THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

FPGA packet filter with Ethernet MAC and web server using a RISC-V softcore processor

Project Proposal

Matthew Gilpin

45801600

Semester 1, 2023

The University of Queensland

School of Information Technology and Electrical Engineering

List of Abbreviations

Abbreviations	
IoT	Internet of Things
FPGA	Field Programmable Gate Array
pf	Packet Filter
MAC	Medium Access Control
ISA	Instruction Set Architecture
ASIC	Application Specific Integrated Circuit
SoC	System on Chip
TRL	Technology Readiness Level
IP	Intellectual Property
PHY	Physical layer
RMII	Reduced Media Independant Interface
FIFO	First-In First-Out
LSB	Least Significant Bit
FSM	Finite State Machine
CLI	Command Line Interface
GUI	Graphical User Interface
RTOS	Real Time Operating System

Contents

List of Figures

List of Tables

Introduction

1.1 Motivation

In a technology age of growing numbers of cyber attacks and record number of connected devices, it's paramount to ensure these devices operate safely and securely. The Australian Cyber Security Center (ACSC) received in excess of 76,000 cybercrime reports and growing in the 2021-22 financial year [2]. The growing trend of Internet of Things (IoT) will provide more opportunity for black hats (malicious attackers). IHS Markit estimates 125 billion IoT devices will be connected by 2030 [3].

To cope with the increase in IoT devices, a common shift to edge computing has evolved in favour over the traditionally more centralised cloud computing architecture. Edge computing as [4] puts it, the paradigm envolves the computation and analysis of data at the *edge* of the network to be as close as possible to the source of the data. This has many advantages including: lower latency, bandwidth requirements, availability, energy, security and privacy [4]. Due to the distributed load, smaller and more efficient computers can be used at the edge/perimeter of these networks [5]. Just like any other computer connected to the broader network, these edge networks also need to be protected from bad actors.

By using field programmable gate arrays (FPGAs), custom hardware could be designed to incentivise low latency, high throughput yet efficient wirespeed firewalls. This report proposes a project that attempts to meet these design criteria.

Literature review

Some of the concepts behind the proposed project, such as an Ethernet MAC or RISC-V processor are not new, there is a variety of previous works in these areas. This part of the proposal will explore the prior work related to the project.

2.1 Packet Filter Firewall

Usually, the first line of defence against bad actors, it is firewalls play a vital component in computer networks and can become vastly complex. In essence, the job of a firewall is to isolate and restrict acess to an internal network from an external one [6].

There are several types of Firewalls such as packet filters (PF), stateful packet firewalls and application firewalls [7]. PFs are considered as stateless and traditionally filter exlcusively on the fields in the network (layer 2) and transport (layer 3) layer headers [7]. Such fields include IP addresses, port numbers and protocol type.

Due to this, PFs are inherently simple and efficient. Consequently, they are widely available and can be either implemented in software or in hardware [6]. The book, [6], also highlights some inherent flaws with PFs. These include not being able to suprress a sophisticated attacks and in some cases can be challenging to properly configure. More advanced firewalls can perform deep packet inspection and explore the contents of the higher layers to better evaluate a packets true intention [7].

2.2 Field Programmable Gate Arrays

First introduced by Xilinx in 1984, field programmable gate arrays (FPGAs) allowed for large custom logic designs to be recognised without the need for expensive application specific integrated circuits (ASICs). More importantly, FPGAs did not suffer from the same scalability issues that programmable array logic (PAL) encountered and has allowed for larger and more complex designs [8].

A big advantage to custom logic is the ability to create highly parallelised designs while also with a lower latency than software based serialised algorithms. As such, FPGAs became ubiquitous in both digital signal processing and for accelerating an assortment of computing architectures and processes [9]. System on chip (SoC) design with custom hardware acceleration modules is an active area research. As [9] points out, there is a focus towards using both hardware and software in *edge* devices due to growing numbers of IoT devices.

