Introduction

1.1 Topic Definition

In recent years, growth of the Internet of Things (IoT) has led to an unprecedented increase in the number of internet-connected devices, with estimates suggesting that there will be over 75 billion IoT devices by 2025 [1]. However, this growth has brought to attention the importance of network security in embedded systems, especially where sensitive data is handled. Cyber-attacks targeting IoT devices have become more sophisticated and frequent, with the number of IoT attacks increasing by 300% in 2019 alone [2]. The financial impact of these attacks is staggering, with the annual cost of data breaches speculated to reach $10.5 trillion by 2025 [3].

Trends in the IoT space such as edge computing, 5G networks, and artificial intelligence (AI) are sky-rocketing the demand for highly-performant processing solutions [4] while traditional software-based security approaches often struggle to keep up with the real-time requirements and resource constraints of IoT devices [5]. Frustaci *et al.* [5] highlight the limited memory, processing power, and energy resources of IoT devices make it challenging to implement strong security measures using software alone without compromising performance and battery life. This is where FPGAs can offer a solution. By leveraging the flexibility of FPGAs, it is possible to create an optomised and dedicated network security solution.

This thesis proposes the development of a RISC-V softcore processor that prioritises network security in embedded IoT applications. The use of RISC-V has several advantages over other contemporary architectures. For one, it is an open-source instruction set architecture that is concise, modular, and extensible [6]. Its open nature allows for customization of the processor design to meet specific requirements, while its modular design enables the addition of custom instructions and extensions which can emphasise security-related tasks [7]. With this, the project aims to show how a dedicated solution can be highly-performant while also mitigating potential threats at the hardware level.

1.1.1 Aims

A successful FPGA-based softcore processor for network security is:

1. Secure within networks.
2. Low-latency.
3. Power efficient.
4. Resource-minimising.
5. And scalable to peripherals/add-ons.

1.1.2 Key Performance Indicators

The previous aims will be evaluated against the following criteria:

1. **Network Security:**
   Packet sniffing and interception testing will be conducted against the ingoing/outgoing encrypted packets to see if the device is encrypting properly. The subsequent key performance indicators also contribute to overall network security.

2. **Processing Latency:**
   Packet processing time will be measured under idle, average and peak network conditions. This will involve testing the system with differing packet amounts and sizes with security features enabled/disabled to assess any performance bottlenecks. The latency results will also be compared with pre-existing security solutions to benchmark the performance of the softcore processor.

3. **Power Efficiency:**
   Power consumption will be measured also during idle, average, and peak loads. The energy efficiency ratio (performance per watt) will be calculated to provide a metric for comparison. Other FPGA-based and software-based solutions will be used to then assess relative performance. There's also the possibility of a thermal camera being used to visualise heat radiation. Additionally, there is also the integrated temperature sensor on the Arty S7 Board [8].

4. **Resource Utilisation:**
   Resource utilization will be evaluated by measuring the counts of FPGA resources in the FPGA design software, such as look-up tables (LUTs), flip-flops and block RAM (BRAM). Memory and CPU utilization on the target IoT devices will also be observed to ensure the system does not overburden the resources available.

5. **Scalability:**
   The processor will have hardware connections open to any additional peripherals to which are Zephyr-compatible. Performance/resource impacts will then be observed.

1.2 Project Overview

This section will now pertain to the complete hardware and software stack of of the proposed solution. A high-level overview of which can be seen in figure 1.1.

Here, the local network refers to the arrangement of the edge devices in the network (left-hand side). It consists of the project's primary focus, the RISC-V softcore processor as well as a Kubernetes cluster network which is scalable to any number of test edge devices. On the right hand side is the router which will interface the local network with the external internet structure, allowing external communications.

