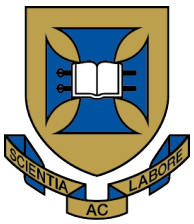

THE UNIVERSITY OF QUEENSLAND

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
CPU	Central Processing Unit
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 Independent Interface
CRC	Cyclic Redundancy Check
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 era of increasing numbers of cyber attacks and record number of connected devices, ensuring these devices operate safely and securely is paramount. The Australian Cyber Security Center (ACSC) received in excess of 76,000 cybercrime reports and growing in the 2021-22 financial year [1]. 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 [2]. This proliferation of IoT devices necessitates robust and adaptable security measures to counter the evolving threats posed by malicious actors.

To manage the surge of IoT devices, a shift to edge computing has emerged in favour of the traditionally more centralised cloud computing architectures. Edge computing as [3] puts it, is the paradigm which involves 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, lower bandwidth requirements, enhanced availability, energy efficiency, improved security and privacy [3]. Consequently, smaller and more efficient computers can be deployed at the edge/perimeter of these networks [4].

Just like any other computer connected to the broader network, edge networks must also be safeguarded from malicious bad actors. Field programmable gate arrays (FPGAs) offer the flexibility of custom hardware that can be designed to incentivise low latency, high throughput yet efficient wire-speed firewalls. While current hardware firewalls exist in todays markets, they often come at a cost and high power usage rendering them unsuitable in edge networks. To address this, this thesis proposal attempts to design a FPGA firewall that fulfils these criteria.

Literature review

Research into existing literature has been conducted on multiple topics in relation to the project. These topics include, field programmable gate arrays, packet filter firewalls, RISC-V processors, Ethernet MAC, webs servers and network stacks.

2.1 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 [5].

A big advantage to custom logic is the ability to create highly parallelised designs with lower latencies than software based serialised algorithms. This comes down to having a great degree of freedom when it comes to designing the architecture and ability to optimise for specific tasks. As such, FPGAs have became ubiquitous in both digital signal processing and for accelerating an assortment of heterogeneous computing architectures and processes [6]. System on chip (SoC) design with custom hardware acceleration modules is an active area research. As [6] points out, there is a focus towards using both hardware and software in *edge* devices due to growing numbers of IoT devices.

Several papers, [7] [8] [9], have proposed a range of other related FPGA based firewalls that have different properties and focus on different optimisations. The key benefit to these firewalls is their high performance - namely, low latency, and high throughput. Article [7] proposed an Ethernet firewall using LwIP (A TCP/IP stack) with five-tuple binding (the five filtered parameters in packet filters) to achieve a throughput of 950Mbps with a latency of 61.266us. A conference proceeding in 2000 [8] used a comparator unit to check the fields of the IP headers 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 hardware such as the Ethernet or packet filtering like the proposed packet filters in [7]. Having a microprocessor in the FPGA design can significantly reduce the complexity of the design and allows for quick and easy development in software instead of hardware [10]. In FPGA design, softcore processors are configurable and can be modelled in a hardware description language (HDL) which can then be synthesised onto ASICs or FPGAs hardware [10]. There are several softcore processors available for FPGA designs including ARM Cortex, Nios II, MicroBlaze, and RISC-V.

While recently the royalty free RISC-V based cores have been popular amongst many SoC designs, other older processors are still common in the literature. The two big FPGA vendors, Xilinx (now

AMD) and Altera (now Intel) have their own RISC based softcores. As an example, Janik et al. [11] used Xilinx's MicroBlaze processor as a media converter between optical (SFP interface) and copper (Ethernet) networks. Likewise, Altera's Nios II can be found in a variety of research papers including an embedded web server which significantly simplified the design [12].

2.2 Packet Filter Firewall

Usually, the first line of defence against bad actors, firewalls play a vital component in computer networks and as such can become vastly complex. In essence, the job of a firewall is to isolate and restrict access to an internal network from an external one to increase security [13].

There are several types of firewalls such as packet filters (PF), stateful packet firewalls and application firewalls [14]. Traditional PFs are considered as stateless and filter exclusively on the fields in the network (layer 2) and transport (layer 3) layer headers [14]. 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 [13]. The book, [13], also highlights some inherent flaws with PFs which include not being able to suppress sophisticated attacks and in some cases, can be challenging to properly configure. More advanced firewalls can perform deep packet inspection which explore the contents of the higher layers to better evaluate a packets true intention [14].