Several papers, [10] [11] [12], have proposed a range of related FPGA based firewalls, all having slightly diffrent properties and using different classificiation mechanisms. Each of the propsoed firewalls in the afformentioned papers use custom hardware to first detect and then handle the incomming network packet. The key benefit to these firewalls is their high performance - namely, low latency,

and high throughput. Article [10] proposed an ethernet firwall using LwIP and five-tuple binding to achieve a throughput of 950Mbps with a latency of 61.266us. A conference proceeding in 2000 [11] which used a comparator unit to check the fields of the IP header obtained a filtering rate of 500,000 packets per second.

The enabling concept behind the above FPGA based firewalls is SoC design which involves integrating multiple components into a single package, or in this case a single FPGA. Often these will include small softcore microprocessors and some custom ethernet hardware like the proposed packet filters in [10]. Softcore processors are configurable and can be modeled in a hardware description language (HDL) which can then be synthesised onto ASICs or FPGAs hardware [13]. Recently, the royalty free RISC-V based cores are a popular softcore architecture used on SoC design.

2.3 RISC-V processor

In the world of processor architectures, there are four major families, namely AMD64, x86, ARM and RISC-V. The two former instruction set architectures (ISA) are appart of the complex instructions sets (CISC) and are found in the majority of computers. ARM and RISC-V are a reduced version of the CISC family and subsequently fall under the RISC family and are ideal for low power microprocessors [14].

RISC-V is an open and royalty free ISA and as a result, a plethora of softcore based custom implementations have been designed [15]. Consequently, there is an abundance of articles dealving into RISC-V from evaluating the ISA [16] to creating multicore architectures [17]. A 2019 paper, [15] evaluated a variety of different RISC-V softcore processors. RISC-V International have also published a list[1] of different RISC-V implementations that have a unique architecture ID. The majority of these are either written in a HDL for either application specific integrated circuits (ASICs) or FPGAs. The *NEORV32 RISC-V* softcore processor is written purely in vendor agnostic VHDL and importantly has a considerable amount of documentation.

Being a softcore processor, control is given over which modules are implemented. Some basic features of the *NEORV32 RISC-V* include UART, SPI, and GPIO interfaces [18]. The datasheet, [18], also mentions that it supports a *'Wishbone b4'* external bus interface. A Wishbone B4 (or just 'wishbone') interconnection is designed specifically to connect modular pieces of hardware together on a SoC into the memory mapped 32bit address space in the processor [19]. This approach has the benefit of not neededing to create custom instructions for the microprocessor.

[1] See: https://github.com/riscv/riscv-isa-manual/blob/master/marchid.md

2.4 Ethernet MAC

First introduced in 1983, the IEEE 802.3 standard [1], more commonly known by the name of 'Ethernet', defines the *'Medium Access Control'* (MAC) protocol amongst other things for two or more devices to communicate over a network. This standard is just one part in the layered network models such as the OSI model or TCP/IP model, namely the network layer - layer 2.

A core function of the Ethernet MAC is to attach the required MAC layer headers to the head and tail of the layer 3 payload to create an ethernet packet. The fields in an ethernet packet can be seen in figure 2.1.

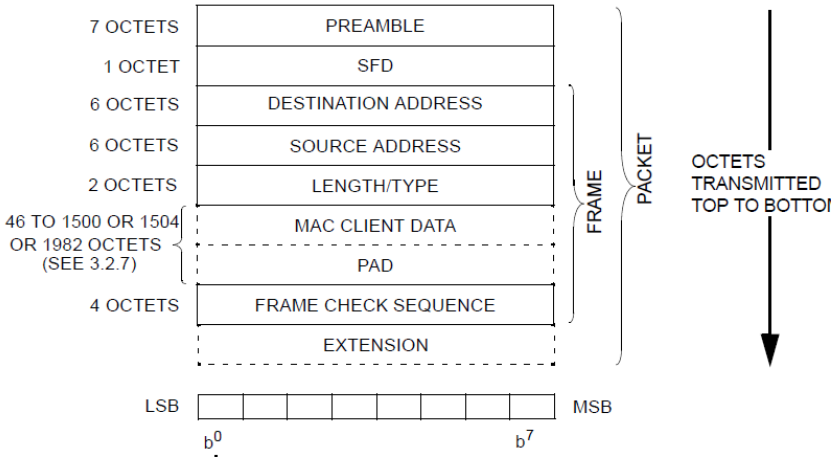

Figure 2.1: MAC layer headers [1]