Figure 1.1: High-Level Network Overview

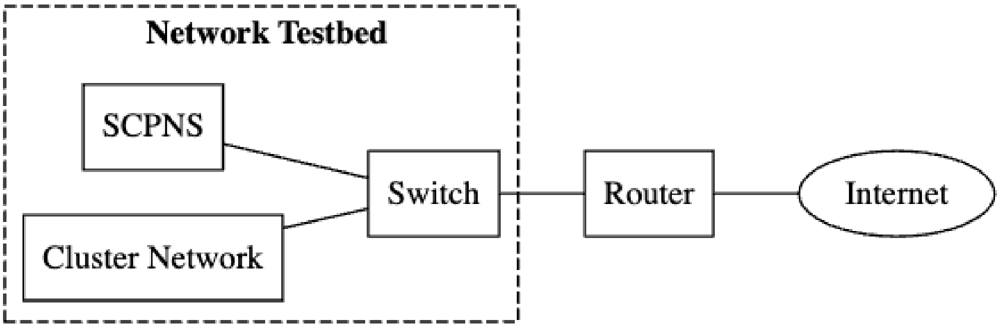
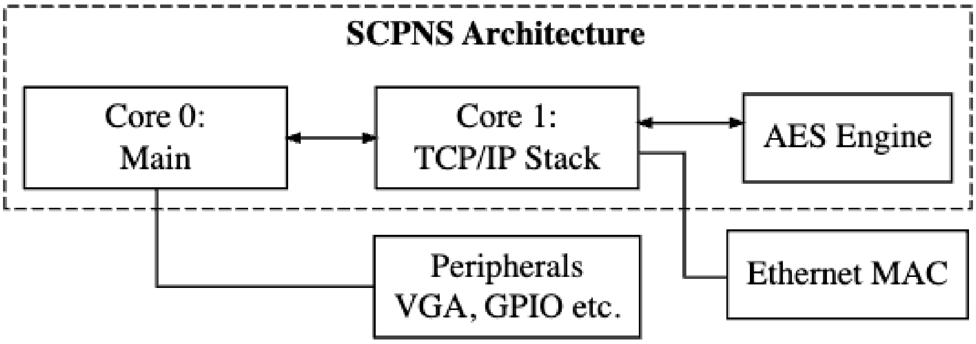


Figure 1.2: High-Level FPGA Architecture Overview



1.2.1 Softcore Processor for Network Security, (SCPNS)

The processor will be running a real-time operating system, Zephyr. Zephyr has support for multicore designs as well as support for many types of peripherals and networking services. Functionality will be split across two cores as seen in figure 1.2. Core 0 will be the main core which will handle task-dispatching, an integrated shell and peripheral interfacing (VGA, GPIO *etc.*). Core 1 will handle the TCP/IP network stack as well as interfacing with the AES engine for capabilities regarding cryptography. Lastly, both cores will have direct communication to each other, allowing the sending/receiving of packets. As the project progresses, the resource allocation for each core will be adjusted exclusively to their requirements.

1.2.2 Cluster Network

Basic software containers, deployed and managed via Kubernetes, will run in the same network alongside the SCPNS. These containers will use custom images consisting of established libraries that provide TCP/IP stack interfacing, as well as the ability to send/receive test packets. The deployment of these containers can also be scaled, allowing any strains to the SCPNS to be observed.

1.2.3 Cryptographic Accelerator, (AES Engine)

To emphasise the network security aspect of the project, an additional layer of cryptographic protection will be implemented in the form of a AES engine SoC (Advanced Encryption Standard) [9], see section 2.1.1. Basically, everytime the network core, Core 1, needs encyption/decryption capabilities it can send data to the SoC via UART. The SoC will be implemented as part of the FPGA architecture as well, ensuring closeness to the network core, akin to Zang *et al.* [10].

