Design and Implementation of RISC-V Processor with ML Accelerator

By

Faizan Ahmad

Enrollment No.	01-133192-030

Sawera Aslam

Enrollment No.	01-133192-123

Hamna Shakil

Enrollment No.	01-133192-037

Supervised By

Dr. Atif Raza Jafri

Session 2019-23

This Report is submitted to the Department of Electrical Engineering, Bahria University, Islamabad.
In partial fulfillment of requirement for the degree of BS(EE).

Certificate

We accept the work contained in this report as a confirmation to the required standard for the partial fulfillment of the degree of BS(EE).

____________________.	____________________
Head of Department	Supervisor
____________________.	____________________
Internal Examiner	External Examiner

Dedication

We express our gratitude to our family, friends, and mentors who have been a constant source of support and encouragement throughout our dissertation journey. Our parents, in particular, have played a significant role in motivating us to persevere through challenges. We extend our heartfelt thanks to our supervisors for their guidance and assistance in developing our skills, which will undoubtedly prove valuable in our future endeavors. This dissertation is dedicated to all those who have helped us along the way.

Acknowledgments

We would like to express our gratitude to our Creator Allah S.W.T. who primarly helped in each and every step of achieving any progress in this project. We can say with full and complete confidence that without His blessings and help we would never be able to accomplish such a huge project and InshaAllah He will always be there for us. Secondly He gave us such good and helping friends and supervisors who also guided us whereever we were stuck. Thank you Dr. Atif Raza Jafri and Mr. Muhammad Ali for being supportive and an encouragement for us in this project.

Abstract

We are currently witnessing the dawn of a new industrial revolution, characterized by the rapidly changing industrial landscape. In this revolution, RISC-V has emerged as a widely used term to describe the development of custom processors designed to meet the power and performance requirements of newer workloads for AI, ML, and IoT. RISC-V is an open standard ISA that is built on RISC principles and plays a critical role in linking software and hardware layers of computer abstraction. The primary drivers of this product include building custom processors, boosting speed, reducing costs, enhancing security, developing a platform for new students in this field, promoting innovation and skills, and competing with other companies such as Intel (x86), ARM (ARM ISA), and others.

This project presents the implementation of a RISC-V processor on FPGA with an integrated accelerator for machine learning (ML) applications. The processor used in this project is VexRiscv with ztachip accelerator. The main objective of this project is to develop a hardware system with high performance and efficiency for ML applications. To achieve this objective, the VexRiscv processor is modified to transfer tensor instructions to ztachip for accelerating ML computations.

The implementation of this integrated system is carried out on an FPGA platform, which allows for flexibility and easy reconfiguration. The design is verified using debugging tools and is tested on various ML applications. The results show that the designed processor with the integrated accelerator outperforms traditional processors and accelerators in terms of both speed and energy efficiency. The tensor instructions of ztachip for ML computations also provide significant speedup compared to standard RISC-V instructions.

Overall, the project demonstrates the feasibility and effectiveness of implementing a RISC-V processor with an integrated accelerator for ML applications. This work has implications for the development of high-performance and energy-efficient RISC-V accelerators for various applications, particularly in the field of ML.

Contents

List of Figures

List of Tables

Abbreviations

A	Atomic Instructions
ADC	Analog to Digital Converter
AEAD	Authenticated Encryption with Associated Data
AI	Artificial Intelligence
ALU	Arithemetic and Logic Unit
AMD	Advanced Micro Devices
APB	Advanced Peripheral Bus
ARM	Advanced RISC Machines
ASIC	Application Specific Integrated Circuit
ATSC	Advanced Television Systems Committee

AXI	Advanced eXtensible Interface
B	Bit Manipulation
BOOM	Berkeley Out of Order Machine
C	Compressed Instructions
CEO	Chief Executive Officer
CHIPS	Common Hardware for Interfaces, Processors and Systems
CISC	Complex Instruction Set Architecture
CLB	Configurable Logic Block
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
CSR	Control and Status Register
DC	Direct Current
DDR	Dual Data Rate
DFA	Differential Fault Attacks
DLX	Deluxe
DMA	Direct Memory Access
DMIPS	Dhrystone Million Instructions per Second
DMR	Dual Modular Redundancy
DPA	Differential Power Attacks
DSA	Domain Specific Architecture

DSL	Domain Specific Language
DVD	Digital Video Disc
E	Embedded
ECC	Error Correcting Codes
EECS	Electrical Engineering and Computer Science
EEES	Energy Efficient Embedded Systems
ETH	Eidgenössische Technische Hochschule
EX	Execution
F	Single-Precision Floating Point
FIFO	First In First Out
FMA	Fused Multiplication and Addition
FMS	Fused Multiplication and Subtraction
FPGA	Field Programmable Gate Arrays
FPS	Frames Per Second
G	General
GCC	GNU Compiler Collection
GDB	GNU Debugger
GFLOPS	Giga Floating-Point Operations per Second
GHz	Giga Hertz
GOPS	Giga Operations Per Second

GPIO	General Purpose Input Output
GPR	General Purpose Register
GPU	Graphic Processing Unit
GUI	Graphical User Interface
HDL	Hardware Description Language
HDMI	High Definition Multilmedia Interface
HPC	High Performance Computing
HPSC	High Performance Spaceflight Computer
HS	Horizontal Sync
I	Integer Base ISA
IBM	International Business Machines
ID	Instruction Decode
IEEE	Institute of Electrical and Electronics Engineers
IF	Instruction Fetch
IIS	Integrated Systems Laboratory
INT	Integer
IoT	Internet of Things
IP	Intellectual Property
ISA	Instruction Set Architecture
JTAG	Joint Test Action Group

K	Kilo
KB	Kilo Byte
kGE	Kilo Gate Equivalent
LSU	Load and Store Unit
M	Multiplication and Division
MAC	Multiplication and Accumulation
Mbps	Mega Bits Per Second
MHz	Mega Hertz
MIG	Memory Interface Generator
mm	Milli Meter
mW	Milli Watt
MXU	Matrix Multiplication Units
NASA	National Aeronautics and Space Administration
NESCOM	National Engineering and Scientific Commission
NUST	National University of Science and Technology
OCD	On Chip Debugger
OTP	One Time Programmable
OV	Omni Vision
PC	Personal Computer
PGC	Programmable Gain Control

PQC	Post Quantum Cryptography
PULP	Parallel Ultra Low Power
PWDN	Power Down
PWM	Pulse Width Modulation
QSPI	Quad Serial Peripheral Interface
RGB	Red Green Blue
RISC	Reduced Instruction Set Architecture
RISC-V	Reduced Instruction Set Computer Five
RTL	Register Transfer Level
RV	RISC-V
SA	Systolic Arrays
SDRAM	Synchronous Dynamic Random Access Memory
SFA	Statistical Fault Attacks
SHL	Shift Left
SHR	Shift Right
SIFA	Statistical Ineffective Fault Attacks
SoC	System on Chip
SOI	Silicon On Insulator
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory

