future internet

Article

Increasing the Security of Network Data Transmission with a Configurable Hardware Firewall Based on Field Programmable Gate Arrays

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

1 Department of Electrical Energy and Information Engineering "Guglielmo Marconi" (DEI), Alma Mater Studiorum, Università di Bologna, 40136 Bologna, Italy
2 Istituto Nazionale di Fisica Nucleare (INFN) Bologna, 40127 Bologna, Italy; fabrizio.alfonsi@bo.infn.it (F.A.); alessandro.gabrielli@unibo.it (A.G.)
3 Department of Computer Science and Engineering, Alma Mater Studiorum, Università di Bologna, 40126 Bologna, Italy; marco.prandini@unibo.it
4 Department of Physics and Astronomy "Augusto Righi" (DIFA), Alma Mater Studiorum, Università di Bologna, 40127 Bologna, Italy
* Correspondence: marco.grossi8@unibo.it; Tel.: +39-051-2093038

Abstract: One of the most common mitigations against network-borne security threats is the deployment of firewalls, i.e., systems that can observe traffic and apply rules to let it through if it is benign or drop packets that are recognized as malicious. Cheap and open-source (a feature that is greatly appreciated in the security world) software solutions are available but may be too slow for high-rate channels. Hardware appliances are efficient but opaque and they are often very expensive. In this paper, an open-hardware approach is proposed for the design of a firewall, implemented on off-the-shelf components such as an FPGA (the Xilinx KC705 development board), and it is tested using controlled Ethernet traffic created with a packet generator as well as with real internet traffic. The proposed system can filter packets based on a set of rules that can use the whitelist or blacklist approach. It generates a set of statistics, such as the number of received/transmitted packets and the amount of received/transmitted data, which can be used to detect potential anomalies in the network traffic. The firewall has been experimentally validated in the case of a network data throughput of 1 Gb/s, and preliminary simulations have shown that the system can be upgraded with minor modifications to work at 10 Gb/s. Test results have shown that the proposed firewall features a latency of 627 ns and a maximum data throughput of 0.982 Gb/s.

Keywords: network security; firewall; FPGA; Ethernet; packet classification; embedded systems

check for updates

Citation: Grossi, M.; Alfonsi, F.; Prandini, M.; Gabrielli, A. Increasing the Security of Network Data Transmission with a Configurable Hardware Firewall Based on Field Programmable Gate Arrays. *Future Internet* **2024**, *16*, 303. https://doi.org/10.3390/fi16090303

Academic Editors: Massimo Cafaro, Italo Epicoco and Marco Pulimeno

Received: 19 July 2024
Revised: 17 August 2024
Accepted: 21 August 2024
Published: 23 August 2024

1. Introduction

Network activity has increased significantly in the last two decades. From wired communication, networking data has evolved towards wireless communication thanks to the widespread diffusion of smart mobile devices [1]. Web applications have continued to increase in number and cover a large number of applications, such as healthcare [2], smart mobility [3], e-commerce [4] and social networks [5]. Similarly, in recent years, the interconnection of sensors and actuators with high-performance computing devices and wireless communication has led to the deployment of cyber–physical systems in the paradigm of the Internet-of-Things [6–8]. The high volume of exchanged data traffic has resulted in increased security threats due to cyber-attacks that exploit the system's vulnerabilities for unauthorized access to the system, stealing personal information and/or creating damage to the system [9,10].

Several techniques have been adopted to mitigate the risks of cyber-attacks, such as authentication procedures to prevent unauthorized access to the system [11,12], cryptographic data obfuscation to make the information useless to malicious users listening to private communications [13,14], as well as firewalls [15,16] and packet sniffers [17,18] for the control and monitoring of network traffic. In the case of identity authentication and data obfuscation, the procedure is carried out by using passwords and cryptographic keys that are usually stored in a non-volatile memory. This poses significant security threats since the non-volatile memory can be hacked to disclose confidential information. A technique to mitigate this issue is to replace the non-volatile memory with a physical unclonable function (PUF), a device that generates a unique fingerprint by exploiting the random variations in its parameters generated during the production process [19–21].

Firewalls and packet sniffers are the basic building blocks for intrusion detection/prevention systems (IDSs, IPSs). They are used to control and monitor the network traffic and mitigate the security risks resulting from malicious data that exploit the network to reach and compromise vulnerable systems. Firewalls are devices working in active mode, featuring two full duplex ports, which can block the passage of network data if security threats are detected. Packet sniffers, on the other hand, are devices working in passive mode, analyzing the network traffic and sending an alarm message to a remote server if security threats are detected. Firewalls and packet sniffers are usually implemented as software running on a PC. Many software firewalls have been proposed in the literature for the GNU/Linux operating system, exploiting the Iptables kernel feature (now superseded by nftables) to set the rules for the discrimination between safe and potentially dangerous data [22–24]. Similarly, many packet sniffers implemented in software exist, where the most popular are Wireshark [25] and TCPdump [26]. Firewalls and packet sniffers implemented in software can reliably detect network security issues as long as the network data throughput is not too high. On the contrary, when the network data throughput exceeds a certain threshold, the software implementation can lose effectiveness, mainly due to the CPU being unable to keep up with the full network data speed.

In the case of high network data throughput, a hardware implementation is preferable. The hardware implementation of firewalls and packet sniffers is usually realized using Application Specific Integrated Circuits (ASICs) since this approach provides the best performance and the minimization of power consumption. However, the ASIC implementation of firewalls and packet sniffers also has significant drawbacks, such as high non-recurring engineering costs and long times for the system design. As a consequence, ASIC-based commercial firewalls are usually expensive and their design is a highly protected intellectual property, hindering the possibility of users knowing what a security-critical device, placed in their network, is doing. From this point of view, a Field Programmable Gate Array (FPGA) implementation has significant advantages in terms of faster design, lower costs, in particular for the realization of a small number of prototypes, and openness.

The objective of our research is the development of a hardware firewall on an FPGA to check the safety of Ethernet traffic among universities and research centers. The protection of scientific data is an essential aspect of ensuring data security since cyber-attacks and data breaches are increasing in number in higher education institutions and universities [27]. The designed firewall should be able to support high data rates (at least 1 Gb/s) and must feature a high configurability, where the single user is able to define the features of the data that must be allowed or blocked. In the current stage of development, our research group has focused on the design of a stateless firewall, but future works on this research project will be aimed at the integration of stateful packet inspection and/or machine-learning algorithms for security threat detection.

In this paper, a hardware firewall based on an FPGA is presented. The firewall is implemented on a Xilinx KC705 development board that integrates the Kintex-7 XC7K325T-2FFG900C FPGA. It is designed to work at a maximum data throughput of 1 Gb/s, but preliminary simulations have shown that the system performance can be extended to a higher data throughput of 10 Gb/s. The system features a highly configurable set of rules to discriminate between safe and potentially dangerous data that can be defined according to a whitelist or blacklist approach. It also calculates a set of statistics (such as the number of received/transmitted packets, the packet length, etc.) that can be downloaded to a PC using the USB-UART interface for further data analysis. Experimental results have shown that the designed firewall can achieve a data throughput of 0.982 Gb/s with a latency of 627 ns, and the percent of packet loss due to memory resource exhaustion is negligible ($3.29 \cdot 10^{-5}$%). The results of the paper represent the continuation of the research line presented in [17] and [18], where a packet sniffer was designed on an FPGA for network data throughputs of 1 Gb/s and 10 Gb/s, respectively.

The paper is organized as follows. In Section 2, a short review of the related works of hardware firewalls implemented on an FPGA is presented. In Section 3, the design of the proposed hardware firewall and its main features are presented. In Section 4, the experimental results achieved with the proposed firewall are presented along with preliminary simulations to extend the device data throughput to 10 Gb/s. In Section 5, the performance of the designed firewall is compared with other firewalls from the literature. Finally, concluding remarks are presented in Section 6.

2. Related Work

Different firewalls have been implemented in hardware on FPGAs and the results have been presented in the literature. The most common type of firewall is the one often referred to as the stateless firewall. This device does not store the status of the network connection, but every packet is analyzed as it is, by comparing against a set of rules the information contained in the header of the network and transport layers.

Mohammed and Ueno, in 2018, proposed a network firewall based on an FPGA to monitor Ethernet traffic [28]. The system was implemented on the NetFPGA-1G hardware accelerator that integrates a Xilinx Kintex-7 325T FPGA. The verification of the firewall rules was implemented using a content addressable memory (CAM) for high-speed data search. The firewall was designed with both software and hardware components and its performance was tested on a network with four different Ethernet ports. The results showed that the firewall latency was 402 µs and the maximum data throughput was about 800 Mb/s, over two times the data throughput achieved with a Linux-Iptables software firewall.

Lin et al., in 2017, presented the design of an Ethernet firewall based on an FPGA [29]. The proposed system was implemented on an Altera EP4CE115F29 FPGA, exploiting a microcontroller (NXP Semiconductor MK60DN512VLQ10) to test the firewall rules based on a whitelist approach and two external devices (Marvell Technology 88E1111) to handle the physical layer of the Ethernet protocol. Tests were carried out with two PCs exchanging Ethernet data through the firewall, and the results showed that a maximum data throughput of 950 Mb/s and a latency of 61.266 µs could be achieved.

Maloji Keni and Mande, in 2018, presented the design and implementation of a hardware firewall using FPGA [30]. The proposed firewall, implemented on a Xilinx Spartan 6 FPGA development board, features logic to compare the packet IP addresses with a set of addresses stored on non-volatile memory to decide if the packet must be transmitted or discarded. The authors reported that the proposed hardware firewall is faster than a conventional software firewall. However, the system was tested only with packets generated on the same FPGA board and with a fixed size of 512 bits.

Ajami and Dinh, in 2011, proposed a hardware network firewall on an FPGA [31]. The proposed system was implemented on an Altera Stratix II EP2S60F672C5 FPGA and was designed using a NIOS II soft-core 32-bit microprocessor implemented in the FPGA programmable logic, an Ethernet module LAN91C111 (Microchip Technology) to handle the physical and medium access control layers of the Ethernet protocol, two RAM modules to store the firewall rules and a CAM for low packet processing latency. The system was tested by interfacing the firewall with two PCs used to generate packets. The measured packet processing time was 1.24 µs for ARP packets and 1.76 µs for ICMP packets.

Antonov et al., in 2016, presented an FPGA-based firewall for enhancing the security of IP networks [32]. The proposed firewall architecture features packet filtering algorithms realized with circuits implemented in the FPGA programmable logic, two cores of an

ARM Cortex A9 microprocessor, 1 GB of dynamic RAM (DDR3) and an SD card reader to store the microprocessor OS and statistical data generated by the firewall. The proposed architecture was simulated for both Xilinx and Altera FPGAs, and a maximum working clock frequency of 300 MHz was estimated.

Ricart-Sanchez et al., in 2019, presented a fully functional FPGA firewall for the detection of cyber-attacks in 5G networks [33]. The firewall was implemented on a P4-NetFPGA development board and was designed using a pipeline to minimize the latency. Tests were carried out by connecting the firewall between two PCs, and the results showed how the firewall latency was only marginally affected by the number of firewall rules. The system was designed for a 10 Gb/s network but a maximum data throughput of 3.67 Gb/s was reported. The percent of packet loss was reported to be about 2.5% in the worst-case scenario.