1.2.4 Ethernet MAC

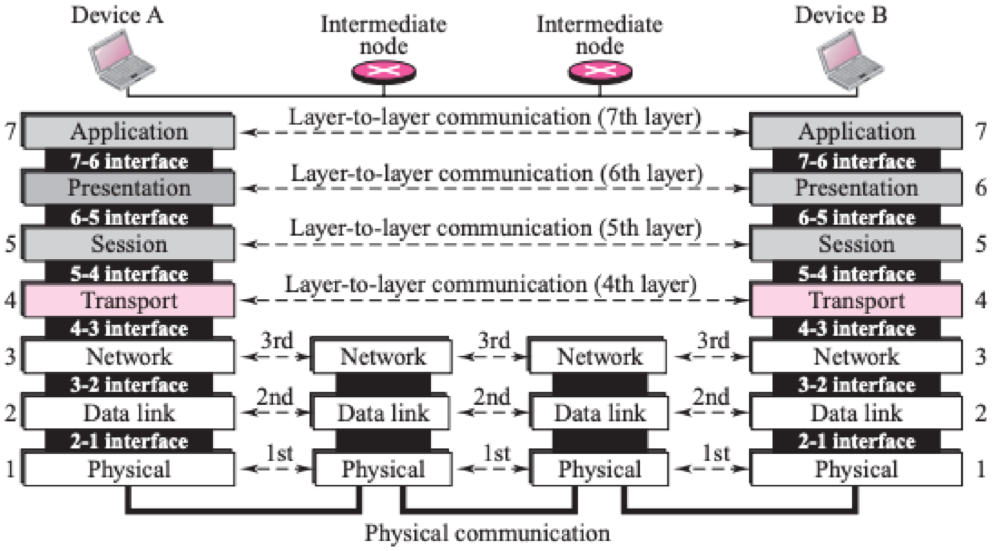
Architecturally, the ethernet MAC module will be placed near the ethernet port on the Arty S7 board, and then be connected directly to the network core (Core 1). Here, the core will have priority access to any incoming or outgoing packets, ensuring that any TCP/IP stack management is fully-offloaded from the main core.

Background

2.1 Network Stacks

A network stack, also known as a protocol stack, is a set of protocols that allow communication between devices on a network, regardless of underlying architecture. The most common network stack is the TCP/IP model [12].

Figure 2.1: TCP/IP Model of Two Devices and Two Intermediary Nodes [12]



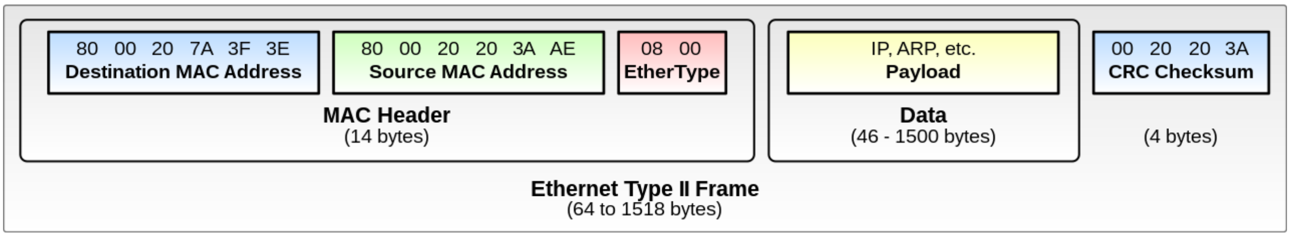
In FPGA-based projects, the TCP/IP stack can be implemented using a combination of hardware and software components [13]. The hardware components, such as the Ethernet MAC and packet processing accelerators, can be designed to handle low-level tasks and offload computationally intensive operations from the software, while the software components, running on the softcore processor, can handle higher-level protocol processing and application-specific tasks [13].

Ethernet MAC

The Ethernet Media Access Control (MAC) sublayer, controls access to the shared medium, i.e. switch, in an ethernet-based network [14]. In this project, the softcore processor will include an Ethernet MAC module to facilitate communication with other devices on the local area network, such as the switch and connected devices. The MAC module encapsulates higher-layer data into Ethernet frames, transmits them over the physical medium, receives incoming frames, and extracts the data for processing by higher layers [14], the exact buffer format for this can be seen in 2.2.