SSD	Single Shot Detection
symbol	description
TPU	Tensor Processing Units
UART	Universal Asynchronous Receiver Transmitter
UC	University of California
US	United States
USB	Universal Serial Bus
V	Volts
VCR	Videocassette Recorder
VDMA	Video Direct Memory Access
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language
VLIW	Very Long Instruction Word
VS	Vertical Sync
W	Watt
YAML	Yet Another Markup Language / YAML Ain't Markup Language

Chapter 1

Introduction

Our project is "Implementation of RISC-V based processor on FPGA kit with integrated accelerator for ML algorithms". It is the initial and significant step towards innovation and improvement. Software and hardware interface is the important abstraction layer in the computer system. RISC-V is distinctive, innovative, because it is a common, free, open-source ISA to which software can be interfaced, hardware can be design, and processors can be built to support it.

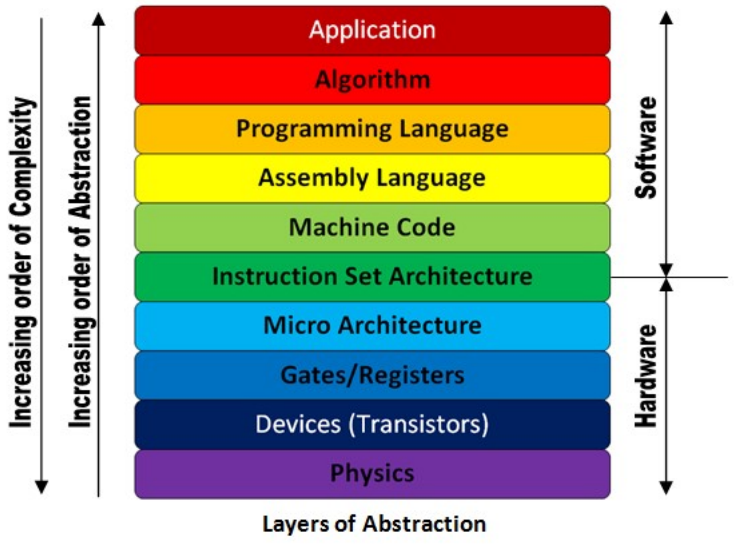


Figure 1.1: Layers of Abstraction of Computer Architecture

Source: Secplicity

As this figure shows different layers in an architecture of computer in order. The most important of them all is the middle most layer named as Instruction Set Architecture. It links the software and the hardware of any computer or processor based system i.e. laptops, cellphones, electronic devices etc.

1.1 Project Background

RISC-V, which is based on RISC architecture and pronounced as "risk-five," was created at the University of California, Berkeley. The concept of simpler and more effective computers had been around since the 1980s, but the design principles were not yet fully understood. There was academic interest in computers with high power and the ability to execute multiple instructions per cycle. DLX, a RISC processor architecture, was designed by John L. Hennessy and David A. Patterson with a focus on education and research. Later, David Patterson contributed to the development of RISC-V. Although researchers created DLX using field programmable gate arrays, it was not intended for commercial use.

The earlier versions of ARM CPUs that utilized RISC architecture had an instruction set that was available in the public domain and is still supported by the GNU Compiler Collection (GCC), a widely used free software. GCC can compile code, manage library dependencies, and convert high-level programming languages such as C/C++ into assembly code, which can then be transformed into executable files. While there were three open-source core designs based on this instruction set, they were never actually manufactured. OpenRISC is an open-source instruction set architecture (ISA) that is based on DLX and can be supported by GCC and Linux implementations, although there have been very few commercial implementations to date. Krste Asanovic, along with some of his graduate students, embarked on a brief three-month project in 2010 with the goal of creating a design that would appeal to both academic and industrial users. David Patterson, the creator of Berkeley RISC, also joined the project, which became known as RISC-V and was the fifth generation of his RISC-based research projects. Despite its initial brief timeline, the project had

Figure 1.2: RISC-V Logo

Source: RISC-V Foundation

significant impact and was successful in achieving its objectives.

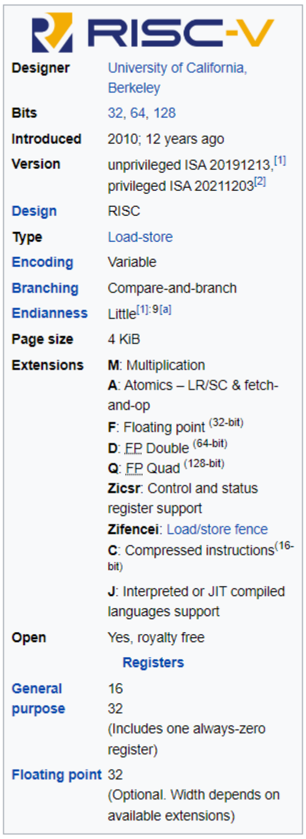
1.2 Project Overview

This chart shows some of the descrption of RISC-V ISA. It comes with three variants i.e. 32, 64 and 128 bits. It has various optional extensions e.g. M extension contains instructions related to multiplication and division, F extension means that it will support single precision floating points of different width depending on the variant being used and many other extensions. It also contains 16 or 32 general purpose registers again depending on the extension being used either I (32) or E (16). It supports little endian while reading or writing in memory.

1.2.1 Keypoints about RISC-V

- The RISC-V instruction set architecture is a free and open ISA that draws on three decades of RISC architecture development. It is designed to be simple and easy to understand, with a reduced instruction set suitable for both lower-power embedded systems and high-performance computers. RISC-V's open nature and simplified instruction set make it accessible for anyone to use freely and contribute their skills and work towards its development.