While firewalls such as *iptables* in Linux are software based, hardware acceleration can vastly improve the performance of a packet filter. As stated in section 2.1, hardware acceleration allows for parallelised algorithms to be executed independently of a central processing unit (CPU). Wicaksana and Sasongko, [15], proposed a packet classification engine as shown in figure 2.1. To obtain a fast and reconfigurable packet classifier, the authors of [15] used a hierarchical tree-based algorithm that inspects the multidimensional fields of the IP header through the use of parallel decision trees.

Essentially, the architecture in figure 2.1 employs memory to store the ruleset and uses a multiplexer and a comparator to evaluate each of the fields in the header. As an safegaurd, the authors opted for a *default-deny* ruleset to prevent any unwanted traffic.

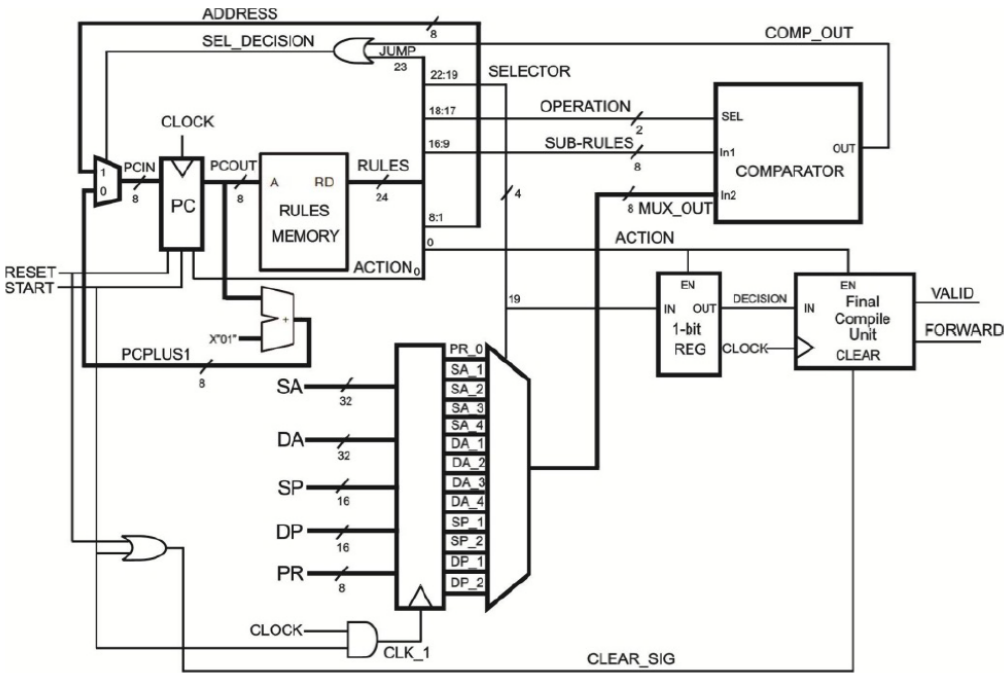

Figure 2.1: Packet classifier [15]

Wasti [16] presents several other classification algorithms for both hardware and software packet filters. *'Sequential matching'* provides the most trivial solution as it matches each rule to the incoming packet. While simple, this design has scalability issues as more rules get added. Another method proposed in [16] is by using a *'Grid of tries'* which uses tries (a type of tree datastructure) to help pattern match the packets, but fails to extend to multiple fields. Hardware algorithms using *Ternary CAMs* (stores words with 3-valued-digits - namely '0', '1' and '*') and *Bit-parallelism* were also discussed. Both of these exploited the parallelised nature of hardware design. One limiting factor with the classification methods cited in [16] is their configurability and expandability.

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 apart of the complex instructions sets (CISC) and are found in the majority of computers. ARM and RISC-V have a reduced instruction set compared to the CISC family and subsequently fall under the RISC family and are ideal for low power microprocessors [17].

RISC-V is an open and royalty free ISA and as a result, a plethora of softcore based custom implementations have been designed [18]. Consequently, there is an abundance of articles delving into RISC-V from evaluating the ISA [19] to creating multicore architectures [20]. A 2019 paper, [18] 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

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

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 [21]. The datasheet, [21], also mentions that it supports a *'Wishbone b4 classic'* 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 [22]. This approach has the benefit of not needing to create custom instructions for the microprocessor.