Salopek and Mikuc, in 2023, proposed a hybrid hardware/software firewall based on an FPGA for the mitigation of distributed denial of service (DDoS) attacks [34]. The system was implemented on a NetFPGA SUME development board that features a Xilinx Virtex-7 690T FPGA, four 10 Gb/s SFP+ interfaces, static and dynamic RAM and other peripherals. The packet processing is implemented in hardware with the FPGA programmable logic and the packet data are forwarded to a software filter after some metadata have been appended. The results showed how the proposed architecture that distributes the workload between hardware and software components can achieve a very good performance and is effective in the mitigation of DDoS attacks.

A more advanced type of firewall is the one referred to as a stateful firewall, which checks the network packets, tracking the state of connections based on knowledge of the protocols used in the network connection.

Bianchi et al., in 2016, proposed Open Packet Processor, a programmable architecture for platform-independent stateful in-network processing [35]. The system was implemented on the development board NetFPGA SUME, with a clock of 156.25 MHz and a 64-bit data path from the Ethernet ports, corresponding to a 10 Gb/s data throughput for each port. The authors reported a data throughput between 10^7 and $8{\cdot}10^7$ packets/s.

Pontarelli et al., in 2019, presented FlowBlaze, a firewall based on Extended Finite State Machines (EFSMs) that can support a wide range of complex network functions and can make stateful packet inspection [36]. The proposed device was implemented on the NetFPGA SUME SmartNIC, an x8 Gen3 PCIe adapter card containing a Xilinx Virtex-7 690T FPGA and four SFP+ transceivers providing four 10 Gb/s Ethernet links. Experimental measurements of the FlowBlaze performance showed that the device is characterized by a latency of a few microseconds and a maximum data throughput of $14.88{\cdot}10^6$ packets/s in the case of packet sizes of 64 bytes, but the authors state that, in principle, a maximum data throughput of 40 Gb/s is possible.

Research activities were recently carried out on the exploitation of machine-learning algorithms to detect potential security threats in network communication.

Tran et al., in 2017, designed a heterogeneous anomaly-based intrusion detection system (HA-IDS) which is built on an FPGA and a Graphics Processing Unit (GPU) [37]. The system uses the FPGA (Xilinx Virtex-5 XC5VTX240T) to collect the network data and extract the packets' information, while the GPU (Gigabyte GeForce GTX 1080) is used to implement a back-propagation neural network to detect anomalies in the network data. Experimental results from studies carried out on a 2 GB dataset (consisting of regular packets as well as DDoS attack packets) show that the system can achieve an accuracy of 80.42% with a data throughput of 200 Mb/s.

Le Jeune et al., in 2021, presented an intrusion detection system on an FPGA based on deep learning [38]. The system was implemented on a PYNQ-Z2 development board with a Xilinx ZYNQ XC7Z020-1CLG400C FPGA on board and achieved an accuracy of 99.41% with a data throughput of 90 Mb/s.

Murovič and Trost, in 2021, proposed a binary neural network (BNN) on an FPGA to design an intrusion detection system [39]. Different single hidden layer BNNs were implemented on a Xilinx Kintex Ultrascale+ FPGA (XCKU3P) and trained with two different datasets (NSL-KDD and UNSW-NB15). The results showed that the designed system can achieve an accuracy between 77.77% and 98.96%, it is very efficient in resource usage (8606 to 17,990 lookup tables) and features a very low classification latency (from 16 ns to 19 ns).

A different type of firewall deals with the security of bus transactions of microprocessors-based embedded systems from potential attackers. Since these systems perform computationally intensive algorithms and manipulate confidential information such as passwords, the minimization of detection latency is very important.

Lázaro et al., in 2022, presented a firewall based on the Advanced eXtensible Interface (AXI) for System-on-Chip security [40]. The objective was to design a system to detect attacks against microprocessors and the designed firewall was located on an AXI bus to protect an AXI slave device from the attacks of a fraudulent AXI master, such as accessing non-allowed registers or carrying out a denial of service (DoS) attack. The proposed AXI firewall was implemented on a Xilinx Zync7 FPGA device. Experimental results showed that the system can efficiently work at a maximum frequency of 166 MHz and, due to the implementation of the AXI to AXI path with combinatorial logic, it can achieve a latency of zero clock cycles.

Restuccia and Kastner, in 2022, proposed a novel switching method for multicomponent communication architectures on FPGA systems on chips to enable safe and secure bus access and minimize the impact on performance and resource usage [41]. The proposed technique was implemented on the Zynq Ultrascale+ (ZCU102) and Zynq Z-7020 (PYNQ) platforms and the results showed the feasibility of detecting a wide range of software attacks with significant benefits in terms of low resource usage and processing latency minimization.

3. The Proposed Firewall on an FPGA

The schematic of a firewall working principle is presented in Figure 1. Here the firewall is protecting a Local Area Network (LAN), which interconnects several PCs in a limited area from potentially dangerous data packets. The firewall features two different ports: Port A, connected to a router that manages the network traffic between the PCs in the LAN and the internet, and Port B, which transmits/receives the network traffic to/from the internet. Both ports are full duplex; that is, each port can transmit and receive data. When a network packet is received at Port A, the firewall analyzes the packet against a set of rules and, if no potential security threats are detected, it forwards the packet to Port B for transmission, or blocks it otherwise. Each firewall port features its set of rules that can be defined according to a whitelist approach, where only the packets that follow the rules are allowed to pass, or a blacklist approach, where only the packets that follow the rules are blocked.

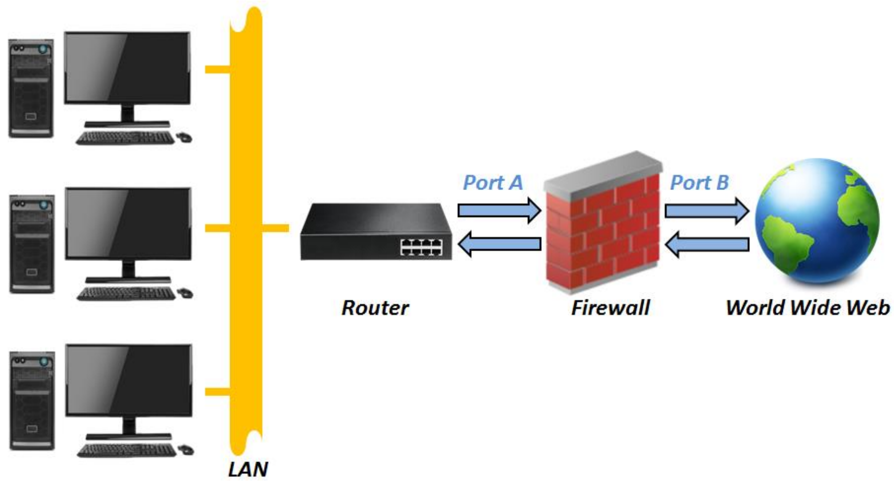

Figure 1. Schematic of the working principle of a firewall.

The proposed firewall is designed using the Verilog HDL [42] and is implemented on a Xilinx KC705 development board [43] that integrates the Kintex-7 XC7K325T-2FFG900C FPGA. The KC705 development board has been selected since it features interfaces for Ethernet communication at both 1 Gb/s and 10 Gb/s, and the on-board FPGA device has a good amount of hardware resources (326,080 logic cells, 840 DSP slices, about 2 MB of SRAM, 16 GTX transceivers and 500 pins) to implement our project. The designed firewall works at a maximum data throughput of 1 Gb/s, but preliminary simulations have shown that its performance can be extended to a higher data throughput of 10 Gb/s. Both ports of the firewall can be configured with a set of 256 rules, defined according to a whitelist or blacklist approach. The set of rules for each port can be defined using ad-hoc software developed in LabVIEW (National Instruments) [44] and uploaded to the FPGA memory using the USB-UART interface. The developed software generates a text file with a set of firewall rules: each rule is 176 characters on one line of the text file and sets a range for the IP source address, the IP destination address, the source and destination ports and the transport layer (i.e., Level 4) protocol. During the firewall initialization, the text file is read and its content is transferred to the FPGA memory using the USB-UART interface. The firewall can analyze packets that belong to the network layer (i.e., Level 3) protocols ARP and IP, and, in the case of the IP protocol, can analyze packets that belong to the transport layer (i.e., Level 4) protocols UDP, TCP and ICMP. It also calculates a set of statistics of the network traffic at both ports. These statistics include the number of packets that are transmitted to the output port, the number of packets that are blocked due to checksum errors, the number of packets that are blocked due to firewall rules violation, the number of packets dropped when the memory is full, the protocol of the received packets and the amount of used memory. Such statistics can be downloaded to a PC using the USB-UART interface for further data analysis.

The schematic of the proposed hardware firewall is presented in Figure 2, where the different hardware blocks implemented in the FPGA programmable logic and their interconnections are shown. Only the logic for the analysis of packets received at Port A is shown, for simplicity. In the real system, such logic is also duplicated for the packets received at Port B. The synchronization of the digital circuits is carried out using the 200 MHz LVDS crystal oscillator present on the KC705 development board. The 200 MHz clock is used to generate two different clock signals using the Xilinx IP 'Clock Wizard': a 125 MHz clock used for the circuits devoted to the packets transmission/reception and analysis (CLK_125MHz), and a 10 MHz clock used for the communication with a PC through the USB-UART interface (CLK_10MHz).

The USB-UART communication is managed by the 'UART controller' module that communicates with the PC using the 'UART RX' and 'UART TX' modules (baud rate 115,200, 8-bit data, 1 stop bit, no parity bit). The communication of the 'UART controller' module with the circuits implemented in the FPGA programmable logic is realized with a custom interface with an 8-bit data bus: UART_RX_DATA[7:0] that is valid when the signal UART_RX_DV is set for the received data, and UART_TX_DATA[7:0] that is valid when the signal UART_TX_DV is set for the transmitted data. The signal UART_TX_DONE informs the 'UART controller' module when a data byte has been transmitted.

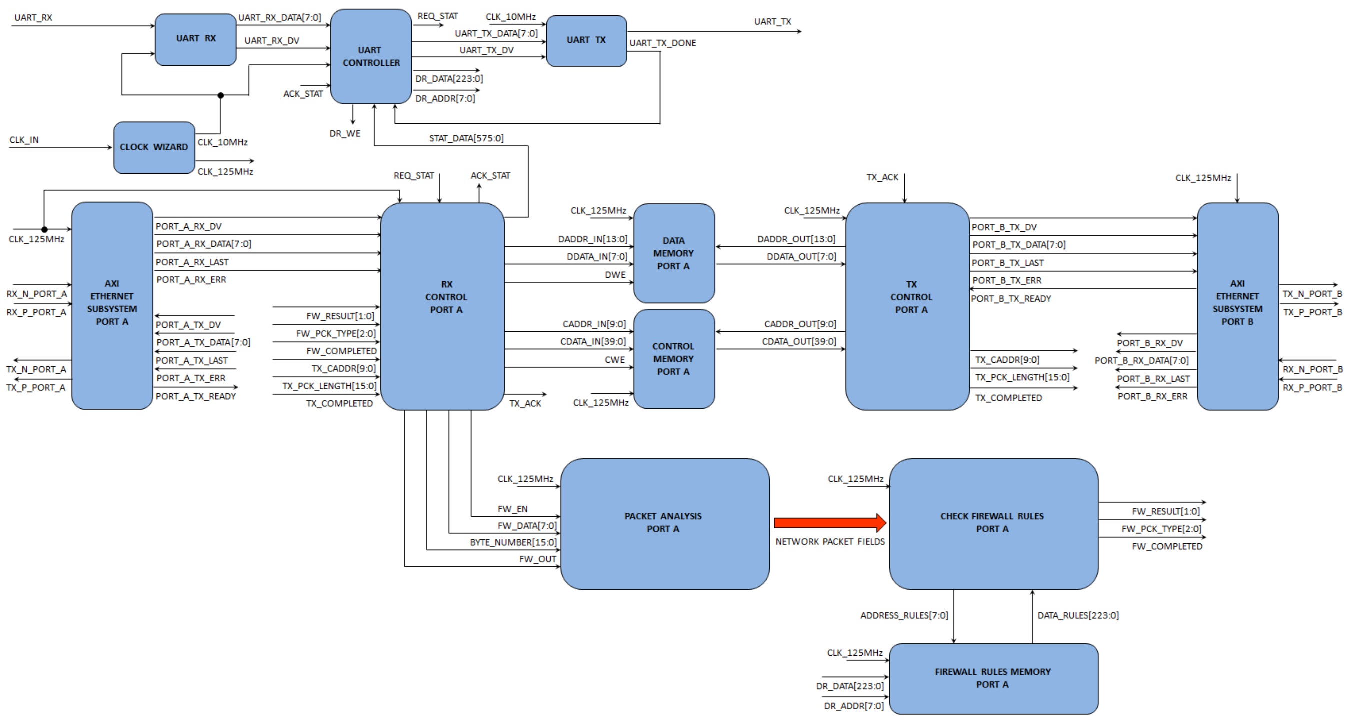

Figure 2. Schematic of the designed hardware firewall on an FPGA.