- RISC-V has flexible nature, which means that it can be used for



Designer	University of California, Berkeley
Bits	32, 64, 128
Introduced	2010; 12 years ago
Version	unprivileged ISA 20191213,[1] privileged ISA 20211203[2]
Design	RISC
Type	Load-store
Encoding	Variable
Branching	Compare-and-branch
Endianness	Little[1]:9[a]
Page size	4 KiB
Extensions	**M**: Multiplication **A**: Atomics – LR/SC & fetch-and-op **F**: Floating point (32-bit) **D**: FP Double (64-bit) **Q**: FP Quad (128-bit) **Zicsr**: Control and status register support **Zifencei**: Load/store fence **C**: Compressed instructions(16-bit) **J**: Interpreted or JIT compiled languages support
Open	Yes, royalty free
	Registers
General purpose	16 32 (Includes one always-zero register)
Floating point	32 (Optional. Width depends on available extensions)

Figure 1.3: Descrption of RISC-V ISA

Source: Wikipedia, The Free Encyclopedia

embedded applications as well as for high end computations. This is possible due to its feature of allowing optional extensions. It's base extension, although, has all the instructions required to compile a C program but there are also optional extensions e.g., for multiplication or atomic operations or vector operations etc. If RISC-V is being used by an individual or a small-scale industry which require it to perform low end computations or want to use less hardware, then they can use the base extensions or any other optional extension along with it. But

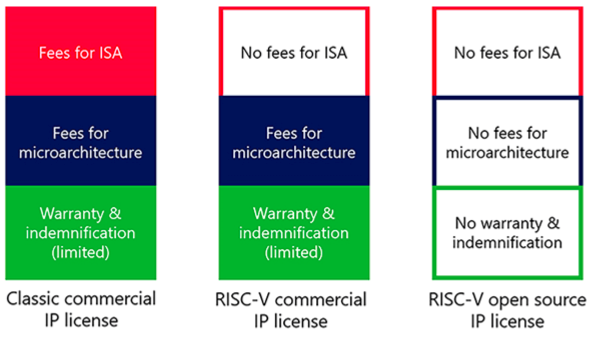

Fees for ISA
Fees for microarchitecture
Warranty & indemnification (limited)
Classic commercial IP license
No fees for ISA
Fees for microarchitecture
Warranty & indemnification (limited)
RISC-V commercial IP license
No fees for ISA
No fees for microarchitecture
No warranty & indemnification
RISC-V open source IP license

Figure 1.4: Open Source RISC-V ISA [?]

Source: Microcontroller Tips

if RISC-V is being used by a very large organization and applications required are high end and need to perform a lot of computations or being used in a PC or laptop then RISC-V can have all other required extensions to perform those complex operations in no time. No wonder NASA and Google are also start using RISC-V in their space missions and data centers respectively.

- RISC-V ISA is more secure and reliable as compared to other ISA's because when we will build our own processors using RISC-V ISA we will surely be knowledgeable about the algorithms and logic inside of that processor so there will be probably zero possibility of data thieving. If we talk about other processor providers or ISAs, then Intel and ARM are on the top of the list. Intel, as we all know, doesn't allow access to its ISA i.e., 8086 or x86 etc. Whereas ARM has its ISA named as ARMISA and a lot of companies make processors for cellphones based on ARMISA e.g., Qualcomm or Snapdragon etc. But ARM doesn't allow the access to its ISA for free. It gives license to their customers, and which can be pretty out of range for small scale companies. It erects a barrier to the commercialization of successful research ideas.

- RISC-V uses simple instructions in its instruction set as compared to other ISAs which uses complex instructions both in form of no. of operations as well as size of the instructions which is not same for all instructions. Intel ISA x86 was traditionally built as a CISC (Complex Instruction Set Computer) ISA, while ARM was built as RISC (Reduced Instruction Set Computer). Originally, CISC machine's goals to execute fewer, more complex instructions and do more computation in single instruction. RISC based on simpler instructions that are easier and faster to execute and only execute one instruction per cycle.

Figure 1.5: Open-Source vs. Closed

Source: Bluespec

- RISC-V will support both proprietary and open implementations. Another aspect of RISC-V which makes it different from other ISAs is that it allows, and in fact, encourage custom extension. Adding custom extensions is a feature which can bring innovation to the the computer industry. It means that anyone can add any self-made instruction into the ISA of RISC-V for their own use as well as others which will be useful for any specific work load. This is a new way of getting things faster.

1.3 Problem Statements

- Pakistan doesn't make its own processors whenever we require a processor, we import it from foreign countries.

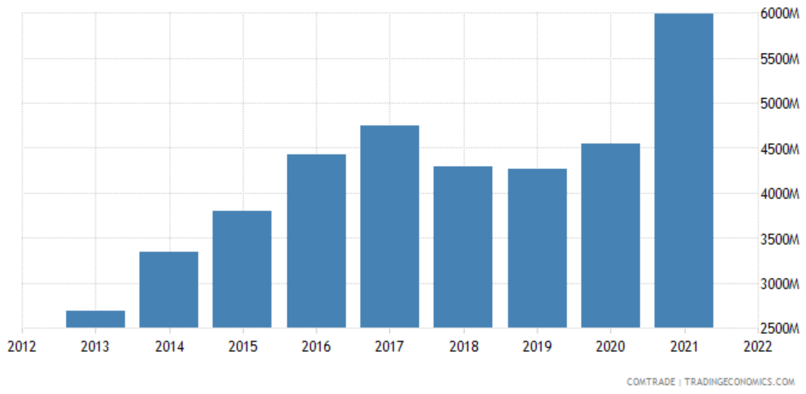


Figure 1.6: Pakistan Imports of Electrical, and Electronic Equipment - 2022

Source: Trading Economics

As we can see in this bar graph the import of Pakistan of electronic products is increasing every year. Most of these products have a processor in their core doing all the working and computations. So the lack of processor is one of the major issues due to which we are unable to develop our own electronic products.

- Now to make any processor one should have access to any ISA. But unfortunately, we don't have access to the ISAs of other processors (for free) like Intel's ISA x86, ARM's ISA ARMISA and many others. So, there is a need to explore and implement an ISA to develop our own processor or to develop our own ISA but it is not a simple task to do.

- Another major issue of using imported foriegn country made processors is of security. We don't know what kind of algorithms are used inside those processors and there is a chance of data stealing or hacking. Such issues are very critical especially in those areas where we need tight securities for example in military areas, NESCOM etc. Any individual would also not like that his or her personal data may be accessed by any unknown person so this issue is also crucial in cloud computing, business purposes, import and export, personal information and so on.
- In the field of processor designing, Pakistan lags behind due to the lack of a suitable platform for engineers in this domain. This is unfortunate, as processors are a crucial component in both computer and electronics industries. Most daily-use products such as home appliances, cars, planes, medical devices, and security systems incorporate microprocessors in some way.

  Despite this, it is rare for electrical or electronics engineers to have knowledge about processor designing. The importance of microprocessors is evident in the wide range of products that rely on them, from televisions, VCRs, and DVD players to elevators, computer servers, and even some doors with automatic entry. Raising awareness about processor design can help bridge the gap and enable Pakistan to keep up with global advancements in this domain.
- Moore's law, proposed by Gordon Moore in 1965, predicted that the number of transistors on semiconductor chips would double approximately every 18 months. This implied that computing capabilities

Figure 1.7: Electronic Devices based on Microprocessor

Source: VectorStock

would become significantly faster, smaller, and more efficient over time. However, chip densities are no longer doubling every 18 months.