2.4 Ethernet MAC

First introduced in 1983, the IEEE 802.3 standard [23], 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 2.2.

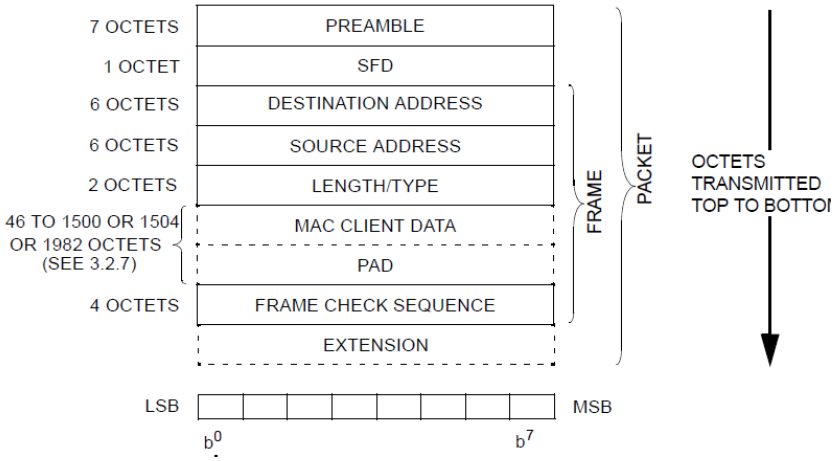

Figure 2.2: MAC layer headers [23]

After the packet has been constructed, the data is forwarded out to the physical (PHY) layer least significant bit (LSB) first [23]. Typically, a PHY management chip is used to handle the physical layer channel encoding amongst other things. These PHY chips often can be interfaced with the media independent interfaces such as MII, RMII, GMII and RGMII [24]. The reduced media independent interface (RMII) is one of these standards defined in [23] and consists of a reference clock, 2 bit wide transmit (TX), 2 bit wide receive (RX) lines and a few other supplementary signals as defined in the LAN8720A datasheet [25].

The MAC layer itself is usually implemented in hardware as it has several advantages over a software implementation. The core reasons behind this are due to parallelised nature of FPGAs and that parts of the MAC can operate independently [26]. One key example is the calculation of the frame check sequence (FCS in figure 2.2). The FCS for Ethernet is a 32bit cyclic redundancy check (CRC) [23] and in addition to Etherent, the CRC32 can be found in an extensive amount of applications. As such, research has been conducted into parallelising the calculation. Noteably, Mitra and Nayak [27] proposed a low latency parallelised architecture for FPGA design on CRC32. As a result, packets can be assembled faster and offload additional processing burden from the CPU.

Numerous articles [24] [28] [29] can be found about Ethernet MACs implemented on FPGAs each with a slightly different approach. Fundamentally though, as best highlighted in [24], a simple way of implementing a MAC is by employing a finite state machine (FSM) to set the required fields. Another technique found in these articles is the use 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 assembly logic at a much higher clock rate than the output RMII reference clock speed [28].

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 incomplete 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-based 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 into consideration when it comes to memory and resource usage [30].

Article [7] 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 [31] [30]. 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 apart from existing documentation. The library aims to provide a threadsafe Berkley sockets API and

[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

network stack supporting multiple protocols such as DHCP, DNS, TCP, and UDP [32]. LwIP is not threadsafe and typically suffers from memory issues as found in [30].

Topic Definition

3.1 Topic

While no single solution can completely safeguard an edge network, an effective approach to mitigating unauthorised/unwanted network traffic is by simply filtering out potentially malicious packets. The proposed thesis project aims to alleviate some of these potential security issues by designing a customised FPGA packet filter featuring a customised Etherent MAC and web server on a RISC-V processor. By accelerating the filtering in hardware, new levels of parallelism, latency and efficiency can be gained compared to current software based implementations.

The packet filter will selectively filter based on various criteria that can be configured through a customised web interface running on the RISC-V softcore processor. This web interface will also display real-time metrics enabling users to monitor network activity. This project will have potentially significant practical applications in the field of edge network security offering an efficient and cost-effective solution alternative to current options in the market.

3.2 System Overview

3.2.1 Hardware Overview

The proposed system consists of two inputs and outputs, namely two ethernet interfaces which can be seen in figure 3.1. This project uses the Xilinx Artix 7 XC7A100T FPGA and two LAN8720A RMII interface chips. The Interface on the left of figure 3.1 is where the internal/private network is attached. The right Ethernet interface is where the external/public network connects.