Each firewall port features the following:

1. An 'AXI 1G/2.5G Ethernet Subsystem' IP module by Xilinx [45]. This module manages the media access control (MAC) layer and the physical (PHY) layer of the Ethernet protocol. It is interfaced on the KC705 board with an SFP/SFP+ connector that transmits/receives the Ethernet data using fiber optic cables (differential signals RX_P, RX_N for the received data and differential signals TX_P, TX_N for the transmitted data). On the FPGA side, the packet data are transferred with an 8-bit data bus using the AXI communication protocol. In the case of the receiving port, for example, RX_DATA[7:0] stores the 8-bit data that are valid when the signal RX_DV is set, while RX_LAST is set during the transfer of the last byte of the packet. RX_ERR is a signal to indicate the presence of errors during the reception of the packet.
2. An 'RX control' module, designed using the Verilog HDL. This module is responsible for receiving the packet data from the 'AXI 1G/2.5G Ethernet Subsystem' module, storing the data in memory, and providing the packet data to the 'Packet analysis' module.
3. A 'Data memory' module that instantiates a dual-port SRAM using the block memory inside the FPGA, i.e., with the Xilinx IP module 'Block memory generator' [46]. This memory has a size of 16 kB and features 16,384 words of 1 byte each. It is used to store the received packet data before a decision is made on the compliance of the packet with the firewall rules. Each port of the memory features an address bus (14-bit), a data bus (8-bit), and a write enable (WE) signal that is set during a write operation.
4. A 'Control memory' module that instantiates a dual-port SRAM using the block memory inside the FPGA, i.e., with the Xilinx IP module 'Block memory generator' [46]. This memory has a size of 5 kB and features 1024 words of 40 bits each. Each port of the memory features an address bus (10-bit), a data bus (40-bit), and a write enable (WE) signal that is set during a write operation. Each word of the Control memory is composed of three fields and is associated with a specific packet whose data are stored in the Data memory. The most significant bits store the STATUS BYTE (8-bit), which defines the status of the packet: h0 for quarantined packets before a decision on the compliance with the firewall rules is made; h37 for packets that comply with

the firewall rules and can be transmitted; h2C for packets that do not comply with the firewall rules and must be blocked; h21 for packets with a checksum error that must be blocked. The other two fields are the PACKET START ADDRESS (16-bit), which stores the Data memory address of the first byte of the packet, and the PACKET LENGTH (16-bit), which stores the size of the packet in bytes.

5. A 'Packet analysis' module, designed using the Verilog HDL. This module receives the packet data (FW_DATA[7:0]) from the 'RX control' module: the data are valid when the signal FW_EN is set, while BYTE_NUMBER[15:0] indicates the position of the current data inside the packet. The signal FW_OUT is set one clock cycle after the reception of the last data byte. The 'Packet analysis' module is implemented as a combinational circuit (zero clock cycles latency) and calculates a set of parameters, such as the MAC source and destination addresses, the IP source and destination addresses and the source and destination ports. The packet parameters are then used by the 'Check firewall rules' module to decide on the compliance of the packet with the firewall rules.
6. A 'Check firewall rules' module, designed using the Verilog HDL. This module checks if the packet parameters comply with the firewall rules stored in the 'Firewall rules memory' and sends its decision to the 'RX control' module with the 2-bit data FW_RESULT[1:0] (that indicate if the packet can be considered safe or not) and the 3-bit data FW_PCK_TYPE[2:0] (that indicate the protocol of the processed packet). Both FW_RESULT[1:0] and FW_PCK_TYPE[2:0] are valid when the signal FW_COMPLETED is set.
7. A 'Firewall rules memory' module that instantiates a dual-port SRAM using the block memory inside the FPGA, i.e., with the Xilinx IP module 'Block memory generator' [46]. This memory has a size of 7 kB and features 256 words of 224 bits each. Each 224-bit word stores a single firewall rule. Each port of the memory features an address bus (8-bit), a data bus (224-bit) and a write enable (WE) signal that is set during a write operation. The firewall rules are written in the memory during the initialization by the 'UART controller' module and are read by the 'Check firewall rules' module during the packet processing step.
8. A 'TX control' module, designed using the Verilog HDL. This module is interfaced with the Data memory and Control memory and checks the status of the packet before a decision is made if the packet must be transmitted to the output port or must be blocked. The packets that are allowed to pass the firewall are sent to the 'AXI 1G/2.5G Ethernet Subsystem' module using the same 8-bit AXI communication protocol of the 'RX control' module.

The operations of the packet processing in the firewall are presented in the flow-chart of Figure 3, while the waveforms of the signals obtained with a simulation in the Vivado design suite are presented in Figure 4. The packet processing steps in Figure 3 are not carried out sequentially, but the operations are pipelined to minimize the firewall latency (i.e., the delay between the packet reception at the input port and the packet transmission at the output port) and maximize the data throughput.

The packet reception is carried out as long as the Data memory occupation (MEM_{used} in Figure 3) is lower than 12 kB, i.e., after the processing of one packet is completed the value of MEM_{used} is checked (step R7 in Figure 3). When Data memory occupation exceeds 12 kB, the firewall stops receiving packets until the Data memory occupation falls below 5 kB (step R8 in Figure 3). The operations of packet reception and subsequent analysis, as presented in Figure 3, can be summarized as follows:

1. When the Xilinx IP module 'AXI 1G/2.5G Ethernet Subsystem' receives an Ethernet frame from the fiber optic cable connected to the SFP/SFP+ connector, it handles the physical layer of the Ethernet frame, checks the correctness of the frame check sequence (FCS) field and transmits the packet (of length n bytes) to the FPGA logic using AXI bus protocol. The latency associated with this operation (Step R1 in Figure 3) can be estimated from the module datasheet as $T_{SFP\ to\ AXI}$ = 200 ns [45].

2. When the module 'RX control' receives the packet, it stores the packet data in the Data memory from address i to address i + n − 1 (Step R2 in Figure 3) and updates the content of Control memory, by setting the control word at address j with STATUS BYTE = 0 h, PACKET START ADDRESS = i and PACKET LENGTH = n (Step R3 in Figure 3). The value of Data memory occupation is then increased (Step R4 in Figure 3) and the address of the Data memory and Control memory is updated (Step R5 in Figure 3). Concurrently, it sends the packet data to the 'Packet analysis' module. When all data for packet analysis are available, the signal FW_OUT is asserted. The latency associated with these operations (i.e., the delay between the reception of the last byte of the packet and the assertion of FW_OUT) is 1 clock cycle, thus $T_{\mathrm{RX\ to\ FW\_OUT}}$ = 8 ns.
3. The module 'Packet analysis' calculates the packet fields (Step A1 in Figure 3) and operates concurrently with the 'RX control' module. The module 'Packet analysis' is implemented with a combinational circuit. Thus, when the signal FW_OUT is asserted, all the packet fields are valid and no latency is associated with this operation.
4. When the packet fields are valid, the 'Check firewall rules' module checks the compliance of the packet fields with the firewall rules (Step C1 in Figure 3) and generates the response in the 2-bit register FW_RESULT[1:0] (0 for a packet with checksum error, 1 for a packet that violates the firewall rules, 3 for a packet that can be transmitted) and the 3-bit register FW_PCK_TYPE[2:0] that defines the protocol of the packet (0 for ARP, 1 for TCP, 2 for UDP, 3 for ICMP, 4 for IPv6 and 5 for others). Then, the signal FW_COMPLETED is asserted to inform the 'RX control' module that the results of the packet analysis are available. The latency associated with this operation is 18 clock cycles, thus $T_{FW\_OUT\ to\ FW\_COMPLETED}$ = 144 ns.
5. When the signal FW_COMPLETED is asserted, the 'RX control' module updates the value of the STATUS BYTE for the Control Word present at address j of the Control memory, according to the value of the register FW_RESULT[1:0] (Step R6 in Figure 3). The latency associated with this operation is of 1 clock cycle, thus $T_{CW\ UPDATE}$ = 8 ns. At this time, the 'TX control' module can decide if the packet can be transmitted or must be blocked.

The operations of the transmission of the packets that comply with the firewall rules, as presented in Figure 3, can be summarized as follows:

1. The 'TX control' module continuously reads the Control Word stored at address k of the Control memory (Step T1 in Figure 3) and checks if the value of the STATUS BYTE is different from 0 (Step T2 in Figure 3). If the value of the STATUS BYTE is 37h (i.e., the packet complies with the firewall rules), the packet is sent to the 'AXI 1G/2.5G Ethernet Subsystem' module to be transmitted, or it is discarded, otherwise (Step T3 and T4 in Figure 3). Then, the signal TX_COMPLETED is set to inform the 'RX control' module that the packet was transmitted/discarded. The 'RX control' module can update the content of Control memory (Step R9 in Figure 3) and decrease the value of Data memory occupation (Step R10 in Figure 3). Finally, the value of k is incremented by the 'TX control' module (Step T5 in Figure 3). The latency associated with these operations is of 7 clock cycles, thus $T_{TX\ DECISION}$ = 56 ns.
2. The Xilinx IP module 'AXI 1G/2.5G Ethernet Subsystem' transmits the Ethernet frame using the fiber optic cable connected to the SFP/SFP+ connector. The latency associated with this operation can be estimated from the module datasheet as $T_{AXI\ to\ SFP}$ = 211 ns [45].

The latency of the firewall, i.e., the delay introduced between the time an Ethernet frame is received at the input port and the time the same Ethernet frame is transmitted at the output port, can be estimated as follows:

$$T_{FIREWALL} = T_{SFP\ to\ AXI} + T_{RX\ to\ FW\_OUT} + T_{FW\_OUT\ to\ FW\_COMPLETED} + T_{CW\_UPDATE} + T_{TX\_DECISION} + T_{AXI\ to\ SFP} \quad (1)$$

where $T_{SFP\ to\ AXI}$ = 200 ns (as reported in the 'AXI 1G/2.5G Ethernet Subsystem' product guide [45]), $T_{RX\ to\ FW\_OUT}$ = 8 ns (since the delay between the last byte of a packet and the assertion of the signal FW_OUT is of 1 clock cycle), $T_{FW\_OUT\ to\ FW\_COMPLETED}$ = 144 ns (since the delay introduced by the operation to check the compliance of a packet with the firewall rules is of 18 clock cycles), $T_{CW\ UPDATE}$ = 8 ns (since the delay introduced by the update of the Control memory by the 'RX control' module is of 1 clock cycle), $T_{TX\ DECISION}$ = 56 ns (since the delay introduced by the 'TX control' module to check the packet STATUS BYTE and to send the packet to in the 'AXI 1G/2.5G Ethernet Subsystem' module is of 7 clock cycles) and $T_{AXI\ to\ SFP}$ = 211 ns (as reported in the 'AXI 1G/2.5G Ethernet Subsystem' product guide [45]). Thus, $T_{FIREWALL}$ = 627 ns.