The Ethernet MAC module also features ”Carrier Sense Multiple Access with Collision Detection”, (CSMA/CD), which manages access to the shared medium and resolves collisions when multiple

Figure 2.2: The Common Ethernet Frame Format, Type II Frame Buffer [15]



devices attempt to transmit simultaneously [14]. By incorporating an Ethernet MAC module, the processor will be easier to integrate into existing networks.

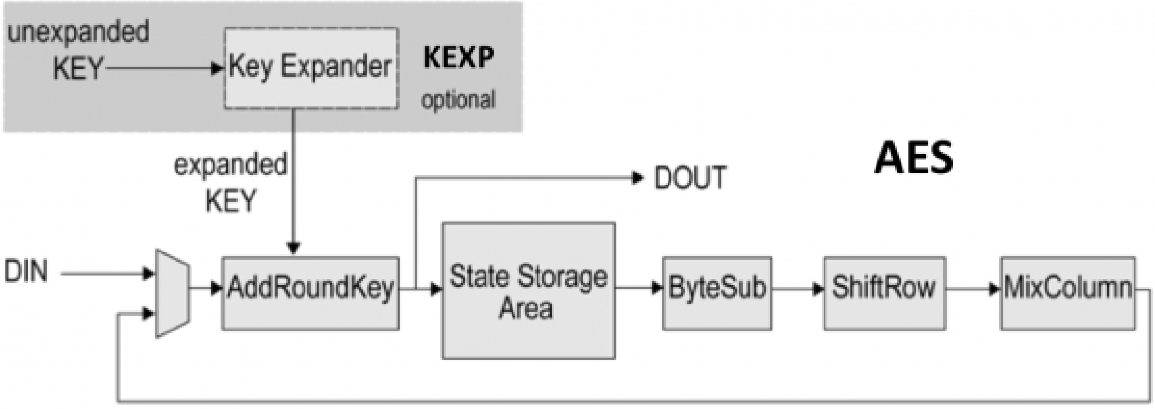
2.1.1 Network Security Methods

Network security methods are techniques used to emphasise integrity, confidentiality, and availability of data transmitted over a network. A common method of enhancing network security is encryption, which will be implemented in this project via a cryptographic acceleration SoC.

AES Engine and Cryptographic Acceleration

Cryptographic acceleration is the use of hardware to perform encryption and decryption operations, offloading these computationally intensive tasks from the main processor. To achieve this in the project, the AES Engine SoC by CAST will be implemented. Figure 2.3 shows the basic control flow and methodology of the SoC.

Figure 2.3: AES Engine Block Diagram [9]



The AES core is fully synchronous to data in/data out transmissions, and customisable to several encryption key lengths. Most notably, there are implementations available in LiteX, see 2.3.1. Using hardware acceleration this way can significantly improve the performance and energy efficiency of cryptographic operations, especially compared to software-based implementations [16].

2.2 Operating Systems

An operating system (OS) is a software that manages computer hardware, software resources, and provides common services for computer programs. This project will explore two different types of operating system design. Real-time operating systems (in this case, Zephyr), and containerised Debian images managed by Kubernetes.

2.2.1 Zephyr (RTOSs)

A real-time operating system (RTOS) is an OS designed to support real-time applications, which require deterministic and timely processing of data. RTOSs are commonly used in IoT devices due to their ability to grant safe, multi-threaded capabilities to processors as opposed to basic cyclic executive code execution. Additionally, they only consist of the absolute minimum requirements for running on bare-metal hardware.

While there are numerous choices for embedded operating systems, Zephyr is one such RTOS that has had continual open-source support and comes packaged with all the services the project requires. This includes thread management primitives, multicore support and networking [17].

2.2.2 Kubernetes Clusters

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Kubernetes clusters are a set of *nodes* (physical or virtual machines) that run containerized OS images managed by Kubernetes [18]. With regards to the project, these containers will only need to be capable of networking interfacing and interaction, thus Debian-based images will be more than sufficient.

