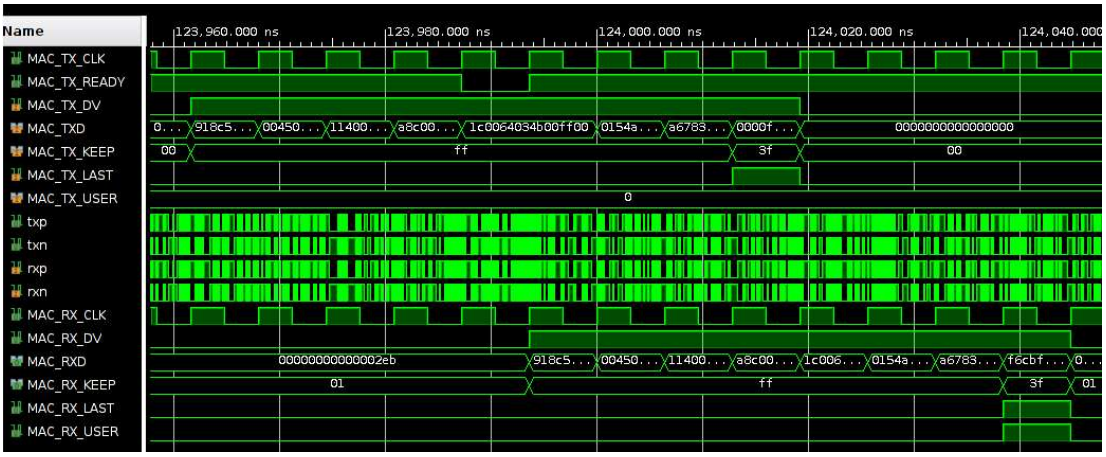


Figure 28. Signal waveforms in the case of data transmission and reception with 10 Gbit/s data rate.

The reduction of the number of clock cycles needed for a data frame transfer for higher data rates results in critical timing for the frame analysis and the verification of the packet sniffer rules. While in the current version (1 Gbit/s) the timing is adequate for the verification of up to 256 different packet sniffer rules, higher data rates pose strict timing requirements that need the redesign of the frame analysis hardware, for example using a content addressable memory (CAM), i.e. a type of memory that can check the presence of particular data in a single clock cycle.

6. Conclusions

In this paper a stateless packet sniffer implemented on FPGA has been described in depth. The packet sniffer is designed using a commercial FPGA development board (KC705 by Xilinx) and supports data transfer rates up to 1 Gbit/s. The designed system analyses Ethernet frames of various types (ARP, IP, UDP, TCP, ICMP), calculates the principal frame fields (such as MAC addresses, IP addresses, source and destination ports), and evaluates potential threats of the received data based on a set of rules defined by the user.

The packet sniffer has been tested first under controlled conditions, generating sets of Ethernet frames using an ad-hoc designed packets generator, and then under real web traffic by connecting the system to the internet. A summary report of the system performance was presented during different tests, including a video streaming and a simulation of a ICMP attack. The results have shown how the designed system can reliably detect potential threats in the received data. Future investigations in this research line are aimed at increasing the speed of the packet sniffer by updating the system to work at data rates of 10 Gbit/s and 100 Gbit/s.

Applications can vary from network monitoring to security of high-speed vertical internet connections. FPGAs have been confirmed to be the best choice for this kind of applications for their inherent low-latency, high-throughput, and simple system reconfigurability.

Author Contributions: Conceptualization, M.G., A.G. and F.A.; methodology, M.G., A.G. and F.A.; software, M.G. and F.A.; validation, M.G. and F.A.; formal analysis, M.G. and F.A.; investigation, M.G. and F.A.; resources, M.G. and F.A.; data curation, M.G.; writing—original draft preparation, M.G.; writing—review and editing, M.G., A.G., F.A. and M.P.; visualization, M.G., A.G. and F.A.; supervision, A.G. and M.P.; project administration, A.G. and M.P.; funding acquisition, A.G.

Funding: Italian Ministry of University and Research, Grant/Award Number: J45F21002000001; "Alma Idea 2022" Linea di Intervento A (D.M. 737/2021); Italian Ministry of Industry Incentives (MISE); Ministry of University and Research (MUR).