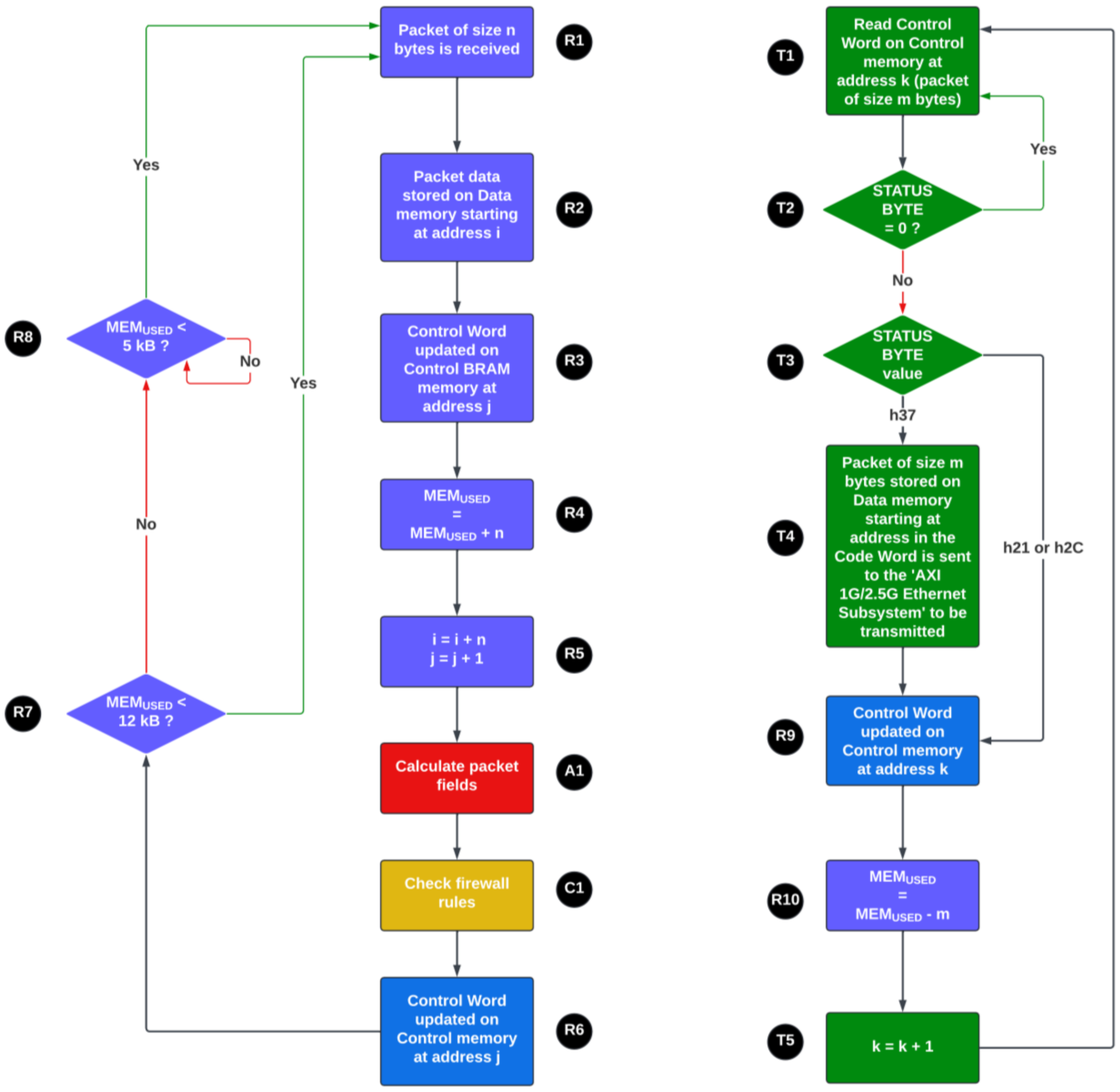
Packet of size n bytes is received
R1
Packet data stored on Data memory starting at address i
R2
Control Word updated on Control BRAM memory at address j
R3
MEM_USED = MEM_USED + n
R4
i = i + n
j = j + 1
R5
Calculate packet fields
A1
Check firewall rules
C1
Control Word updated on Control memory at address j
R6
MEM_USED < 12 kB ?
R7
MEM_USED < 5 kB ?
R8
Yes
No
Yes
No
Read Control Word on Control memory at address k (packet of size m bytes)
T1
STATUS BYTE = 0 ?
T2
Yes
No
STATUS BYTE value
T3
h37
h21 or h2C
Packet of size m bytes stored on Data memory starting at address in the Code Word is sent to the 'AXI 1G/2.5G Ethernet Subsystem' to be transmitted
T4
Control Word updated on Control memory at address k
R9
MEM_USED = MEM_USED - m
R10
k = k + 1
T5

Figure 3. Flow chart of the steps for the packet processing in the firewall. The steps carried out by the 'RX control' module are presented in blue. The steps carried out by the 'Packet analysis' module are presented in red. The steps carried out by the 'Check firewall rules' module are presented in yellow. The steps carried out by the 'TX control' module are presented in green.

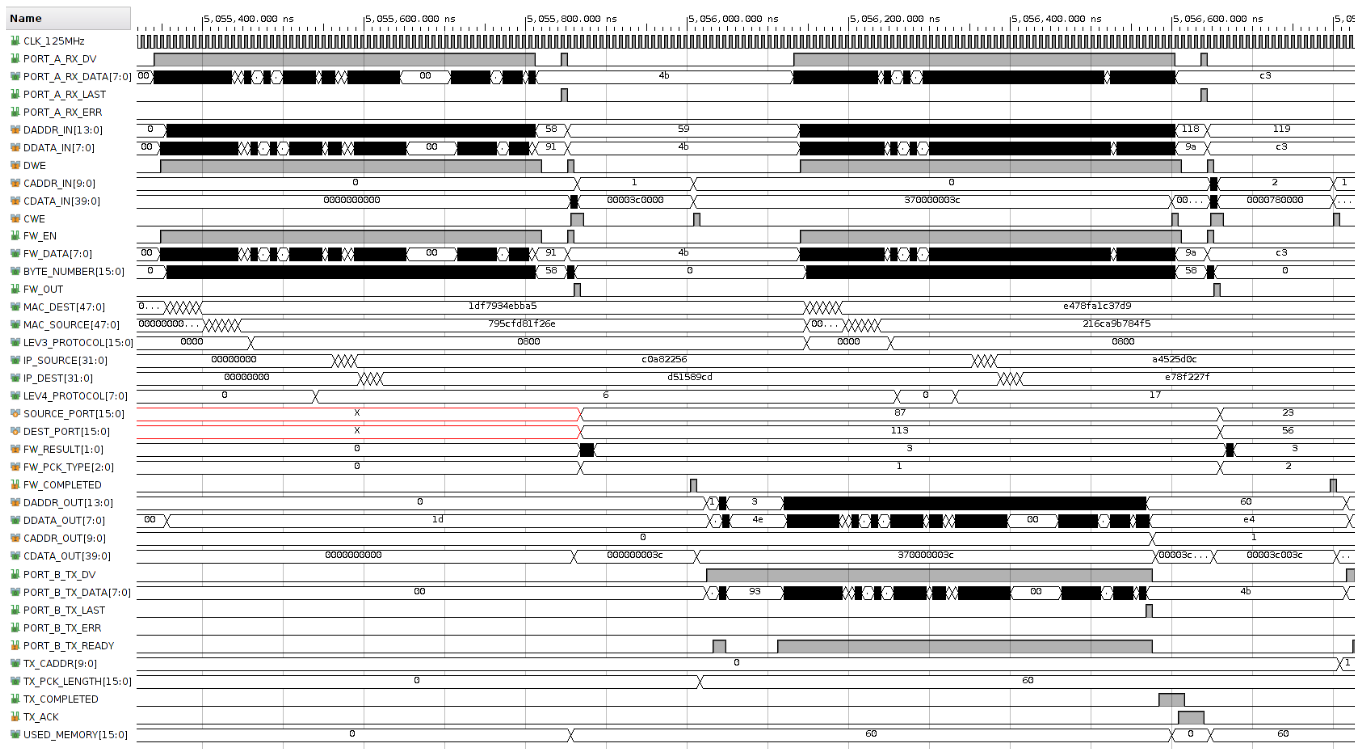

Figure 4. Waveforms of the signals during the processing steps of an Ethernet packet.

The waveforms of the signals obtained with a simulation of the designed firewall hardware are shown in Figure 4. In particular, in the considered time period, two Ethernet frames are received at Port A, which include a TCP packet and a UDP packet. The packet data (PORT_A_RX_DATA[7:0]) are received when the enable signal (PORT_A_RX_DV) is set, and the last byte of the packet is transferred in the clock cycle when PORT_A_RX_LAST is set. The two packets are stored in the Data memory at the addresses (DADDR_IN[13:0]) between 0 and 59 for the first packet, and at the addresses between 60 and 119 for the second packet. The control word for the first packet is stored in the Control memory at the address (CADDR_IN[9:0]) 0, while the second packet control word is stored at Control memory Address 1. The received packets are analyzed and their most important identification fields are determined: the MAC destination address (MAC_DEST[47:0]), the MAC source address (MAC_SOURCE[47:0]), the network layer protocol (LEV3_PROTOCOL[15:0]), the IP source address (IP_SOURCE[31:0]), the IP destination address (IP_DEST[31:0]), the transport layer protocol (LEV4_PROTOCOL[7:0]), the source port (SOURCE_PORT[15:0]) and the destination port (DEST_PORT[15:0]). In particular, the first received packet can be identified as a TCP packet since it is characterized by a network layer protocol of 0800h and a transport layer protocol of 6, while the second received packet can be identified as a UDP packet since it is characterized by a network layer protocol of 0800h and a transport layer protocol of 17. Both received packets are checked against the firewall rules that are loaded in the ‘Firewall rules memory’ during the device initialization. The results of the rules verification steps are valid when the signal FW_COMPLETED is set. In particular, for both packets, FW_RESULT[1:0] = 3, and this means that the packets comply with the firewall rules and can be transmitted to Port B. For both packets, it is also TX_PCK_LENGTH[15:0] = 60 and this indicates that the packet size is 60 bytes. Figure 4 also shows the waveforms of the signals during the transmission of the first packet at Port B. In particular, the first packet is read from the Data memory (addresses DADDR_OUT[13:0] between 0 and 59) and transferred to the ‘AXI 1G/2.5G Ethernet Subsystem’ module of Port B using the 8-bit AXI protocol (PORT_B_TX_DATA[7:0] transferred when PORT_B_TX_DV = 1 with the signal PORT_B_TX_LAST set during the last data transfer). The signal PORT_B_TX_READY is set when the ‘AXI 1G/2.5G Ethernet Subsystem’ module is ready to receive data.