As we can see in this graph that although transistor size are almost going ideal but the clock frequency has kind of stopped at almost 5 GHz. Clock frequency is one of the major factors on which computational speed of a processor depends. Now to overcome this issue more number of cores are being implemented in a single processor but the issue in this method is one that number of cores per chip cannot exceed after a certain limit and the other is that it consumes more power. Power consumption is also a big hurdle in going advance in computer technology. So there is a need of an energy efficient and powerful processor or computer system (RISC based).

- Machine Learning is widely used nowadays to achieve artificial in-

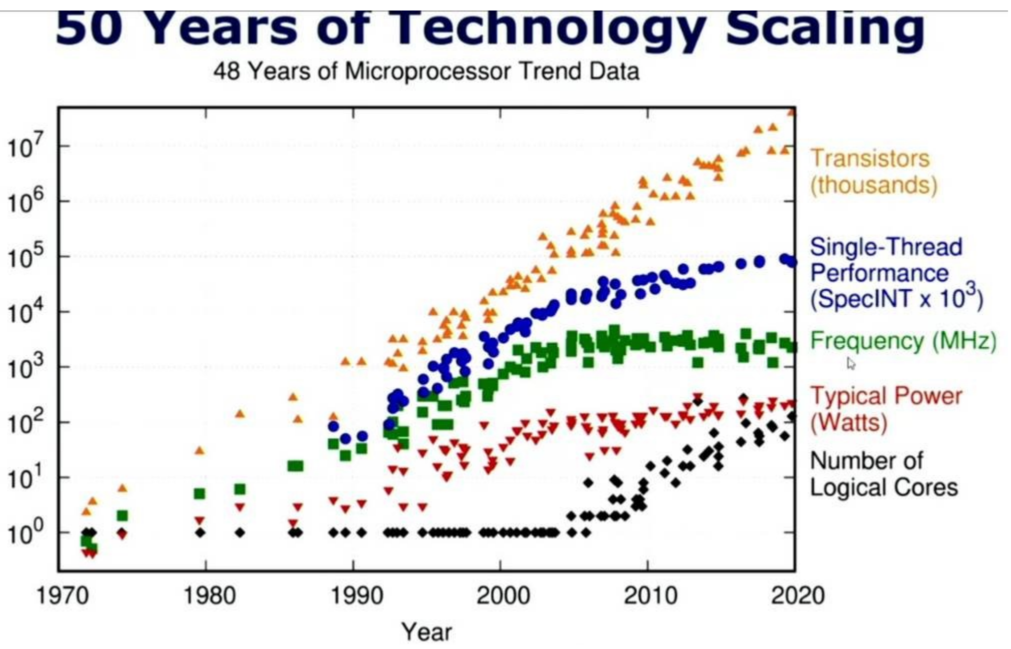


Figure 1.8: 50 Years of Technology Scaling

Source: Data Center Knowledge

telligence. There are a lot of applications of AI nowadays such as image recognition, object detection, image classification, in medical and military fields and a lot more. Normally PCs are use to achieve desire output. And nowadays CPUs and GPUs (Graphic Processing Units) are used in PCs. If we use CPU to do the required computations, then it takes a lot of time which most users don't like. On the other hand, GPUs are not affordable by small scale industries or individuals working on a project. Other than that, for space constraint applications like security camera in a military or sensitive area. The system for the computations needs to be placed near the security camera to detect the presence of any intruder. These applications require embedded platforms to do the computations in less time and using less space. So there is also a need for domain specific accelerator to run the ML algorithms faster than CPU and costs less than GPU.

1.4 Project Objectives

1. First objective is design and implementation of RV32IMC processor on FPGA. The HDL (hardware desciption language) used for this purpose is SpinalHDL. SpinalHDL is a language to describe digital hardware. It is used to generated VHDL/Verilog files. It is much more powerful than VHDL, Verilog, and SystemVerilog in its syntax and features Much less verbose than VHDL, Verilog, and SystemVerilog. It allows you to use Object-Oriented Programming and Functional Programming to elaborate your hardware and verify it. In this RISC-V processor descrption, RV32I is base instruction set, M extension is for multiplication and division instructions and C is for compressed instructions. Compressed instructions actually mean that instead of 32 bit of instruction size, most commonly used instructions will be of 16 bit width so that they occupy less instruction memory. RISC-V has many optional extensions but we are using only two extensions which are enough for our general-purpose processor design. We will test the processor by making software application in C language, then we can clearly see the result if our processor will show the results for exact same implementations which we will give in C language. The core we are using is named as VexRiscv and was developed by same developer as of SpinalHDL and is available on GitHub [1].

2. The second objective of the project is to implement a specialized machine learning accelerator for different ML applications, which will

increase the computational speed of those algorithms and provide faster results. The accelerator will take on ML related tasks from the processor and expedite the process, delivering results back to the processor with minimal delay. One example of a RISC-V accelerator designed for vision and AI edge applications is ztachip, which can operate on low-end FPGA devices or custom ASICs, making it a highly efficient and low-power solution for embedded systems. Compared to a non-accelerated RISC-V implementation, ztachip can provide acceleration capabilities of up to 20-50x for many vision and AI tasks. It even surpasses the performance of a RISC-V processor that has a vector extension. One of ztachip's standout features is its innovative tensor processor hardware, which enables it to accelerate a wide range of different tasks, including edge detection, optical flow, motion detection, color conversion, and the execution of TensorFlow AI models. This sets it apart from other accelerators that are limited to accelerating a narrow range of applications, such as convolutional neural networks.

3. In the last objective of our project, we aim to integrate the VexRiscv processor and ztachip (hardware accelerator) on an FPGA kit. One of the significant advantages of using an FPGA is its reconfigurability, even after the synthesis of a circuit. Moreover, an FPGA requires less board space and is more energy-efficient compared to an equivalent discrete circuit or ASIC (Application Specific Integrated Circuits). The kit used for this project is Genesys 2 Kintex-7 (xc7k325t-2ffg900C). Then the results generated by the integrated system will be compared with other accelerators out there in terms of speed, performance and memory usage.

1.5 Project Scope

1.5.1 RISC-V

RISC-V has bright future. RISC-V is now using in companies and companies are taking benefit from it to build custom processors for newer technologies' power and performance, for example ML (Machine Learning), AI (Artificial Intelligence), and IoTs (Internet of Things).