After the packet has been constructed, the data is forwarded out to the physical (PHY) layer least significant bit (LSB) first [1]. Typically a PHY management chip is used to handle the physical layer channel encoding amougst other things. These PHY chips often can be interfaced with the media independant interfaces such as MII, RMII, GMII and RGMII [20]. The reduced media independant interface (RMII) is one of these standards defined in [1] and consists of a reference clock, 2 bit wide transmit (TX), 2 bit wide recieve (RX) lines and a few other supplimentary signals as defined in the LAN8720A datasheet [21].

Numerous articles [20] [22] [23] can be found about ethernet MACs implemented on FPGAs each with a slightly different approach. Fundamentally though, as best highlighted in [20], a simple way of implementing a MAC is by employing a finite state machine (FSM). Another technique found in these articles is the use of first-in first-out (FIFO) buffers to cross clock domains. This is a common technique used in FPGA design as it allows you to have the packet asssembly logic at a much higher clock rate than the output RMII reference clock speed [22].

In addition to the papers, there are a plethora of intellectual property (IP) blocks for xMII interfaces in HDL which have their own benefits and drawbacks. Some freely available HDL modules for ethernet MACs can be found in both a complete [1] [2] [3] and uncomplete state [4].

2.5 Web servers and network stacks

Almost all firewalls need to be configured with a ruleset which can be configured in two common ways, using a command line interface (CLI) or by a Web graphical user interface (GUI). Before a web server can be realised, the network stack (Layers 3, and 4) need to be established since a web server operates at the application layer (layer 4). As embedded platforms are resource limited, special precautions need to be taken when it comes to memory and resource useage [24].

Article [10] investigated using the open source lightweight IP (LwIP) network stack as a mechanism for interfacing with the firewall. The LwIP library is a popular lightweight TCP/IP stack which has been investigated in a plethora of research papers and projects [25] [24]. Often these papers run LwIP on real time operating systems (RTOS) such as FreeRTOS or Zephyr.

FreeRTOS is a leading RTOS for microprocessors and is distributed freely under the MIT license. As an RTOS, it provides an abstraction to the hardware that allows for multitasking and brings other OS-Like features to embedded systems. Several ports are available including one for RISC-V.

FreeRTOS also provide their own TCP/IP network stack called *FreeRTOS-Plus-TCP* which includes a HTTP web server example and is much newer than LwIP. Consequently less research can be found appart from existing documentation. The library aims to provide a threadsafe Berkley sockets API and network stack supporting multiple protocols such as DHCP, DNS, TCP, and UDP [26]. LwIP is not threadsafe and typically suffers from memory issues as found in [24].

[1] See: https://github.com/yol/ethernet_mac
[2] See: https://github.com/alexforencich/verilog-ethernet/
[3] See: https://opencores.org/projects/ethernet_tri_mode
[4] See: https://github.com/pabennett/ethernet_mac

Topic Definition

3.1 Topic

While there is no single solution that will fully protect an edge network, a common and effective way to reduce the unauthorised/unwanted network traffic is by simply filtering out the potentially malicious packets. While this may seem overly complex, in reality a few simple rules can be followed to decide on whether to forward or deny/drop/block packets from entering or exiting a network. These packet filters (pf) are a type of firewall that do not follow any complex rules and keep state between packets or use deep packet inspection to check the contents of the payload to ensure it's not malicious.

The proposed project consists of making a hardware implementation of a pf with custom Ethernet Media Access Controllers (MAC) connected to a hardware based filtering block which is all controlled by a RISC-V softcore processor. This will then also have a web interface so that a user can configure the rules for the pf.