4. Experimental Results

The performance of the proposed firewall was tested by experimental measurements. In Section 4.1, the firewall performance is presented with two packet generators, used to generate controlled network traffic, connected to the two ports of the firewall. In Section 4.2, the firewall performance is presented under real operative conditions when the firewall controls the network traffic between a PC and the Internet. In Section 4.3, preliminary simulations are presented to show the feasibility of upgrading the proposed design to work at a higher data throughput (10 Gb/s).

4.1. Firewall Performance under Controlled Network Traffic

In the first instance, Ethernet data were provided to the firewall using a packet generator, designed on another KC705 FPGA device, which can generate packets of type ARP, TCP, UDP and ICMP of different sizes and with different data throughputs. A detailed description of the packet generator was presented in [17].

The measurement setup is presented in Figure 5. Two packet generators are connected to the two firewall ports implemented with a Quad SFP28 FPGA Mezzanine Card (FMC) Ethernet module [47]. A PC is used to communicate with the two packet generators and the firewall using the USB-UART interface. Initially, the two packet generators are configured to generate packets of a selected protocol, size and data throughput. Then, the firewall statistics (including the number of received/transmitted packets, the level of occupation of data memory, etc.) are acquired at time intervals of 1 s.

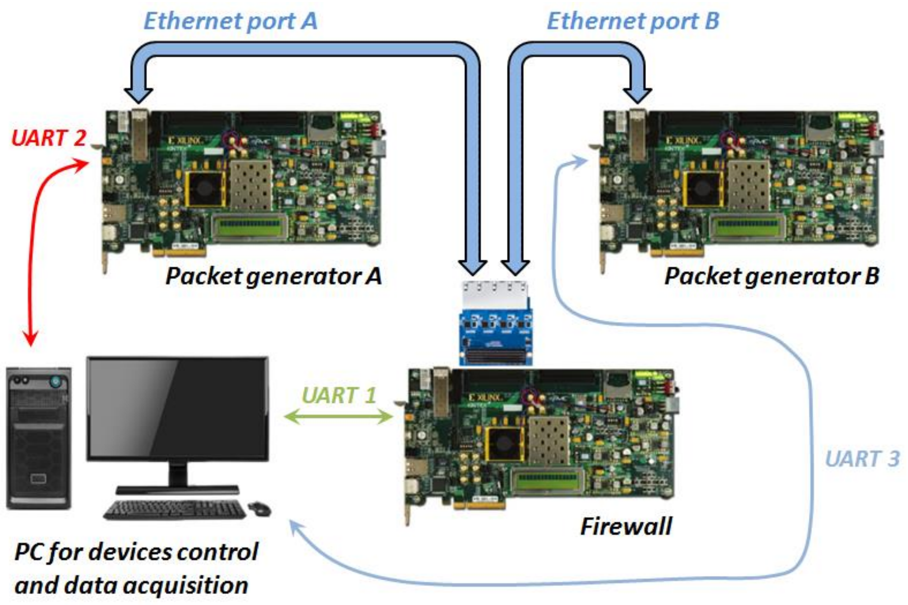

Figure 5. Experimental setup for the measurement of the firewall performance under conditions where the network data are generated by an ad-hoc designed packet generator.

The firewall performance was evaluated in the case of UDP packets. First of all, the packets were continuously generated (i.e., no delay between packets) to evaluate the maximum data throughput for different sizes of the generated packets. Tests were carried out for five different sizes of UDP packet: 100 bytes, 250 bytes, 500 bytes, 750 bytes and 1000 bytes. The results are reported in Figure 6. As can be seen, the data throughput increases for packets of larger size. In the case of packets of 100 bytes in size, the data throughput is 0.811 Gb/s, which is about 80% of the maximum data throughput of 1 Gb/s. In the case of packets of 250, 500, 750 and 1000 bytes in size, the data throughput was measured as 0.917 Gb/s, 0.959 Gb/s, 0.974 Gb/s and 0.982 Gb/s, respectively. The increase in data throughput for packets of larger size can be easily explained by the fact that an Ethernet frame contains a packet that is preceded by a preamble and start frame delimiter

(8 bytes) and ends with a frame check sequence (4 bytes) used to detect data corruption. Thus, in the case of packets of larger size, this overhead has less impact on the data throughput.

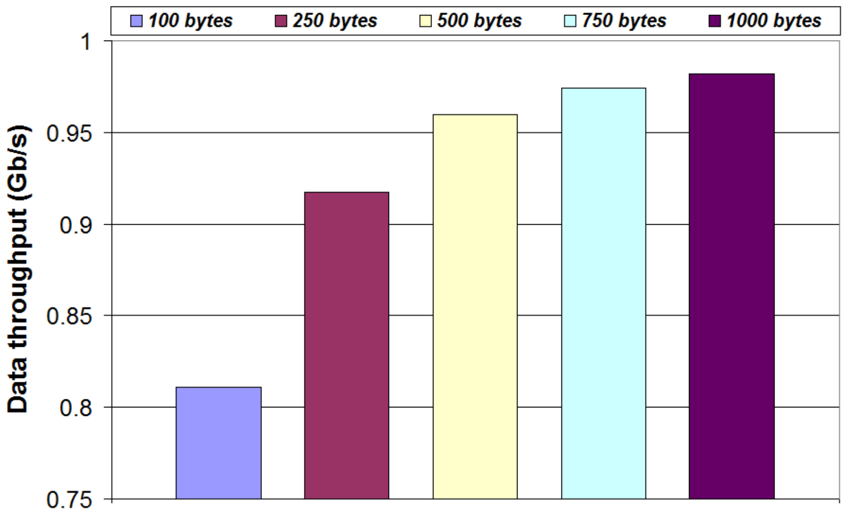

Figure 6. Data throughput of the proposed firewall for UDP packets of different sizes.

As discussed in Section 3, when an Ethernet frame is received at one port of the firewall, its content is stored in the 16 kB data memory before a decision is made if the Ethernet frame must be transmitted or discarded. Thus, the size of the used data memory (i.e., used memory hereafter) was measured at time intervals of 1 s for UDP packets of different sizes and different values of the inter-frame delay. The firewall was designed to receive data as long as the used memory is lower than 12 kB, while, when this threshold is exceeded, the firewall discards the received packets until the used memory decreases below 5 kB. The results showed how, if the inter-frame delay is lower than 13 clock cycles, the used memory increases with time and eventually reaches the critical threshold of 12 kB, while for inter-frame delays of 13 clock cycles or higher, the firewall works continuously without reaching the critical threshold. The measured used memory is plotted vs. time in Figure 7 in the case of UDP packets of a size of 100 bytes and no inter-frame delay.

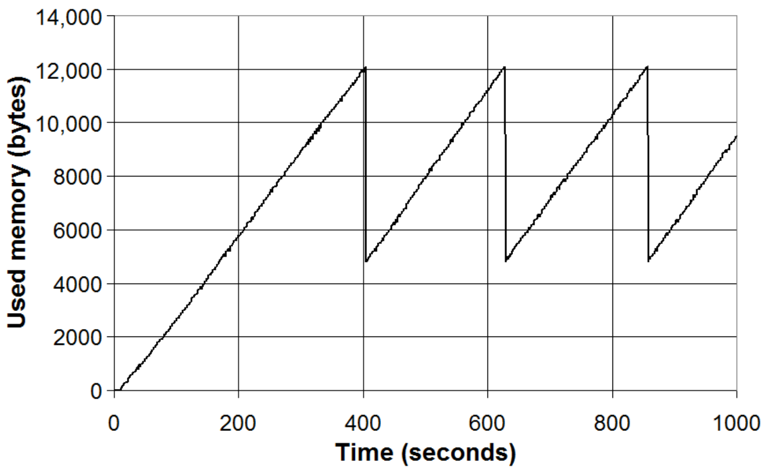

Figure 7. Used data memory vs. time in the case of UDP packets of size 100 bytes and no inter-frame delay.

As can be seen in Figure 7, the used memory increases with a constant rate λ until the critical threshold of 12 kB is reached, then the firewall stops receiving packets until the used memory decreases below 5 kB. The used memory increase rate (λ) was measured for UDP packets of different sizes and different values of the inter-frame delay. The experimental results showed no significant correlation between λ and the inter-frame delay. This can be explained since the 'AXI 1G/2.5G Ethernet Subsystem' IP module introduces a delay of 12 clock cycles to process the Ethernet frame. The measured value of λ as a function of the packet size is reported in Figure 8 for both the average value and the standard deviation.

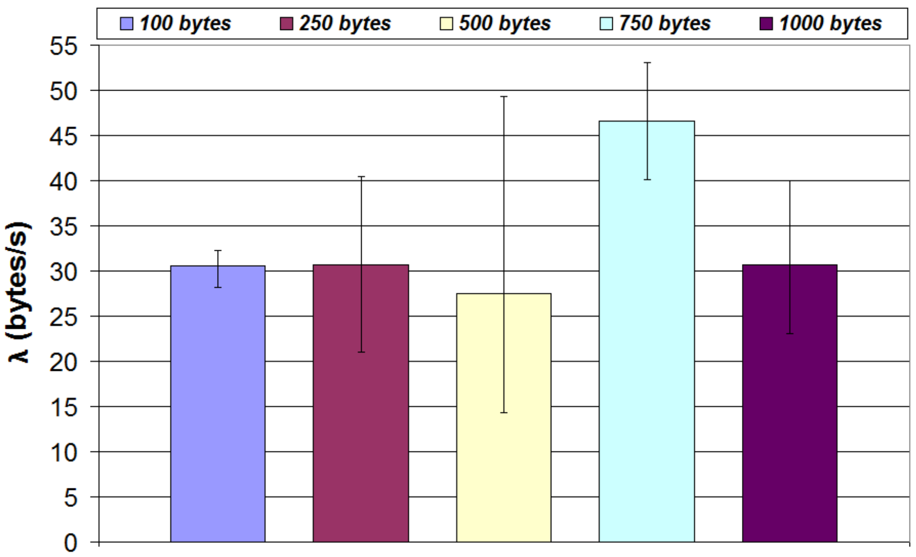

Figure 8. Measured value of the parameter λ in the case of UDP packets of different sizes and no inter-frame delay.

The measured λ resulted in comparable values for the different sizes of packet, with the case of a packet size of 750 bytes producing a slightly higher value. The average value of λ for the different packet sizes was calculated as 33.2 bytes/s. Thus, the used memory increases from 5 kB to 12 kB in an average time of 215.8 s. When the used memory threshold of 12 kB is reached, the firewall stops receiving packets, while the packets in the data memory are transferred until the used memory decreases below 5 kB. At this time, the firewall returns to being functional. Considering the worst-case scenario when all packets in the data memory are transmitted and the firewall data throughput is 0.811 Gb/s (in the case of 100-byte size), the time needed for the used memory to decrease under 5 kB can be estimated as 70.71 µs. This means that the firewall is not operative for a negligible part of the time ($3.29 \cdot 10^{-5}$%).

4.2. Firewall Performance under Real Operative Conditions

After the firewall performance was evaluated in controlled conditions, where Ethernet frames of selected protocol, size and data throughput were generated, the system was tested in real operative conditions. The experimental setup in this case is presented in Figure 9, where the proposed firewall is placed between a PC used to generate real network traffic (Port A) and the Internet (Port B). Another PC was used to acquire the firewall statistics at regular time intervals of 1 s using the USB-UART interface.

Under the new operative conditions, the proposed firewall was tested in the following cases:

1. Download/upload of a file (the results are reported in Section 4.2.1).
2. Access to websites with different restrictions on the firewall rules (the results are reported in Section 4.2.2).