RISC-V is now using for edge computing to cloud servers and HPC (High-Performance Computing). However, companies are slowly going towards RISC-V for general-purpose processors which can be used in dashtops, laptops, and data centers. According to the RISC-V International Foundation, they expect the adoption of RISC-V over the entire CPU spectrum by 2025. Several companies and academic institutions are currently developing RISC-V-based processors that are expected to be released in the coming years, including ATSC, Cambridge, Esperanto Technologies, ETH Zurich, NVIDIA, and more. While current RISC-V technologies can't currently compete with x86 offerings from AMD and Intel, there's enough momentum for the open-source instruction set architecture to produce competitive hardware soon.

NASA and RISC-V:

Nowadays NASA is planning to use RISC-V power space computer for future space missions. SiFive, a well-known manufacturer of electronic devices based on RISC-V architecture, has signed an agreement with NASA to supply custom RISC-V based processors for the space agency's High-Performance Spaceflight Computer (HPSC). This product is made with the collaboration of SiFive with Microchip and contains 12 cores based on

RISC-V ISA and they are expected to offer 100 times more performance than NASA's previously used processor BAE RAD750. [2]

Figure 1.9: NASA selects RISC-V for Space Exploration

Source: EENews

This is a huge step up for the space industry and will allow us to explore the universe and discover many things in a more efficient way and much faster than before. The reason for the growing interest in RISC-V processors is due to their power-efficient design, which follows the same principles as ARM's proprietary cores. However, unlike ARM processors that require royalty payments each time they are incorporated into a chip, RISC-V offers a royalty-free alternative. On the other hand, we all know that RISC-V is a free, open-source ISA and is also flexible in nature. No wonder it is being used from embedded platforms to now in space exploration!

RISC-V with SiFive:

SiFive was established by the creators of RISC-V, who have been working on the RISC-V instruction Set Architecture (ISA) since 2010. Their goal is to empower big and small companies to innovate with the next generation of high-performance processors based on RISC-V. They enable the development of application-specific silicon faster than ever. SiFive has the capability to fully leverage the potential of the open-source RISC-V architecture on a large scale. [3]

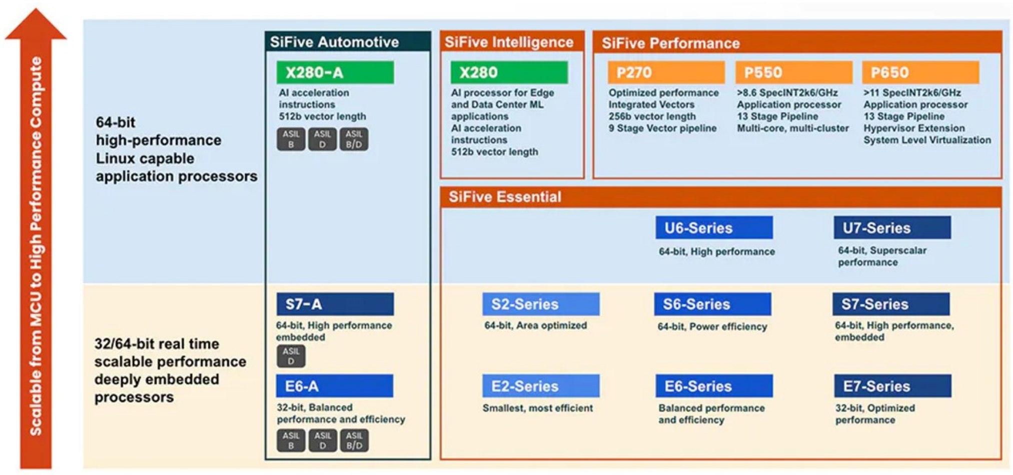

Scalable from MCU to High Performance Compute
64-bit high-performance Linux capable application processors
32/64-bit real time scalable performance deeply embedded processors
SiFive Automotive
X280-A
AI acceleration instructions 512b vector length
ASIL B
ASIL D
ASIL B/D
S7-A
64-bit, High performance embedded
ASIL D
E6-A
32-bit, Balanced performance and efficiency
ASIL B
ASIL D
ASIL B/D
SiFive Intelligence
X280
AI processor for Edge and Data Center ML applications
AI acceleration instructions
512b vector length
SiFive Performance
P270
Optimized performance
Integrated Vectors
256b vector length
9 Stage Vector pipeline
P550
>8.6 SpecINT2k6/GHz
Application processor
13 Stage Pipeline
Multi-core, multi-cluster
P650
>11 SpecINT2k6/GHz
Application processor
13 Stage Pipeline
Hypervisor Extension
System Level Virtualization
SiFive Essential
U6-Series
64-bit, High performance
U7-Series
64-bit, Superscalar performance
S2-Series
64-bit, Area optimized
S6-Series
64-bit, Power efficiency
S7-Series
64-bit, High performance, embedded
E2-Series
Smallest, most efficient
E6-Series
Balanced performance and efficiency
E7-Series
32-bit, Optimized performance

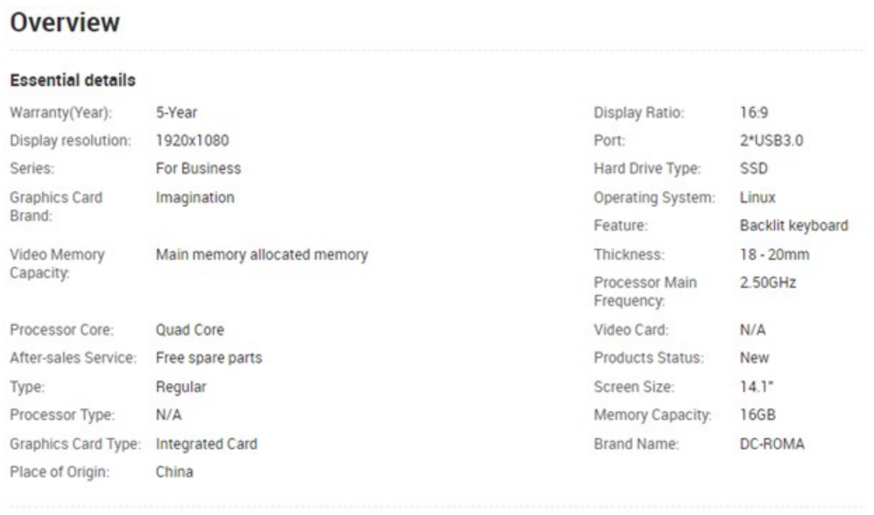
Figure 1.10: Description of four Distinct Processor Families of SiFive Core IP Portfolio based on the RISC-V ISA

Source: SiFive

SiFive is working on core IPs and have divided them into four families based on their specifications and necessities.

RISC-V based Laptops:

It is encouraging to see the increasing adoption and software compatibility of RISC-V processors. The fact that this open-source architecture is now being utilized in tangible products is a positive development. One such product is the Alibaba Roma RISC-V laptop [4], which was announced in the summer and has now become available. This laptop features a quad-core processor and a range of capabilities that are commonly found in Intel and AMD computers. The news was reported by CNX Software.