Acknowledgments: The authors would like to thank the National Institute for Nuclear Physics (INFN, Bologna division) and the National Center for Frame Analysis (CNAF, Bologna division) for the support in the development and testing of the presented packet sniffer.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sudar, K.M.; Deepalakshmi, P.; Nagaraj, P.; Muneeswaran, V. Analysis of cyberattacks and its detection mechanisms. *IEEE Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, **2020**, 12 − 16.
2. Lezzi, M.; Lazoi, M.; Corallo, A. Cybersecurity for Industry 4.0 in the current literature: A reference framework. *Computers in Industry*, **2018**, *103*, 97 − 110.
3. Corallo, A.; Lazoi, M.; Lezzi, M. Cybersecurity in the context of industry 4.0: A structured classification of critical assets and business impacts. *Computers in Industry*, **2020**, *114*, 103165.
4. Mullet, V.; Sondi, P.; Ramat, E. A review of cybersecurity guidelines for manufacturing factories in industry 4.0. *IEEE Access*, **2021**, *9*, 23235 − 23263.
5. Coventry, L.; Branley, D. Cybersecurity in healthcare: A narrative review of trends, threats and ways forward. *Maturitas*, **2018**, *113*, 48 − 52.
6. Anwar, R.W.; Abdullah, T.; Pastore, F. Firewall best practices for securing smart healthcare environment: A review. *Applied Sciences*, **2021**, *11 (19)*, 9183.
7. Giansanti, D. Cybersecurity and the digital-health: The challenge of this millennium. *Healthcare*, **2021**, *9 (1)*, 62.
8. Neupane, K.; Haddad, R.; Chen, L. Next generation firewall for network security: a survey. *IEEE SoutheastCon*, **2018**, 1 − 6.
9. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, **2019**, *2 (1)*, 1 − 22.
10. Siswanto, A.; Syukur, A.; Kadir, E.A. Network traffic monitoring and analysis using packet sniffer. *IEEE International Conference on Advanced Communication Technologies and Networking (CommNet)*, **2019**, *1* − 4.
11. Nahar, N.; Kumar, R. An improved Linux firewall using a hybrid frame of netfilter. *IEEE International Conference on Trends in Electronics and Informatics (ICEI)*, **2017**, *657* − 662.
12. Nivethan, J.; Papa, M. A Linux-based firewall for the DNP3 protocol. *IEEE symposium on technologies for homeland security (HST)*, **2016**, *1* − 5.
13. Tirumala, S.S.; Nepal, N.; Kumar Ray, S. Raspberry pi-based intelligent cyber defense systems for SMEs and smart-homes: An exploratory study. *EAI Endorsed Transactions on Smart Cities*, **2022**, *6 (18)*, e4 – e4.
14. Phalguni, J.; Santosh Krishna, M. Design of a Firewall Based on Linux Netfilter using ARM9. *International Journal of Scientific Engineering and Technology Research*, **2015**, *4 (36)*, 7744 − 7748.
15. Oluwabukola, O.; Oludele, A.; Ogbonna, A.C.; Chigozirim, A.; Amarachi, A. A Packet Sniffer (PSniffer) application for network security in Java. *Proceedings of the Informing Science and Information Technology Education Conference*, **2013**, 389 − 400.

16. Phang, S.Y.; Lee, H.; Lim, H. Design and implementation of V6SNIFF: an efficient IPv6 packet sniffer. *IEEE Third International Conference on Convergence and Hybrid Information Technology*, **2008**, *2*, 44 − 49.
17. Goyal, P.; Goyal, A. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. *IEEE 9th International Conference on Computational Intelligence and Communication Networks (CICN)*, **2017**, 77 − 81.
18. Niemiec, G.S.; Batista, L.M.S.; Schaeffer-Filho, A.E.; Nazar, G.L. A survey on FPGA support for the feasible execution of virtualized network functions. *IEEE Communications Surveys & Tutorials*, **2019**, *22 (1)*, 504 − 525.
19. Wicaksana, A.; Sasongko, A. Fast and reconfigurable packet classification engine in FPGA-based firewall. *IEEE International Conference on Electrical Engineering and Informatics*, **2011**, 1 − 6.
20. Fiessler, A.; Hager, S.; Scheuermann, B.; Moore, A.W. HyPaFilter: A versatile hybrid FPGA packet filter. *Symposium on Architectures for Networking and Communications Systems*, **2016**, 25 − 36.
21. Ezzati, S.; Naji, H.R.; Chegini, A.; Habibimehr, P. Intelligent firewall on reconfigurable hardware. *European Journal of Scientific Research*, **2010**, *47 (4)*, 509 − 516.
22. Sommer, J.; Gunreben, S.; Feller, F.; Kohn, M.; Mifdaoui, A.; Saß, D.; Scharf, J. Ethernet–a survey on its fields of application. *IEEE Communications Surveys & Tutorials*, **2010**, *12 (2)*, 263 − 284.
23. Briscoe, N. Understanding the OSI 7-layer model. *PC Network Advisor* , **2000**, *120 (2)*, 13 − 15.
24. Tiller, J.S. *A technical guide to IPSec virtual private networks*, CRC Press, **2017**.
25. AMD Kintex7 FPGA KC705 Evaluation Kit: https://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html (accessed on August 4th 2023).
26. Tri-Mode Ethernet Media Access Controller (TEMAC): https://www.xilinx.com/products/intellectual-property/temac.html (accessed on August 4th 2023).