2.3 Field Programmable Gate Arrays (FPGAs)

FPGAs are integrated circuits that can be programmed to implement custom hardware designs. FPGAs offer flexibility, reconfigurability, and hardware acceleration, making them attractive for implementing custom processors and accelerators for specific applications [19], such as our network security module.

FPGA devices have inherent resource constraints that must be adhered to. These constraints include the available logic resources (e.g., lookup tables, flip-flops), memory (e.g., block RAM, distributed RAM), and specialized resources (e.g., DSP slices, clock management tiles) [20]. This often involves making trade-offs between the number of CPU cores, cache sizes, peripheral support, and the allocation of resources for each portion of hardware [20] [21].

2.3.1 Chosen CPU Model: VexRiscv

The VexRiscv CPU, a 32-bit RISC-V soft processor core, has been chosen due to its open-source nature, flexibility, and compatibility with the Zephyr RTOS and the LiteX SoC builder framework. VexRiscv is specifically designed for FPGA implementations, meaning that it is conformant to the architectural characteristics of FPGAs, promising a performant and resource-efficient processor system [22].

Zephyr officially supports the RISC-V architecture, and VexRiscv has been successfully integrated with Zephyr in various projects [17]. Moreover, VexRiscv is compatible with LiteX, an open-source SoC builder framework that simplifies the integration of processor cores, peripherals, and custom hardware modules [23]. LiteX provides a high-level framework for generating and configuring SoC designs, making it easier to integrate VexRiscv into a complete system [23]. This compatibility is proven by well-documented examples and guides available in the Zephyr and LiteX communities [17] [23].Compared to other typical CPU models, such as ARM Cortex-M, Xilinx MicroBlaze, or Intel NIOS II, VexRiscv also has the advantages of being open-source. This lack of IP protection enables the addition of custom instructions and extensions tailored to the specific requirements [22], allowing the project to take full advantage of the Arty S7 Board.

The Arty S7 board features a Xilinx Spartan-7 FPGA (XC7S50), which offers a moderate amount of logic resources (52,160 logic cells) and memory (2,700 Kb), with a universal clock rate of 100MHz [24]. VexRiscv's resource-efficient design makes it a viable option for the Arty S7 board. By configuring VexRiscv with appropriate options, such as a balanced pipeline, moderate cache sizes, and essential peripherals, it is possible to implement a dual-core system that fits within the FPGA's constraints while leaving room for additional modules, such as the ethernet MAC or AES engine.

Lastly, VexRiscv benefits from ongoing community support. The availability of open-source tools, such as SpinalHDL, LiteX, and SBT (Scala Build Tool), form the basis for a complete development environment [23] [25].

Literature Review

3.1 Pre-Existing Projects/Solutions

This section pertains to three recent and unique approaches to the ”FPGA RISC-V Softcore processor IoT Application” space. These papers explore the use of RISC-V processors, FPGAs, and accompanying hardware acceleration techniques for network security in resource-constrained edge devices.

3.1.1 Cryptography Integration for Edge Devices

One notable study by Zang *et al.* [10] focuses on using existing cryptographic modules with an FPGA-based RISC-V processor. The authors demonstrate high speed and minimal overhead with an AES SoC accompanied with a RV32I processing core, compared to software-based solutions. However, even though the architecture for the RV32I is programmable and extensible, its only purpose is to interface and test the AES SoC and does not address other aspects of IoT, such as real-time applications or peripherals. Overall, though, it shows us the theoretical maximum for speed in cryptography regarding FPGA boards clocked at approximately 100MHz.

Another article by Yang *et al.* [26] proposes another similar IoT framework using a RISC-V processor and hardware encryption accelerators on an FPGA. Their system leverages the flexibility of FPGAs to implement hardware-based security primitives, of which they describe more-indepth than Zang *et al.* [10]. While their approach also shows significant speed-ups compared to software-based approaches, once again it does not specifically target network security challenges or the impacts on performance once there’s integration with an RTOS.