Overview

Essential details

Warranty(Year):	5-Year	Display Ratio:	16:9
Display resolution:	1920x1080	Port:	2*USB3.0
Series:	For Business	Hard Drive Type:	SSD
Graphics Card Brand:	Imagination	Operating System:	Linux
		Feature:	Backlit keyboard
Video Memory Capacity:	Main memory allocated memory	Thickness:	18 - 20mm
		Processor Main Frequency:	2.50GHz
Processor Core:	Quad Core	Video Card:	N/A
After-sales Service:	Free spare parts	Products Status:	New
Type:	Regular	Screen Size:	14.1"
Processor Type:	N/A	Memory Capacity:	16GB
Graphics Card Type:	Integrated Card	Brand Name:	DC-ROMA
Place of Origin:	China		

Figure 1.11: Specification of Alibaba Roma RISC-V Laptop

Source: Alibaba.com

Below is some description of RISC-V based laptop available for sale on Alibaba

RISC-V and Google:

As we all know that Google requires much computation power for their datacenters. According to SiFive they can fulfill their demand by providing a processor from their intelligence family. The processor of interest here is Intelligence X280 which is a multicore processor based on RISC-V ISA along with vector extensions. This processor is optimized for AI/ML applications and the datacenters of Google also used Machine Learning so this may be a good decision to use this processor. It can be combined with MXU (matrix multiplication units) which are used in the Google's TPUs (Tensor Processing Units). SiFive claims that it will deliver greater flexibility for programming and running machine learning workloads. [5]

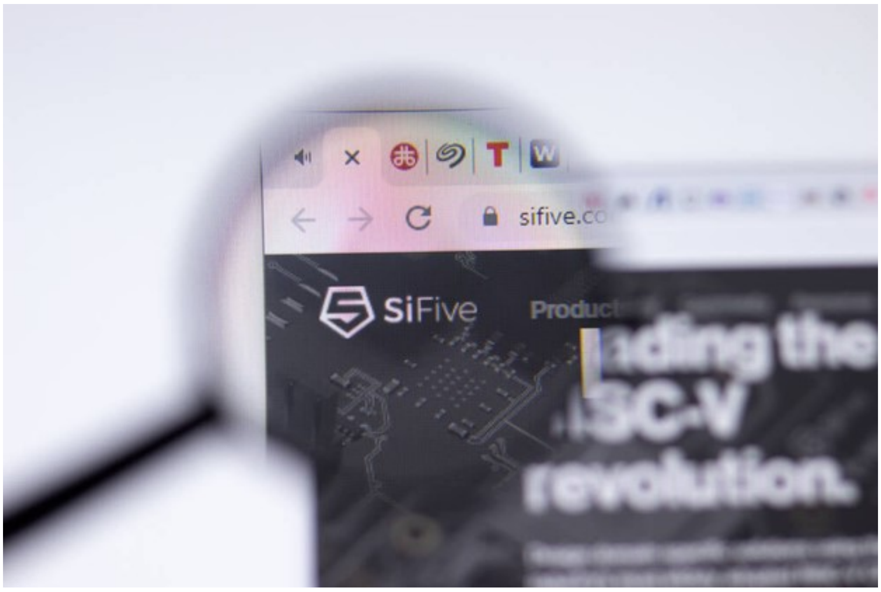


Figure 1.12: Google selected SiFive

Source: The Register

It has been heard that Google has placed its custom acceleration units in the same chip of SiFive (multicore X280) and it is directly connected to the Google designed MXU blocks. These RISC-V core complex chips are being used in the datacenters, according to SiFive, to speed up the machine learning work.

Additionally, it has been discovered that an all-encompassing software stack was necessary for handling the accelerator, and consumers came to the realization that they could address this concern by employing an X280 core complex in close proximity to their sizeable accelerator. The RISC-V CPU cores are responsible for supervising and executing maintenance and operation code, accomplishing mathematical operations that the large accelerator is incapable of performing, and furnishing diverse other functions. In essence, the X280 has the potential to function as a managerial hub for the accelerator.

RISC-V and China:

ARM recently stated that RISC-V poses no threat to them in datacenters, but China's interest in RISC-V could prove otherwise. Due to trade bans and restrictions imposed by the US on China, the country has been banned from importing chips from other countries, including x86 processors and AI kit from Intel, AMD, and Nvidia. [6]

Figure 1.13: China showing interest in RISC-V

Source: The Register

Chinese companies are increasingly adopting RISC-V as a means to quickly develop their own architecture, bypassing the need for proprietary solutions. This trend is reflected in the fact that a significant portion of RISC-V members are Chinese-based, and that the government-backed Academy of Sciences is actively engaged in developing high-performance, open-source RISC-V processors.

1.5.2 Machine Learning

Machine learning is a powerful tool nowadays and everyone use it such as Amazon, Netflix, Facebook, and so on. Machine learning is so versatile we can take unpredictable benefit from it.

In machine learning we don't need to remember anything, machine learning itself can memorize anything through experiences and trails. Machine learning is an application of Artificial Intelligence, create computer program and assist computer to memorize without the help of human interaction.

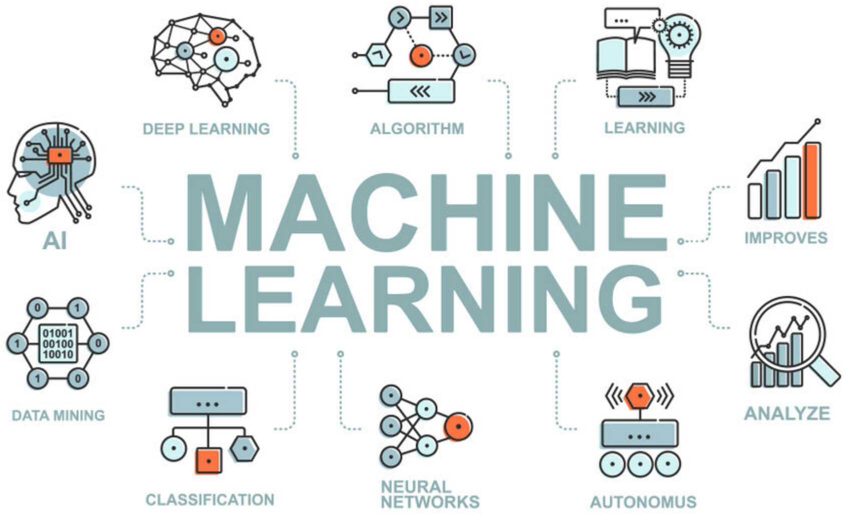


Figure 1.14: Machine Learning, Its Importance, Types, and Applications

Source: FORE School of Management

Machine learning plays a crucial role in the field of businesses as it allows entrepreneurs to understand customers' requirements and business functioning behavior. Nowadays many famous companies such as Amazon, Google, Facebook, and many more are professionally exploiting these technologies, and machine learning becoming a core operational part of functionality. The future of machine learning is quite bright and exciting.