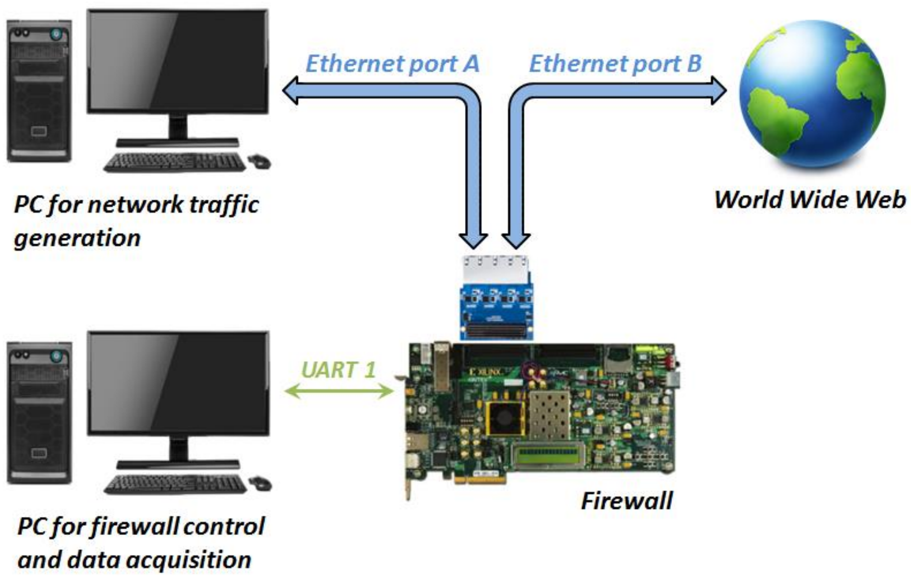
Ethernet port A
Ethernet port B
PC for network traffic generation
World Wide Web
UART 1
Firewall
PC for firewall control and data acquisition

Figure 9. Experimental setup for the measurement of the firewall performance under operative conditions of real network traffic.

In all the reported cases, the network traffic was also monitored with the software Wireshark (version 4.0.3) running on the PC connected to Port A as a reference to verify the correctness of the data provided by the firewall statistics.

4.2.1. Download/Upload of a File

During this first test, the Arduino IDE (file size 197 MB) was downloaded from the official website using the Linux command 'wget'. The file was downloaded four times in sequence by setting different limits for the download bandwidth (2.5 MB/s, 5 MB/s, 10 MB/s, 25 MB/s). The results are reported in Figure 10.

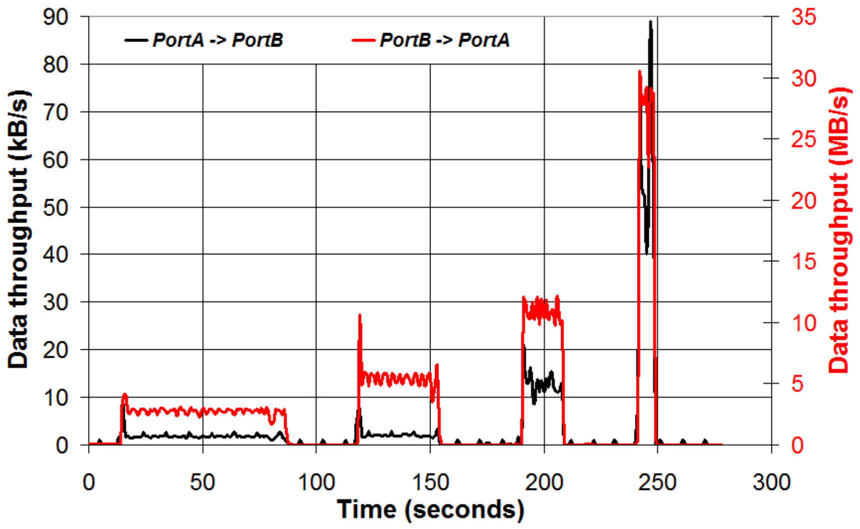

Figure 10. Data throughput measured by the proposed firewall during a file download with the Linux command 'wget' and four different limits for the download bandwidth.

As can be seen, the data throughput from Port A to Port B (i.e., from the PC to the website hosting the file) has values of 1.77 kB/s, 2.23 kB/s, 13.07 kB/s and 58.54 kB/s during the four download operations with a bandwidth limit of 2.5 MB/s, 5 MB/s, 10 MB/s and 25 MB/s, respectively. This can be explained by the fact that, during a download operation, data transferred from the PC to the website hosting the file represent only control packets that are sent to the website to negotiate the file download. The download data are then sent from the website to the PC (from Port B to Port A). In this case, the measured data throughput is 2.76 MB/s, 5.57 MB/s, 10.91 MB/s and 27.9 MB/s during the four download operations with a bandwidth limit of 2.5 MB/s, 5 MB/s, 10 MB/s and 25 MB/s, respectively. The total downloaded data in each of the four download operations are equal to the file size (197 MB).

A second test was carried out using FileZilla, an open-source multi-platform software that can be used to transfer files using the FTP protocol [48]. In this case, the same file of 197 MB in size was first uploaded to a FileZilla server and then downloaded to the PC. The measured data throughput at the two ports of the firewall are plotted vs. time and presented in Figure 11. During the upload phase, lasting 36 s, the average data throughput from the PC to the Filezilla server (i.e., from Port A to Port B) is 5.34 MB/s while the average data throughput from the FileZilla server to the PC (i.e., from Port B to Port A) is 80.88 kB/s. During the download phase, lasting 166 s, the average data throughput from the PC to the Filezilla server (i.e., from Port A to Port B) is 13.75 kB/s while the average data throughput from the FileZilla server to the PC (i.e., from Port B to Port A) is 1.19 MB/s.

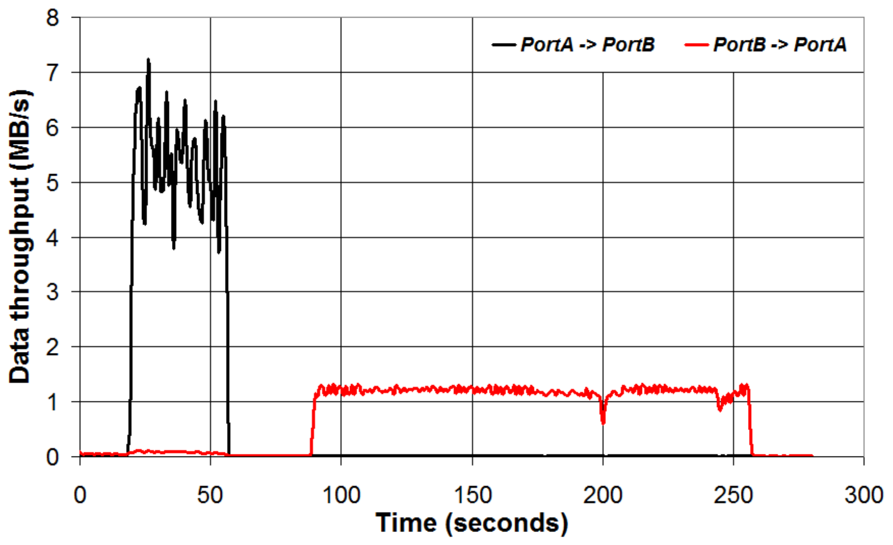

Figure 11. Data throughput measured by the proposed firewall during a file upload/download using the open-source software FileZilla (version 3.66.1).

4.2.2. Access to Websites with Different Restrictions on the Firewall Rules

During this test, the PC tried to access three different websites, characterized by three different IP addresses: 146.75.61.50 for website #1, 13.226.175.71 for website #2 and 88.221.111.115 for website #3. The firewall was programmed with a blacklist approach to block data traffic to the IP address 13.226.175.71 of website #2. Access to website #1 was carried out from 0 to 65 s, access to website #2 was carried out from 75 to 125 s and access to website #3 was carried out from 130 to 185 s. The measured data throughput of TCP and UDP packets are presented in Figure 12 for the network traffic from Port A to Port B (i.e., from the PC to the Internet) and in Figure 13 for the network traffic from Port B to Port A (i.e., from the Internet to the PC).

In the case of websites #1 and #3, which have no restrictions based on the firewall rules, control data are sent from the PC to the website and the website content is transferred back to the PC. These websites were correctly visualized. In the case of website #2, which is present on the firewall blacklist and, thus, must be blocked, some control data are sent to the Port A of the firewall, but these data are blocked and no response from the website is detected. The website #2 was not visualized on the PC.

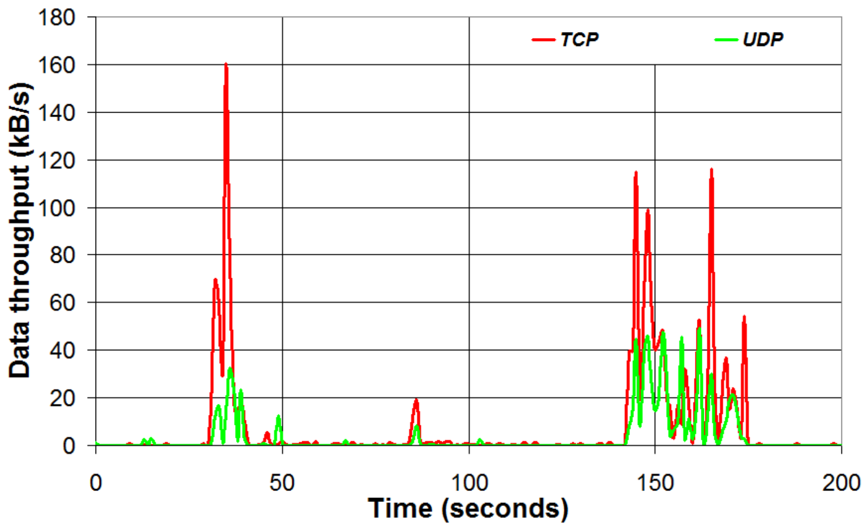

Figure 12. Data throughput of TCP and UDP packets from Port A to Port B during the access to different websites.

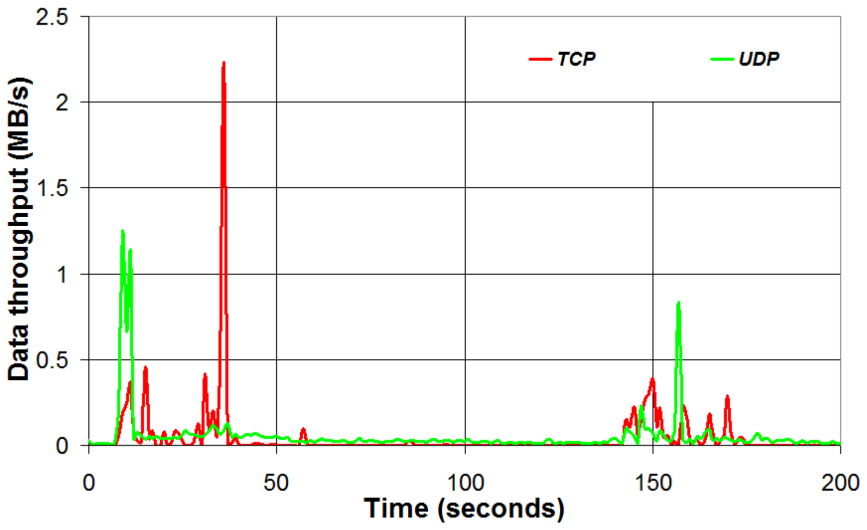

Figure 13. Data throughput of TCP and UDP packets from Port B to Port A during the access to different websites.