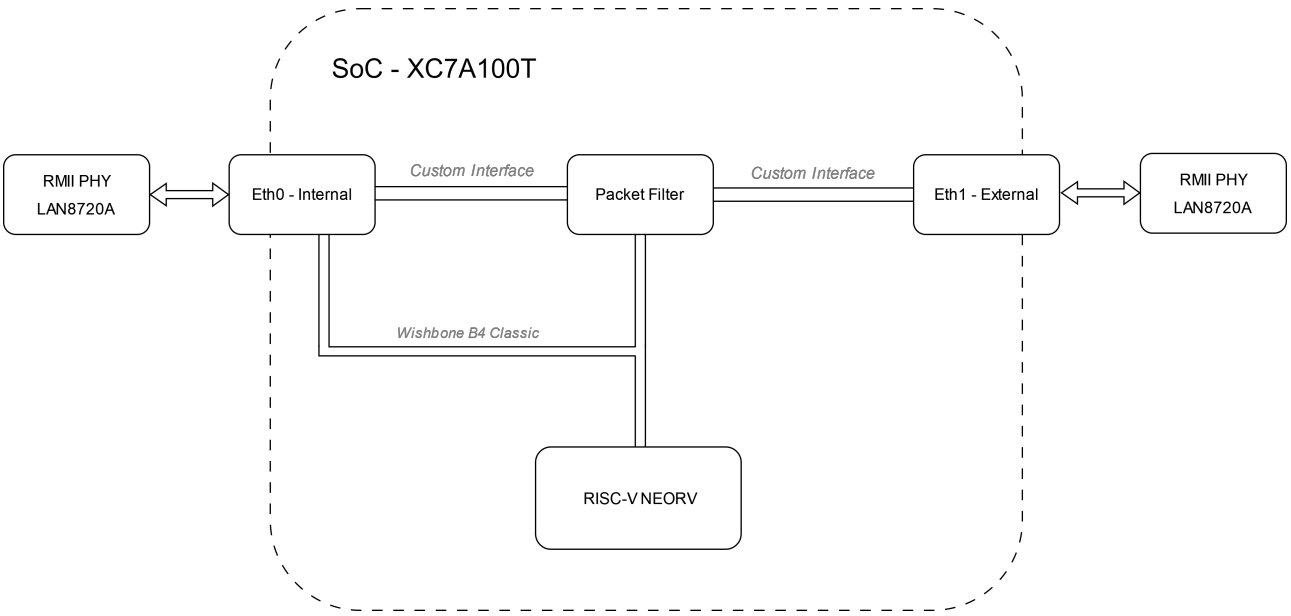

Figure 3.1: System overview

The hardware packet filter will be connected along with the two Ethernet PHY interfaces to the RISC-V softcore processor using the Wishbone B4 classic interface. The two Ethernet PHYs will be connected through the packet filter using a custom high speed interface to reduce the dependance of the microprocessor in the filtering of IP packets.

Implications

Only the internal Ethernet interface is connected to the microprocessor which limits the web page access to devices on the private network. This is proposed to increase the security and only allow configuration of the packet filter from a computer within the protected network.

3.2.2 Packet filtering hardware algorithms

As of this project proposal, a classification algorithm has not been decided on. Several designs are under consideration including both a new customised design and the one proposed in [15].

3.2.3 Microprocessor selection

The proposed project will use the NEORV32[1] RISC-V softcore microprocessor. This microprocessor will run FreeRTOS with the FreeRTOS-Plus-TCP network stack to handle the control of the packet filter and host a web server. Additional libraries such as the FreeRTOS-Plus-FAT will also be used to store web content such as HTML files.

[1] See: https://github.com/stnolting/neorv32

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

The key performance indicators gauge a level of success for the project and are as follows:

- Throughput at which the packet filter can work,
- Added latency to the network,
- Resource utilisation,
- Web server access on a remote machine, and
- Web server concurrency - multiple connections

3.6 Required Equipment

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 development 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. A summary of the required items can be seen below.

- Digilent Nexys A7 100T FPGA board,
- Waveshare LAN8720A ETH Board,
- Waveshare CM4-DUAL-ETH-MINI Router board,
- Raspberry Pi Compute Module 4,
- 32GB MicroSD card,
- USB Power supply,
- An assortment of Cat 5e or better Ethernet cables
- Target/Victim and Attacking Computers

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 development, the milestones table 4.2 below highlights the different design stages and tasks throughout the project. The expected durations of these are also presented. *Note: * delimits assessable items*.