Every common field is powered by machine learning applications, some of them are search engine, healthcare, digital marketing, and education. Machine learning is being so noticeable and leading in our lives today, it's challenging and difficult to imagine a future without it.

Chapter 2

Literature Review

2.1 Introduction

RISC-V, an open and modular Instruction Set Architecture (ISA), has the potential to revolutionize the Internet of Things (IoT), Artificial Intelligence (AI), and Machine Learning (ML) industries due to its clean design and open nature. Although RISC-V is not the first open processor implementation, it is associated with two keywords - freedom and innovation - as it offers users the freedom to use the cores however they wish. The emergence of RISC-V coincides with other industry events, such as the slowing of Moore's Law and the growth of machine learning, leading to the question of whether the timing is right for RISC-V's success.

Simon Davidmann, CEO and founder of Imperas Software, believes that the RISC-V architecture has gained popularity due to the need for flexibility in the hardware design process. Today's electronic products are defined by their functionality, much of which is dependent on software running on processors. As machine learning becomes increasingly important across all industries, the demand for computing power has skyrocketed, leading to a need for numerous processors configured in a customized manner. Standard off-the-shelf technologies are not sufficient to meet this demand for customization. Therefore, there is a growing demand for the freedom to design chips and processors, as well as the fabrics of processors that go into these chips.

RISC-V is gaining popularity among universities, with many creating open cores and incorporating the RISC-V ISA into their courses. For instance, the University of California, Berkeley has developed Rocket cores using RISC-V and is teaching the architecture to students in their EECS department. Likewise, ETH Zurich has created power-efficient cores like ibex, zero-riscy, micro-riscy, and others using their PULP platform. Several industry collaboration groups such as CHIPS Alliance, OpenHW Group, and SiFive are working together with both industry and academia to build open-source cores and make them widely available to the community.

2.2 Purpose of the Review

The purpose of this literature review is to let the readers know about the advancements to RISC-V in industrial area as well as in educational and innovational region. Also, we are going to compare a few examples of RISC-V ISA being used nowadays in many applications with our processor and accelerator and have a detailed discussion on their uses.

2.3 RISC-V based Pakistan's first processor

NUST [7] has Developed Pakistan's First Self-Developed Embedded Microprocessor NTiny-E. Electronic devices and consumer appliances rely heavily on semiconductor chips. In 2021, global semiconductor sales exceeded $556 billion. However, Pakistan has yet to tap into this lucrative market.

Figure 2.1: NUST developed Pakistan's First Processor based on RISC-V

Source: propakistani

In pursuit of its objective to promote applied research and innovation, the National University of Sciences and Technology (NUST) initiated an effort to make Pakistan self-sufficient in semiconductor technology. As a result of this endeavor, researchers at NUST have accomplished comprehensive functional testing of the NTiny-E microprocessor, which is the first-ever domestically developed microprocessor in the country, based on the RISC-V ISA. This microprocessor has been fabricated by Taiwan Semiconductor Manufacturing Company, Limited using a 65nm process node, and is aimed towards the market for embedded systems, consumer electronic products, and IoT devices.

Following is the table containing some of its specifications:

	Feature	**Description**
CORE BASE	RV32I	32-Bit Integer Base
Extensions	M	Integer Multiplication and Division
	F	IEEE Single-Precision Floating Point
Memory	IMEM	Instruction Memory 32KB
	DMEM	Data Memory 8KB
Peripherals	UART	Universal Async. Receiver Transmitter
	I2C	Inter Integrated Circuit
	SPI	Serial Peripheral Interface
	GPIO	General Purpose I/O - 16
	Timer	32-Bit 1x Timer
	PWM	Pulse Width Modulation-2x
	JTAG	IEEE complaint JTAG support for in-system programming
	Debug	OpenOCD Debug support

Table 2.1: Specifications of NTiny-E

Source: propakistani

2.4 UC Berkeley made RISC-V Chips

As we all know RISC-V was a project of UC Berkeley, California. They have also designed many open source cores but in we will be talking about a few of them which are prominent.

Rocket Chip is an open-source System-on-Chip design generator that can create a synthesizable RTL using Chisel HDL, a different approach to traditional hardware description languages. The output is a library of generators for cores, caches, and interconnects that can be integrated into a complete SoC. The generated processor cores are based on the RISC-V ISA, including both in-order (Rocket) and out-of-order (BOOM) core generators. Additionally, Rocket Chip can accommodate custom accelerators, such as instruction set extensions, coprocessors, or new independent cores. By using Rocket Chip, functional silicon prototypes that can boot Linux have been successfully produced. [8]

Another core generator of note is BOOM (Berkeley Out-of-Order Machine), which is a superscalar RV64G core generator designed to serve as a base implementation for research, industry, and education for extensive exploration of out-of-order microarchitecture. BOOM features an advanced load/store unit that enables loads to execute out of order with respect to other loads and stores, as well as store data forwarding to dependent loads. It is developed in Chisel HDL and consists of approximately 10,000 lines of code. [9]

A 45nm SOI process has been used to develop a 64-bit dual-core RISC-V processor with vector accelerators, making it the first dual-core processor produced using the open-source RISC-V ISA created at the University of California, Berkeley. When compared to ARM's similar single-issue in-order scalar core, the RISC-V scalar core surpasses in DMIPS/MHz by

10%, while being 49% more area-efficient, all made in a standard 40nm process. Furthermore, the vector accelerator in this RISC-V processor consumes 1.8× less energy than IBM Blue Gene/Q and 2.6× less than the IBM Cell processor, which were both made using the same process. This dual-core RISC-V processor has a maximum clock frequency of 1.3GHz at 1.2V and can achieve a peak energy efficiency of 16.7 double-precision GFLOPS/W at 0.65V, all within a compact 3mm area. [10]

2.5 Power Management Integrated RISC-V Processor SoC Operating at Submicrosecond Timescales

An efficient RISC-V system-on-chip (SoC) has been designed that combines voltage regulation, adaptive clocking, and power management in a single chip using a 28 nm fully depleted silicon-on-insulator process. The SoC features a simultaneous-switching switched-capacitor DC-DC converter that supplies an application core using a clock generated by an adaptive clock generator. The SoC is highly efficient, with conversion efficiency of 82%-89% and energy efficiency of 41.8 double-precision GFLOPS/W, while consuming up to 231 mW of power. The integrated power-management unit measures the state of the system and adjusts the voltage and frequency of the core, allowing the implementation of power-management algorithms that can respond at submicrosecond timescales. The system has a wide continuous voltage range of 0.45 V-1 V, and an adaptive voltage-scaling algorithm that can reduce energy consumption by 39.8% during synthetic benchmark testing. The system is effective at managing power consumption and has only 2.0% area overhead. [11]