3.2 System Overview

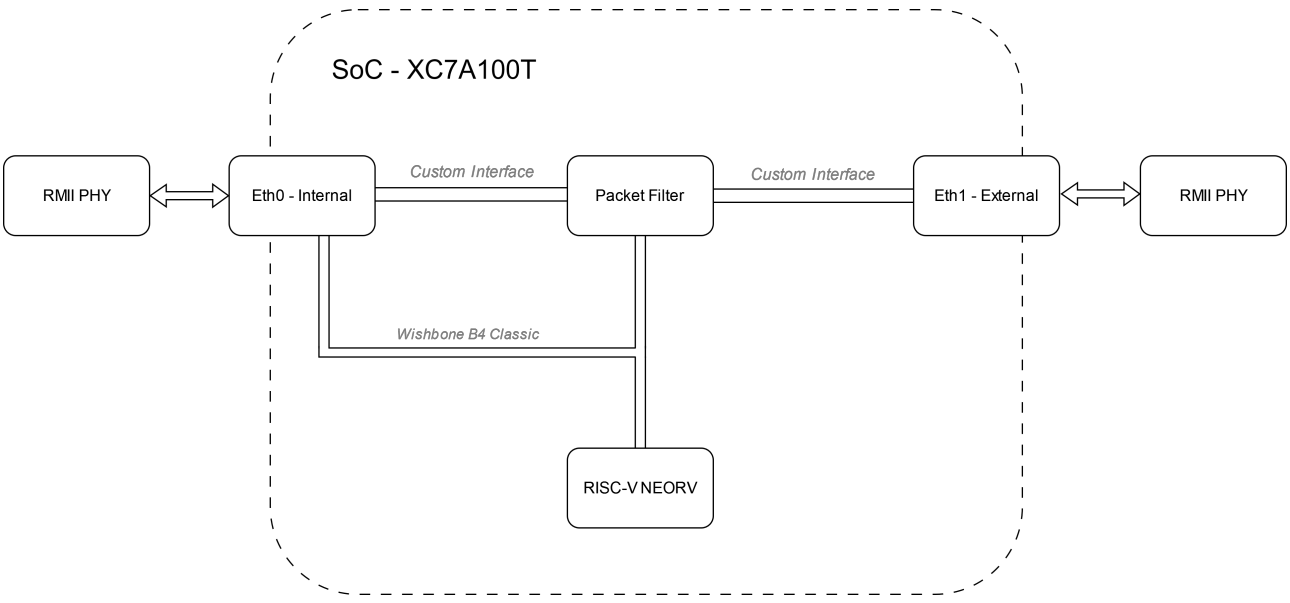

Figure 3.1: System overview

3.3 Aims

The aims of the proposed FPGA Ethernet controller and web interface on a RISC-V processor are:

- Increase security to edge IoT networks,
- Increase the power efficiency for wire-speed firewalls, and
- Decrease the latency for packet filter firewalls.

3.4 Establishing Exclusions

While the proposed project will reduce the likelihood of network based attacks it is not a *'one size fits all'* solution. By the nature of the IoT and edge network ecosystem, there are a myriad of different attack vectors where not all of them will be detectable at the network level.

The proposed project will **not**

- Protect against all attacks,
- Be able to protect against all IoT devices,
- Perform routing,
- IPv6 Packets, or
- Perform network address translation (NAT)

3.5 Performance Indicators

3.6 Required Equiptment

While the hardware design will be developed in such a way that it is vendor agnostic, to test the design a Digilent Nexys A7-100T developement board will be used. Importantly, this board has a RJ45 connector and LAN8720 RMII interface chip which allows for a regular fast ethernet connection to be directly connected to the FPGA board. An additional LAN8720 ETH board from Waveshare is also required to obtain the secondary interface.

To validate the functionality and effectiveness of the design, it will be compared with a Raspberry Pi Compute Module 4 (CM4) with a Waveshare CM4-DUAL-ETH-MINI daughterboard which contains two 1GbE interfaces. This will act as a baseline.