3.1.2 Additional Security Methods at the Hardware-Level

For cybersecurity concerns other than encryption there’s project by Haj-Yahya *et al.* [27] which demonstrates a custom lightweight RISC-V processor focussing on secure booting. The architecture incorporates efficient implementations of the Elliptic Curve Digital Signature Algorithm (ECDSA) for authentication, the Secure Hash Algorithm 3 (SHA3) for hashing, and DMA for improved performance. As a result, these components can be adapted to any RISC-V architecture, creating a robust and secure boot process for the device.

3.1.3 Summary

While these studies provide valuable insights into the use of RISC-V processors and FPGAs for IoT security, still there remains gaps in the research that this project will focus on. Firstly, the integration of RISC-V processors with real-time operating systems like Zephyr, which is crucial for general-purpose computing and IoT applications, is not thoroughly explored. While yes, there is a significant speed-up in optomising hardware for specific applications, it is equally important to consider the flexibility and adaptability of the system to handle a wide range of IoT scenarios and future security challenges. And secondly, most of the reviewed studies do not investigate the potential of multi-core design in RISC-V processors for parallel processing, they only target hardware-acceleration instead. This project aims to address these gaps by developing a multi-core RISC-V processor system running Zephyr, demonstrating a flexible and scalable solution that emphasises network security.

Project Plan

4.1 Milestones and Timeline

Table 4.1 is the full outline and expected completion for each milestone. The idea is to ensure the surrounding testbed is functioning correctly before designing the processor itself.

Table 4.1: Proposed Milestone Timeline

Task	*Description*	*Due*
Semester 1		
Proposal	Project proposal	Week 8
Interface RPi/s	Have the local area network set up and RPis interfaced to switch, ready to run the Kubernetes cluster	Week 9
Set up Cluster Network	Deploy Kubernetes cluster, creating the testbed	Week 10
Seminar	Seminar presentation	Week 11
Mid-year Break		
Create the SCPNS	Implement proposed design	Week 1
Run Zephyr	Flash and run Zephyr on the softcore processor.	Week 2
Interface Network Core	Get the SCPNS interfaced in the LAN	Week 3
AES Encryption Engine	Implement the AES engine	Week 3-4
Semester 2		
Experimentation	Start benchmarking and taking measurements	Week 1
Measure and Compare	Compare measurements to pre-existing solutions	Week 2
Poster & Demonstration	Thesis project demonstration	Week 11
Thesis	Thesis Write-up	Week 14

4.2 Project Risks

This project is low-risk and does contain any ethical concerns as the project is focused on digital programming and interfacing. However, there are still risks which are outlined in table 4.2.

Table 4.2: Project Risks

Risk	*Likelihood*	*Severity*	*Mitigation & Prevention*
Missing Milestones	Likely	Moderate	Perfectionism will be avoided as much as possible, completing milestones will be emphasised instead. There are also several weeks of time in semester 2 that can be extended into if needed.
Losing Work	Catastrophic	Unlikely	Every increment of progress will be committed to an online Git Repository, off-loading work from any local devices.
Hardware Damage	Moderate	Unlikely	Electrical connections to hardware such as the FPGA board will be scrutinised before powering on. As for workstations, work will always be committed online so no progress is lost.
Improper Licensing	Minor	Unlikely	Even though the majority of software that will be used is open-source, licenses will be checked if the use-case is permissible. If not, then alternatives can be sought.
Health Risks	Moderate	High	Breaks/pauses from working time will be allocated. As there will be a lot of sitting, an exercise routine will be maintained outside of working times.
Working Outside Scope	Unlikely	Moderate	The milestones and topic definition will be closely adhered to. In case there's any additions to the scope, it will always be consulted by the supervisor first.

Bibliography

[1] T. Alam, “A reliable communication framework and its use in internet of things (iot),” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 3, May 2018.

[2] M. Michael, “Attack landscape h1 2019: Iot, smb traffic abound,” September 2019.

[3] S. Morgan, “Cybercrime to cost the world $10.5 trillion annually by 2025,” November 2020.