Table 4.1: Milestones for the proposed project

Task	Details	Due	Duration
*Proposal	Write up project proposal	27/04	
Create Wishbone MAC	Create custom Layer 2 Ethernet hardware based on the IEEE 802.3 standard with a wishbone b4 interface	01/05	30h
*Seminar	Create Seminar material and present	08/05	
Create software drivers	Create drivers for the MAC hardware and prepare driver for packet filter	20/05	25h
TCP/IP stack + web server	Create the web server on the NEORV32 Processor. Web page should be accessed from another computer	15/08	75h
Firewall hardware	Create the packet filter hardware to filter based on a ruleset	24/09	20h
Software integration	Add functionality to the server to be able to configure the firewall rules	01/10	10h
Measure and Compare	Compare to pre-existing solutions	09/10	15h
*Demonstration	Demonstrate thesis project	20/10	
*Thesis	Write thesis itself	06/11	

4.2 Project Risk Assessment

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

Table 4.2: Risk assessment of proposed project

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 overworked, and take necessary recovery steps if sick.
Missed Deadlines	Major	Likely	Ensure plans are followed and complete tasks as soon as possible. If behind, spend extra time on project to catch up.
Issue outside project	Moderate	Unlikely	Ensure private life is properly controlled by taking breaks to focus on outside issues.
Eye strain & RSI	Moderate	Likely	Take periodic breaks from using a computer to minimise stain on both eyes, hands and arms.
Minor Burns	Low	Unlikely	Ensure time is given for electronics to cool down and use heatsinks to prevent electronics from getting too hot.

Bibliography

[1] A. C. S. Center, "Acsc annual cyber threat report, july 2021 to june 2022." https://www.cyber.gov.au/acsc/view-all-content/reports-and-statistics/acsc-annual-cyber-threat-report-july-2021-june-2022, Nov 2022.

[2] IHS, "The internet of things: a movement, not a market." https://cdn.ihs.com/www/pdf/IoT_ebook.pdf, 2017.

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

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] L. Janik, M. Novak, L. Hudcova, O. Wilfert, and A. Dobesch, "Lwip based network solution for microblaze," in *2016 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom)*, pp. 1–4, IEEE, 2016.

[12] N. Joshi, P. Dakhole, and P. Zode, "Embedded web server on nios ii embedded fpga platform," in *2009 Second International Conference on Emerging Trends in Engineering and Technology*, pp. 372–377, IEEE, 2009.

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

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

[15] A. Wicaksana and A. Sasongko, "Fast and reconfigurable packet classification engine in fpga-based firewall," in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, (Ithaca), pp. 1–6, IEEE, 2016.

[16] S. Wasti, "Hardware assisted packet filtering firewall." `https://madmuc.usask.ca/Pubs/shw320.pdf`, 2001.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] S. Nolting, "The neorv32 risc-v processor." `https://stnolting.github.io/neorv32/`, 2023. v1.8.1-r17-gd1b295de.

[22] "Wishbone B4 SoC Interconnection." `https://cdn.opencores.org/downloads/wbspec_b4.pdf`, Dec. 2010. Standard.

[23] "IEEE 802.3-2012 IEEE Standard for Ethernet," standard, The Institute of Electrical and Electronics Engineers, Inc., New York, USA, Dec. 2012.

[24] 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.

[25] Microchip Technology Incorporated, "Small footprint rmii 10/100 ethernet transceiver with hp auto-mdix support." `"https://ww1.microchip.com/downloads/en/DeviceDoc/8720a.pdf"`, 2012. Revision 1.4 (08-23-12).

[26] J. Sütő and S. Oniga, "Fpga implemented reduced ethernet mac," in *2013 IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 29–32, 2013.

[27] J. Mitra and T. Nayak, "Reconfigurable very high throughput low latency vlsi (fpga) design architecture of crc 32," *Integration (Amsterdam)*, vol. 56, pp. 1–14, 2017.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] FreeRTOS, "Freertos plus tcp - a free thread aware tcp/ip stack for freertos." `https://www.freertos.org/FreeRTOS-Plus/FreeRTOS_Plus_TCP/index.html`, Aug 2022.