3.7 Technology Readiness Level

One method for estimating the degree of maturity for a technical project is by using the *Technology Readiness Levels (TRL)* benchmark. Within this, there are 9 levels each indicating a different phases of a design. This project is intended to reach a TRL of 6. These six different levels and their relevance to this project are as follows.

1. **TRL 1: Basic Research** - The concepts are researched and a base understanding of the system is gathered,
2. **TRL 2: Applied Research** - Detailed research is conducted into each part of the project,
3. **TRL 3: Proof of Concept** - A cutdown version of the final project protoype that highlights the core functionality or subsystems working,
4. **TRL 4: Lab Testing of Prototype** - Prototype of the core design with majority of the functionality working,
5. **TRL 5: Testing of Integrated system** - Refined prototype that works as intended but may be incomplete, and
6. **TRL 6: Prototype System Verified** - Comparison to pre-existing solutions and verification.

Timeline and Plan

This section of the report details the plan and timeline of the proposed project. It also details the necessary risk assessment.

4.1 Milestones

The TRL benchmark provides a breakdown of the phases of developement, the milestones table 4.2 below highlights the different design stages and tasks throughout the project. The expected durations of these are also presented.

Task	Details	Duration
Create MAC	Create custom Layer 2 Ethernet hardware based on the IEEE 802.3 standard	3-4 Weeks
Wishbone Interface	Connect the Ethernet MAC to the NEORV32 RISC-V Processor using the wishbone interface and access it via software	2 Weels
Webserver	Create and Get the webserver working on the NEORV32 Processor. Web page should be accessed from another computer	5-6 Weeks
Firewall Hardware	Create the hardware between 2 Ethernet MACs to filter out packets based on rules	3-4 Weeks
Integration with software	Add functionality to the server to be able to configure the firewall rules	1 Week
Measure and Compare	Compare to pre-existing solutions	1 Week

Table 4.1: Milestones for the proposed project

4.2 Project Risk Assessment

The majority of the work compeleted in the proposed project is digital and poses little risk outside of the standard office sitting.

Risk	Severity	Likelyhood	Mitigation
Licensing	Minor	Moderate	Avoid software/hardware that requires a specific license.
Data loss	Catastrophic	Unlikely	Ensure all items are backed-up to the cloud and use services such as GitHub where appropriate. Employ a 3 2 1 backup strategy
Hardware Failure	Moderate	Unlikely	Double check all connections to the FPGA board before powering. Reduce excessive handling where necessary to minimise risk of damaging the equiptment
Illness	High	Likely	Take breaks periodically to avoid being over-worked, and take necessary recovery steps if sick.
Missed Deadlines	Major	Likely	Ensure plans are followed and compelete tasks as soon as possible. If behind, spend extra time on project to catch up.

Table 4.2: Risk assessment of proposed project

Bibliography

[1] "IEEE Standard for Ethernet," standard, The Institute of Electrical and Electronics Engineers, Inc., New York, USA, Dec. 2012.

[2] A. C. S. Center, "Acsc annual cyber threat report, july 2021 to june 2022," Nov 2022.

[3] IHS, "The internet of things: a movement, not a market," tech. rep., IHS Markit, 2017.

[4] W. Shi, G. Pallis, and Z. Xu, "Edge computing [scanning the issue]," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1474–1481, 2019.

[5] M. Caprolu, R. Di Pietro, F. Lombardi, and S. Raponi, "Edge computing perspectives: Architectures, technologies, and open security issues," in *2019 IEEE International Conference on Edge Computing (EDGE)*, pp. 116–123, IEEE, 2019.

[6] E. D. Zwicky, S. Cooper, and D. B. Chapman, *Building Internet Firewalls (2nd Ed.)*. USA: O'Reilly and Associates, Inc., 2000.

[7] E. W. Fulp, "Chapter e74 - firewalls," in *Computer and Information Security Handbook*, pp. e219–e237, Elsevier Inc, third edition ed., 2017.