[4] N. Nuttall, “Top strategic iot trends and technologies through 2023,” September 2018.

[5] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, “Evaluating critical security issues of the iot world: Present and future challenges,” *IEEE Internet of Things Journal*, vol. 5, pp. 2483–2495, October 2017.

[6] D. A. Patterson and A. Waterman, “The risc-v revolution,” in *2017 IEEE International Conference on Computer Design (ICCD)*, pp. 1–5, 2017.

[7] A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. Asanović, “The risc-v instruction set manual volume ii: Privileged architecture version 1.9.1,” Tech. Rep. UCB/EECS-2016-161, EECS Department, University of California, Berkeley, November 2016.

[8] Digilent, *Arty S7 Reference Manual*, October 2019.

[9] C. Inc., *AES, Advanced Encryption Standard Engine*, 2024. Accessed: 2024-04-10.

[10] Z. Zang, Y. Liu, and R. C. C. Cheung, “Reconfigurable risc-v secure processor and soc integration,” in *2019 IEEE International Conference on Industrial Technology (ICIT)*, pp. 827–832, 2019.

[11] M. Giplin, “Fpga packet filter with ethernet mac and web server using a risc-v softcore processor,” April 2023.

[12] B. A. Forouzan, *TCP/IP Protocol Suite*. McGraw-Hill Education, 5th ed., 2021.

[13] A. Perrig, J. A. Stankovic, and D. Wagner, “Security in wireless sensor networks,” *Communications of the ACM*, vol. 47, pp. 53–57, June 2004.

[14] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*. Pearson, 8th ed., 2021.

[15] Wikipedia, “Ethernet frame,” Mar 2024. `https://en.wikipedia.org/wiki/Ethernet_frame`.

[16] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Pearson, 8th ed., 2019.

[17] Zephyr Project, *Zephyr OS, An RTOS for IoT*, 2023. `https://www.zephyrproject.org/`.

[18] B. Burns, J. Beda, and K. Hightower, *Kubernetes: Up and Running: Dive into the Future of Infrastructure*. O'Reilly Media, 2nd ed., 2019.

[19] S. Trimberger, "Three ages of fpgas: A retrospective on the first thirty years of fpga technology: This paper reflects on how moore's law has driven the design of fpgas through three epochs: the age of invention, the age of expansion, and the age of accumulation," *IEEE Solid-State Circuits Magazine*, vol. 10, pp. 318–329, April 2018.

[20] I. Kuon, R. Tessier, and J. Rose, "Fpga architecture: Survey and challenges," *Foundations and Trends in Electronic Design Automation*, vol. 2, pp. 135–253, April 2008.

[21] D. Koch, F. Hannig, and D. Ziener, eds., *FPGAs for Software Programmers*. Springer, 2016.

[22] SpinalHDL, *VexRiscv CPU, A FPGA friendly 32 bit RISC-V CPU implementation from SpinalHDL*, 2023. `https://github.com/SpinalHDL/VexRiscv`.

[23] LiteX Project, *LiteX SoC Builder*, 2023. `https://github.com/enjoy-digital/litex`.

[24] Xilinx Inc., *Arty S7 Schematic*, 2023. `https://digilent.com/reference/programmable-logic/arty-s7/start`.

[25] SpinalHDL, *SpinalHDL, A FPGA HDL Code Generator*, 2023. `https://spinalhdl.github.io/SpinalDoc-RTD/index.html`.

[26] S. Yang, L. Shao, J. Huang, and W. Zou, "Design and implementation of low-power iot risc-v processor with hybrid encryption accelerator," *Electronics*, vol. 12, no. 20, 2023.

[27] J. Haj-Yahya, M. M. Wong, V. Pudi, S. Bhasin, and A. Chattopadhyay, "Lightweight secure-boot architecture for risc-v system-on-chip," in *20th International Symposium on Quality Electronic Design (ISQED)*, pp. 216–223, 2019.