4.3. Upgraded Design for Data Throughput of 10 Gb/s

Simulations were carried out to evaluate the feasibility of upgrading the proposed firewall design to work at the higher data throughput of 10 Gb/s. In the case of this higher data throughput, the hardware designed in the FPGA programmable logic was interfaced with the Ethernet SFP/SFP+ connector using the Xilinx IP module 'AXI 10G/25G Ethernet Subsystem' that replaced the 'AXI 1G/2.5G Ethernet Subsystem' module used in the 1 Gb/s firewall. Two different clock signals were generated from the 200 MHz KC705 oscillator using the Xilinx IP 'Clock Wizard': a 156.25 MHz clock used for the circuits devoted to the packets' transmission/reception and analysis (CLK_156.25MHz), and a 10 MHz clock used for the communication with a PC through the USB-UART interface (CLK_10MHz).

The signal waveforms during a simulation are reported in Figure 14, where four different packets (of 60 bytes in size) were received at Port A and processed: a UDP packet, a TCP packet and two ARP packets. The 'AXI 10G/25G Ethernet Subsystem' module transfers the received packet to the designed hardware in the FPGA programmable logic using a 64-bit AXI communication protocol. The 64-bit data bus PORT_A_RX_DATA[63:0] is valid when the signal PORT_A_RX_DV is set. During the transfer of the last 64-bit word of data, the signal PORT_A_RX_LAST is set and the 8-bit control bus PORT_A_RX_KEEP[7:0] defines how many bytes of the 64-bit data bus PORT_A_RX_DATA[63:0] are valid.

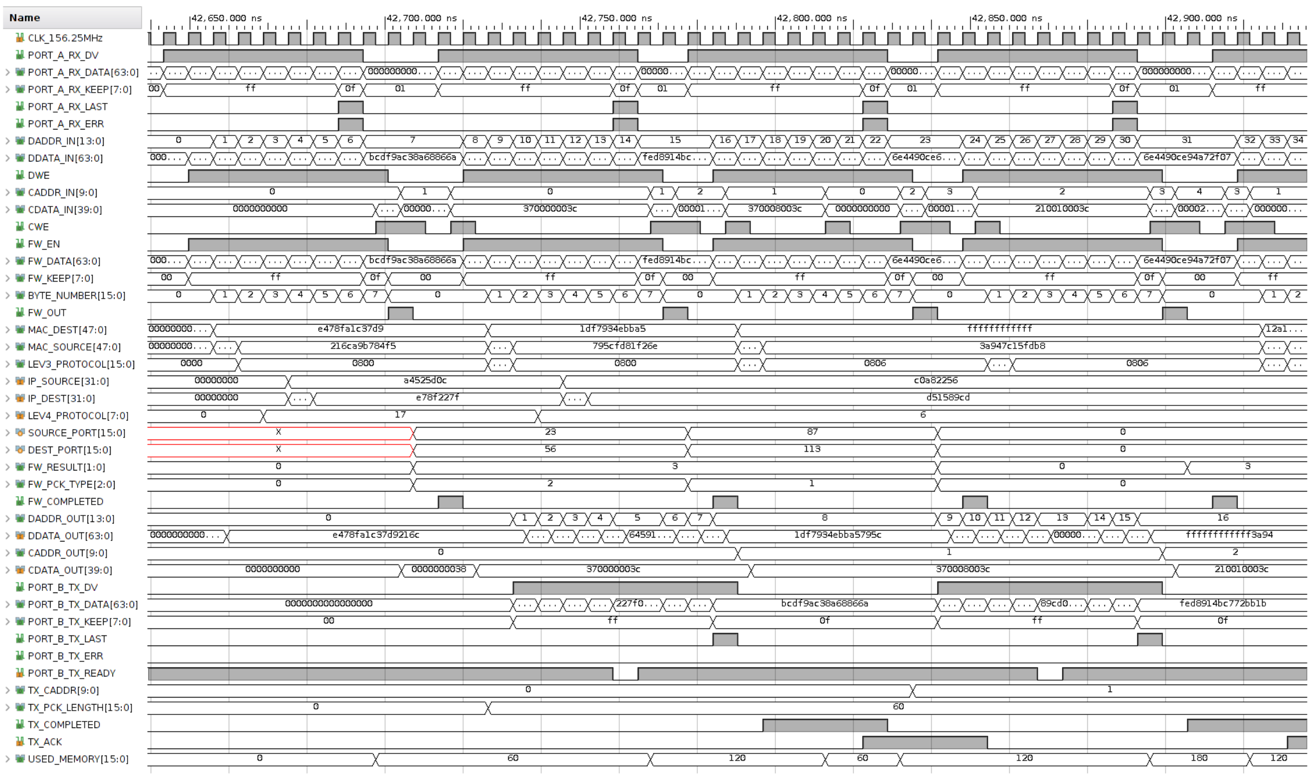

Figure 14. Waveforms of the signals during the packets processing steps in the case of the firewall for the higher data throughput of 10 Gb/s.

The 'Packet analysis' module is implemented using a combinational circuit working on a 64-bit word (FW_DATA[63:0]) for each clock cycle and calculates the same packet identification fields of the 1 Gb/s firewall with a latency of zero clock cycles: the MAC destination address (MAC_DEST[47:0]), the MAC source address (MAC_SOURCE[47:0]), the network layer protocol (LEV3_PROTOCOL[15:0]), the IP source address (IP_SOURCE[31:0]), the IP destination address (IP_DEST[31:0]), the transport layer protocol (LEV4_PROTOCOL[7:0]), the source port (SOURCE_PORT[15:0]) and the destination port (DEST_PORT[15:0]).

When the signal FW_OUT is set, the 'Check firewall rules' module exploits the packet identification fields calculated by the 'Packet analysis' module to check if the received packet complies with the firewall rules. In the case of the 10 Gb/s data throughput, however, a severe constraint is present on the maximum number of clock cycles that can be used to check the compliance of the packet with the firewall rules. Considering a packet size of 64 bytes as a case study, the number of clock cycles available between two consecutive packets is 64 in the case of the 1 Gb/s firewall and 8 in the case of the 10 Gb/s firewall. Thus, the maximum number of rules for each firewall port was decreased from 256 (in the case of the 1 Gb/s firewall) to 32 (in the case of the 10 Gb/s firewall). The packets were checked against the firewall rules, and the results in Figure 14 show how, among the four received packets, the UDP packet, the TCP packet and the second ARP packet complied with the firewall rules (FW_RESULT[1:0] = 3) and were transferred to the 'AXI 10G/25G Ethernet Subsystem' module of Port B to be transmitted, while the first ARP packet presented an error (FW_RESULT[1:0] = 0) and was discarded.

Overall, these preliminary simulations show promising results on the feasibility of upgrading the designed firewall hardware on an FPGA to work at a data throughput of 10 Gb/s. Future activities on this research project will implement the upgraded design on the KC705 development board, and experimental measurements will be carried out to evaluate the system performance.

5. Comparison with the State of the Art

In this section, the features and the performance of the designed firewall hardware on an FPGA are compared with similar systems from the literature. In particular, a comparison is made among the firewalls presented in Section 2 that belong to the same category of stateless firewalls that analyze the packets using a set of rules without storing the status of the network connection. The main features of the considered firewalls are presented in Table 1.

Table 1. Comparison of the performance for the designed firewall with other firewalls from the literature.

FPGA Device	Supported Protocols	Data Throughput	Latency	Reference
Xilinx Kintex-7 325T	ARP, TCP, UDP, ICMP	0.8 Gb/s	402 µs	[28]
Altera EP4CE115F29	ARP, TCP, UDP, ICMP	0.95 Gb/s	61.266 µs	[29]
Xilinx Spartan 6	TCP, UDP (512 bit)	NA	NA	[30]
Altera Stratix II EP2S60F672C5	ARP, TCP, UDP, ICMP	NA	1.76 µs	[31]
Xilinx and Altera devices	NA	NA	NA	[32]
Xilinx Virtex-7 690T	TCP, UDP, TXLAN, GTP	3.67 Gb/s	NA	[33]
Xilinx Virtex-7 690T	ARP, TCP, UDP, ICMP	NA	NA	[34]
Xilinx Kintex-7 325T	ARP, TCP, UDP, ICMP	0.982 Gb/s	627 ns	This work

The proposed design was implemented on a KC705 development board that mounts the Kintex-7 XC7K325T-2FFG900C FPGA. It can classify packets belonging to the protocols ARP, TCP, UDP and ICMP. For each port of the firewall, up to 256 rules can be defined to discriminate between safe and potentially dangerous packets according to a whitelist or blacklist approach. Before a decision is made on the compliance of the packets with the firewall rules, the packet data are stored in a 16 kB memory on the FPGA, and this allows the firewall to be operative almost 100% of the time. A set of statistics, such as the number of received packets, the number of allowed packets, the number of dropped packets and the memory occupation, are calculated in real time and can be downloaded to a PC using the USB-UART interface to detect potential anomalies in the network traffic. Experimental measurements have shown that the designed firewall features a very low latency (627 ns) and a data throughput of 0.982 Gb/s. Preliminary simulations have shown promising results that the design can be upgraded to work at a maximum data throughput of 10 Gb/s with minor modifications.

Regarding the supported network protocols, the proposed firewall can classify packets that belong to the protocols ARP, TCP, UDP and ICMP, just like most of the compared systems [28,29,31,34]. The system presented in Ref. [30] can only analyze packets belonging to the protocols TCP and UDP, and the experiments were only carried out on packets of a fixed size (512-bit). The performance of the system presented in Ref. [32] is not reported since this firewall was evaluated only by simulations and was not implemented on an FPGA device. The firewall presented in Ref. [33] was designed for the detection of cyber-attacks in 5G networks and thus also implements additional protocols for this type of network (TXLAN and GTP).

Regarding the data throughput, the firewall presented in Ref. [33] achieves the best performance (3.67 Gb/s), while all the other systems presented in Table 1 feature a data throughput lower than the proposed firewall (0.982 Gb/s). However, based on the promising simulations presented in Section 4.3, we are confident that our system can be improved with a higher data throughput. Moreover, the firewall of Ref. [33] is characterized by a packet loss rate of about 2.5%, while our system is not operative for a negligible part of the time ($3.29 \cdot 10^{-5}$%).

Regarding the packet latency, the proposed firewall features the lowest value (627 ns) among the systems presented in Table 1.

Overall, the presented firewall hardware on an FPGA features a very good performance that makes it suitable as a configurable intrusion prevention system to guarantee the security of exchanged data among universities and research centers. Nevertheless, there is room for improvement, and the next steps in this research line will be aimed at improving the system's performance. In particular, the simulated system for the higher data throughput of 10 Gb/s will be implemented on an FPGA and its performance will be evaluated in a real network scenario. Moreover, the network data analysis will be improved by introducing stateful packet inspection by storing the status of the network connection and/or by the use of machine-learning algorithms.

6. Conclusions

A hardware firewall implemented on an FPGA for the security of network data transmission using the Ethernet protocol was presented in this work. The firewall was experimentally validated in the case of a data throughput of 1 Gb/s, and preliminary simulations have shown that the proposed design can be upgraded with minor modifications to work at a data throughput of 10 Gb/s. The device can process network packets of the types ARP, UDP, TCP and ICMP with a latency of 627 ns and a data throughput in the range from 0.811 Gb/s to 0.982 Gb/s for a packet size in the range from 100 bytes to 1000 bytes. It is user configurable using a whitelist or blacklist approach, with up to 256 rules for each port that are used to discriminate between safe and potentially dangerous data packets. A set of statistics, such as the number of received/transmitted packets and the amount of received/transmitted data, is calculated and can be used to detect possible anomalies in the network traffic. Overall, the proposed firewall based on an FPGA is a cost-effective solution that is highly competitive with hardware firewalls based on ASICs and can be used to mitigate security threats of malicious data within the network, yielding users full observability of the device behavior. Future steps of this research project will be the implementation of the upgraded 10 Gb/s firewall design on an FPGA to test its performance under real operative conditions and/or the implementation of a stateful packet inspection.