[8] S. M. Trimberger, "Three ages of fpgas: A retrospective on the first thirty years of fpga technology," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 318–331, 2015.

[9] M. Gokhale and L. Shannon, "Fpga computing," *IEEE MICRO*, vol. 41, no. 4, pp. 6–7, 2021.

[10] S. Lin, D. Zhang, Y. Fu, and S. Wang, "A design of the ethernet firewall based on fpga," in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, vol. 2018-, pp. 1–5, IEEE, 2017.

[11] A. Kayssi, L. Harik, R. Ferzli, and M. Fawaz, "Fpga-based internet protocol firewall chip," in *ICECS 2000. 7th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.00EX445)*, vol. 1, pp. 316–319 vol.1, IEEE, 2000.

[12] S. M. Keni and S. Mande, "Packet filtering for ipv4 protocol using fpga," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 400–403, IEEE, 2018.

[13] S. Cuenca-Asensi, A. Martínez-Álvarez, F. Restrepo-Calle, F. R. Palomo, H. Guzmán-Miranda, and M. A. Aguirre, "Soft core based embedded systems in critical aerospace applications," *Journal of systems architecture*, vol. 57, no. 10, pp. 886–895, 2011.

[14] Y.-H. Cheng, L.-B. Huang, Y.-J. Cui, S. Ma, Y.-W. Wang, and B.-C. Sui, "Rv16: An ultra-low-cost embedded risc-v processor core," *Journal of computer science and technology*, vol. 37, no. 6, pp. 1307–1319, 2022.

[15] C. Heinz, Y. Lavan, J. Hofmann, and A. Koch, "A catalog and in-hardware evaluation of open-source drop-in compatible risc-v softcore processors," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–8, IEEE, 2019.

[16] V. A. Frolov, V. A. Galaktionov, and V. V. Sanzharov, "Investigation of risc-v," *Programming and computer software*, vol. 47, no. 7, pp. 493–504, 2021.

[17] M. A. ISLAM and K. KISE, "An efficient resource shared risc-v multicore architecture," *IEICE transactions on information and systems*, vol. E105.D, no. 9, pp. 1506–1515, 2022.

[18] Stephan Nolting (M.Sc.), *The NEORV32 RISC-V Processor*, 2023. v1.8.1-r17-gd1b295de.

[19] "Wishbone B4 SoC Interconnection," standard, OpenCores, Dec. 2010.

[20] J. Hemanth, X. Fernando, P. Lafata, and Z. Baig, "An optimized packet transceiver design for ethernet-mac layer based on fpga," in *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*, vol. 26 of *Lecture Notes on Data Engineering and Communications Technologies*, pp. 725–732, Switzerland: Springer International Publishing AG, 2019.

[21] Microchip Technology Incorporated, *Small Footprint RMII 10/100 Ethernet Transceiver with HP Auto-MDIX Support*, 2012. Revision 1.4 (08-23-12).

[22] P. Guoteng, L. Li, O. Guodong, F. Qingchao, and B. Han, "Design and verification of a mac controller based on axi bus," in *2013 Third International Conference on Intelligent System Design and Engineering Applications*, pp. 558–562, IEEE, 2013.

[23] N. M. Khalilzad, F. Yekeh, L. Asplund, and M. Pordel, "Fpga implementation of real-time ethernet communication using rmii interface," in *2011 IEEE 3rd International Conference on Communication Software and Networks*, pp. 35–39, IEEE, 2011.

[24] T. Stuckenberg, M. Gottschlich, S. Nolting, and H. Blume, "Design and optimization of an arm cortex-m based soc for tcp/ip communication in high temperature applications," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, Lecture Notes in Computer Science, (Cham), pp. 169–183, Springer International Publishing.

[25] L. Liu, N. Li, and L. Feng, "Improvement and optimization of lwip," in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 1342–1345, IEEE, 2016.

[26] FreeRTOS, "Freertos plus tcp - a free thread aware tcp/ip stack for freertos," Aug 2022.