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. All authors have read and agreed to the published version of the manuscript.

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).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

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 hardware firewall on FPGA.

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

References

1. Khan, R.A. A survey on wired and wireless network. Lahore Garrison Univ. *Res. J. Comput. Sci. Inf. Technol.* **2018**, *2*, 19–28.
2. Narayanasamy, S.K.; Srinivasan, K.; Hu, Y.C.; Masilamani, S.K.; Huang, K.Y. A contemporary review on utilizing semantic web technologies in healthcare, virtual communities, and ontology-based information processing systems. *Electronics* **2022**, *11*, 453. [CrossRef]
3. Paiva, S.; Ahad, M.A.; Tripathi, G.; Feroz, N.; Casalino, G. Enabling technologies for urban smart mobility: Recent trends, opportunities and challenges. *Sensors* **2021**, *21*, 2143. [CrossRef]

4. Rahman, S.S.; Dekkati, S. Revolutionizing Commerce: The Dynamics and Future of E-Commerce Web Applications. *Asian J. Appl. Sci. Eng.* **2022**, *11*, 65–73. [CrossRef]
5. Camacho, D.; Panizo-LLedot, A.; Bello-Orgaz, G.; Gonzalez-Pardo, A.; Cambria, E. The four dimensions of social network analysis: An overview of research methods, applications, and software tools. *Inf. Fusion* **2020**, *63*, 88–120. [CrossRef]
6. Ryalat, M.; ElMoaqet, H.; AlFaouri, M. Design of a smart factory based on cyber-physical systems and Internet of Things towards Industry 4.0. *Appl. Sci.* **2023**, *13*, 2156. [CrossRef]
7. Ramasamy, L.K.; Khan, F.; Shah, M.; Prasad, B.V.V.S.; Iwendi, C.; Biamba, C. Secure smart wearable computing through artificial intelligence-enabled internet of things and cyber-physical systems for health monitoring. *Sensors* **2022**, *22*, 1076. [CrossRef]
8. Volosciuc, C.; Bogdan, R.; Blajovan, B.; Stângaciu, C.; Marcu, M. GreenLab, an IoT-Based Small-Scale Smart Greenhouse. *Future Internet* **2024**, *16*, 195. [CrossRef]
9. Saminathan, K.; Mulka, S.T.R.; Damodharan, S.; Maheswar, R.; Lorincz, J. An Artificial Neural Network Autoencoder for Insider Cyber Security Threat Detection. *Future Internet* **2023**, *15*, 373. [CrossRef]
10. Li, Y.; Liu, Q. A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. *Energy Rep.* **2021**, *7*, 8176–8186. [CrossRef]
11. Dang, T.K.; Nguyen, K.D.; Kieu-Do-Nguyen, B.; Hoang, T.T.; Pham, C.K. Realization of Authenticated One-Pass Key Establishment on RISC-V Micro-Controller for IoT Applications. *Future Internet* **2024**, *16*, 157. [CrossRef]
12. Dong, S.; Su, H.; Xia, Y.; Zhu, F.; Hu, X.; Wang, B. A comprehensive survey on authentication and attack detection schemes that threaten it in vehicular ad-hoc networks. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 13573–13602. [CrossRef]
13. Subramani, S.; Svn, S.K. Review of security methods based on classical cryptography and quantum cryptography. *Cybern. Syst.* **2023**, 1–19. [CrossRef]
14. Thabit, F.; Can, O.; Aljahdali, A.O.; Al-Gaphari, G.H.; Alkhzaimi, H.A. Cryptography algorithms for enhancing IoT security. *Internet Things* **2023**, *22*, 100759. [CrossRef]
15. Chakir, O.; Sadqi, Y.; Maleh, Y. Evaluation of open-source web application firewalls for cyber threat intelligence. In *Big Data Analytics and Intelligent Systems for Cyber Threat Intelligence*; River Publishers: Aalborg, Denmark, 2023; pp. 35–48.
16. Dawadi, B.R.; Adhikari, B.; Srivastava, D.K. Deep learning technique-enabled web application firewall for the detection of web attacks. *Sensors* **2023**, *23*, 2073. [CrossRef]
17. Grossi, M.; Alfonsi, F.; Prandini, M.; Gabrielli, A. A Highly Configurable Packet Sniffer Based on Field-Programmable Gate Arrays for Network Security Applications. *Electronics* **2023**, *12*, 4412. [CrossRef]
18. Grossi, M.; Alfonsi, F.; Prandini, M.; Gabrielli, A. A high throughput Intrusion Detection System (IDS) to enhance the security of data transmission among research centers. *J. Instrum.* **2023**, *18*, C12017. [CrossRef]
19. Lata, K.; Cenkeramaddi, L.R. FPGA-Based PUF Designs: A Comprehensive Review and Comparative Analysis. *Cryptography* **2023**, *7*, 55. [CrossRef]
20. Serrano, R.; Duran, C.; Sarmiento, M.; Dang, T.K.; Hoang, T.T.; Pham, C.K. A Unified PUF and Crypto Core Exploiting the Metastability in Latches. *Future Internet* **2022**, *14*, 298. [CrossRef]
21. Grossi, M.; Omaña, M.; Rossi, D.; Marzulli, B.; Metra, C. Novel BTI Robust Ring-Oscillator-Based Physically Unclonable Function. In Proceedings of the IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS), Torino, Italy, 12–14 September 2022; pp. 1–7.
22. Mihalos, M.G.; Nalmpantis, S.I.; Ovaliadis, K. Design and Implementation of Firewall Security Policies using Linux Iptables. *J. Eng. Sci. Technol. Rev.* **2019**, *12*, 80–86. [CrossRef]
23. Wang, B.; Lu, K.; Chang, P. Design and implementation of Linux firewall based on the frame of Netfilter/IPtable. In Proceedings of the IEEE 11th International Conference on Computer Science & Education (ICCSE), Nagoya, Japan, 23–25 August 2016; pp. 949–953.
24. Šimon, M.; Huraj, L.; Čerňanský, M. Performance evaluations of IPTables firewall solutions under DDoS attacks. *J. Appl. Math. Stat. Inform.* **2015**, *11*, 35–45. [CrossRef]
25. Wireshark Packet Sniffer. Available online: https://www.wireshark.org/ (accessed on 19 June 2024).
26. TCPdump Packet Sniffer. Available online: https://www.tcpdump.org/ (accessed on 19 June 2024).
27. Li, J.; Xiao, W.; Zhang, C. Data security crisis in universities: Identification of key factors affecting data breach incidents. *Humanit. Soc. Sci. Commun.* **2023**, *10*, 270. [CrossRef] [PubMed]
28. Mohammed, R.K.; Ueno, Y. An FPGA-based Network Firewall with Expandable Rule Description. *Indones. J. Electr. Eng. Comput. Sci.* **2018**, *10*, 1310–1318. [CrossRef]
29. Lin, S.; Zhang, D.; Fu, Y.; Wang, S. A design of the ethernet firewall based on FPGA. In Proceedings of the IEEE 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, China, 14–16 October 2017; pp. 1–5.
30. Keni, S.M.; Mande, S. Design and implementation of hardware firewall using FPGA. In Proceedings of the IEEE 3rd International Conference for Convergence in Technology (I2CT), Pune, India, 6–8 April 2018; pp. 1–4.
31. Ajami, R.; Dinh, A. Design a hardware network firewall on FPGA. In Proceedings of the IEEE 24th Canadian Conference on Electrical and Computer Engineering (CCECE), Niagara Falls, ON, Canada, 8–11 May 2011; pp. 000674–000678.

32. Antonov, A.P.; Filippov, A.S.; Mamoutova, O.V. Next generation FPGA-based platform for network security. In Proceedings of the IEEE 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT), St. Petersburg, Russia, 18–22 April 2016; pp. 9–14.
33. Ricart-Sanchez, R.; Malagon, P.; Alcaraz-Calero, J.M.; Wang, Q. NetFPGA-based firewall solution for 5G multi-tenant architectures. In Proceedings of the IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 132–136.
34. Salopek, D.; Mikuc, M. Enhancing Mitigation of Volumetric DDoS Attacks: A Hybrid FPGA/Software Filtering Datapath. *Sensors* **2023**, *23*, 7636. [CrossRef]
35. Bianchi, G.; Bonola, M.; Pontarelli, S.; Sanvito, D.; Capone, A.; Cascone, C. Open Packet Processor: A programmable architecture for wire speed platform-independent stateful in-network processing. *arXiv* **2016**, arXiv:1605.01977.
36. Pontarelli, S.; Bifulco, R.; Bonola, M.; Cascone, C.; Spaziani, M.; Bruschi, V.; Sanvito, D.; Siracusano, G.; Capone, A.; Honda, M.; et al. FlowBlaze: Stateful Packet Processing in Hardware. In Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), Boston, MA, USA, 26–28 February 2019; pp. 531–548.
37. Tran, C.; Vo, T.N.; Thinh, T.N. HA-IDS: A heterogeneous anomaly-based intrusion detection system. In Proceedings of the IEEE 4th NAFOSTED Conference on Information and Computer Science, Hanoi, Vietnam, 24–25 November 2017; pp. 156–161.
38. Le Jeune, L.; Goedemé, T.; Mentens, N. Towards real-time deep learning-based network intrusion detection on FPGA. In *Applied Cryptography and Network Security Workshops: ACNS 2021 Satellite Workshops, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, and SiMLA, Kamakura, Japan, 21–24 June 2021*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 133–150.
39. Murovič, T.; Trost, A. Genetically optimized massively parallel binary neural networks for intrusion detection systems. *Comput. Commun.* **2021**, *179*, 1–10. [CrossRef]
40. Lázaro, J.; Bidarte, U.; Muguira, L.; Astarloa, A.; Jiménez, J. Embedded firewall for on-chip bus transactions. *Comput. Electr. Eng.* **2022**, *98*, 107707. [CrossRef]
41. Restuccia, F.; Kastner, R. Cut and forward: Safe and secure communication for FPGA system on chips. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *41*, 4052–4063. [CrossRef]
42. Verilog HDL Tutorial. Available online: https://www.chipverify.com/tutorials/verilog (accessed on 9 August 2024).
43. Xilinx KC705 Development Board. Available online: https://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html (accessed on 19 June 2024).
44. LabVIEW Graphical Programming Language. Available online: https://www.ni.com/en/shop/labview.html (accessed on 9 August 2024).
45. AXI 1G/2.5G Ethernet Subsystem v7.2 Product Guide. Available online: https://docs.amd.com/viewer/book-attachment/GVuCppHToFb1WA89zYBnLA/h3XUJnOY_QWIild5x9SAeQ (accessed on 9 August 2024).
46. Block Memory Generator. Available online: https://www.xilinx.com/products/intellectual-property/block_memory_generator.html (accessed on 19 June 2024).
47. Quad SFP28 FPGA Mezzanine Card (FMC) Ethernet Module. Available online: https://hiteksys.com/interface-modules/x4-sfp-fmc-module (accessed on 19 June 2024).
48. FileZilla Software. Available online: https://filezilla-project.org/ (accessed on 19 June 2024).