2.6 RISC-V Processor Designed for Space Systems

A low-cost fault-tolerant implementation of the RISC-V architecture is proposed. The implementation employs physical and information redundancy to reduce error propagation, which is critical for space systems where reliability is paramount. Physical redundancy involves replicating the hardware components, while information redundancy involves duplicating the data and comparing it to detect errors. The proposed implementation achieves fault tolerance through several techniques. One of these is the use of a dual modular redundancy (DMR) technique, which involves replicating the processor's logic and executing the same instruction on both replicas simultaneously. The output of each replica is then compared to detect errors, and the correct output is selected.

In addition to DMR, the proposed implementation employs other fault-tolerant techniques, such as error-correcting codes (ECC) for memory and data paths, and parity checks for control logic. These techniques help to detect and correct errors in the system, making it more reliable.

The proposed implementation also achieves competitive silicon and power overheads when compared with other RISC-V implementations. This is important in space systems, where size, weight, and power consumption are critical factors. [12]

2.7 RISC-V and PULP

In 2013, the Integrated Systems Laboratory (IIS) at ETH Zürich and the Energy-efficient Embedded Systems (EEES) group at the University of Bologna joined hands to create the Parallel Ultra-Low-Power Processing

Platform (PULP). Its objective is to research and develop innovative and effective architectures for processing with ultra-low power consumption. [15] In the following discussion, we will delve into several of their cores and their specifications.

2.7.1 Ibex Core

To begin with, the Ibex RISC-V Core is a 32-bit CPU core designed in SystemVerilog and available as open-source. [16] This core is highly customizable and ideal for embedded control applications. It has undergone rigorous verification and has been used in multiple tape-outs. Ibex supports various extensions, including Integer (I) or Embedded (E), Integer Multiplication and Division (M), Compressed (C), and Bit Manipulation (B). The core's block diagram indicates a small parameterization with a 2-stage Pipeline.

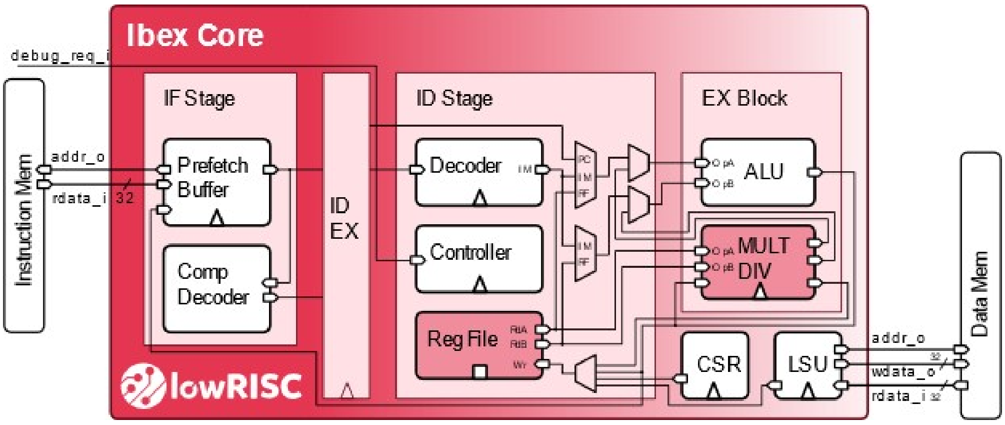


Figure 2.2: Ibex Core Block Diagram

Source: GitHub

This block diagram shows various stages of this CPU. We have IF stage whose job is to fetch the instruction from instruction memory and also to check if the instruction is compressed or uncompressed one.

Then we have ID stage whose job is to decode the given instructions and generate the control signals to perform the required task. It also contains GPR which stands for general purpose registers. There quantity may vary on the configuration being used. If we are using micro-riscy then the no. of registers here will be 16 and other than this configuration total registers will be 32. No matter what the configuration is the register size will remain the same which is 32 bits.

At last, we have EX block whose job is to execute the desired operation using ALU (arithmetic and logic unit) and MultDiv block for multiplication and division. Other than these main blocks it also contains CSRs which stands for Control and Status registers. These registers are used for different purposes including flags and interrupts etc. We also have LSU (load store unit) whose job is to store or load any data to or from the data memory respectively.

Here is a table that displays the performance and area of selected configurations that lowRISC is concentrating on for evaluating performance and verifying designs.

Config	”micro”	”small”	”maxperf”	”maxperf-pmp-bmfull”
Features	RV32EC	RV32IMC, 3 cycle mult	RV32IMC, 1 cycle mult, Branch target ALU, Writeback stage	RV32IMCB, 1 cycle mult, Branch target ALU, Writeback stage, 16 PMP regions
Performance (CoreMark/MHz)	0.904	2.47	3.13	3.13
Area - Yosys (kGE)	16.85	26.60	32.48	66.02
Area - Commercial (estimated kGE)	∼15	∼24	∼30	∼61
Verification status	Red	Green	Green	Green

Table 2.2: Comparison of Different PULP Cores

Source: GitHub

2.7.2 Zeroriscy Core

Now finally this is the core we have been working on as a starting point. As it can be seen in the previous table that it is derived from Ibex core and is also area and power efficient. Zeroriscy is a specific implementation of the RISC-V instruction set architecture, designed to be a small and efficient 32-bit processor core. It is an in-order processor with a 2-stage pipeline, meaning that it processes instructions in a linear sequence and has two stages of processing. [21]

The design of zeroriscy is configurable, allowing it to support four different ISA (instruction set architecture) configurations, which are determined by two parameters. This configurability allows the processor core to be tailored to specific use cases or requirements, such as optimizing for power efficiency or maximizing performance.

The RISC-V instruction set architecture, which zeroriscy is based on, is an open-source architecture that is gaining popularity in both academic and industrial settings. Its modular design and flexibility make it a suitable choice for a wide range of applications, from small microcontrollers to high-performance computing systems. Ibex contains an extra extension of Bit Manipulation (B). Interesting thing is that zeroriscy can further be configured and get smaller named as micro-riscy. The difference between two of them is that zeroriscy support I base extension containing 32 temporary core registers while micro-riscy supports E extension, short-form for Embedded, and only contains 16 base registers instead of 32. [17]

If we compare the power efficiency of all three cores then the figure below shows it in detail.

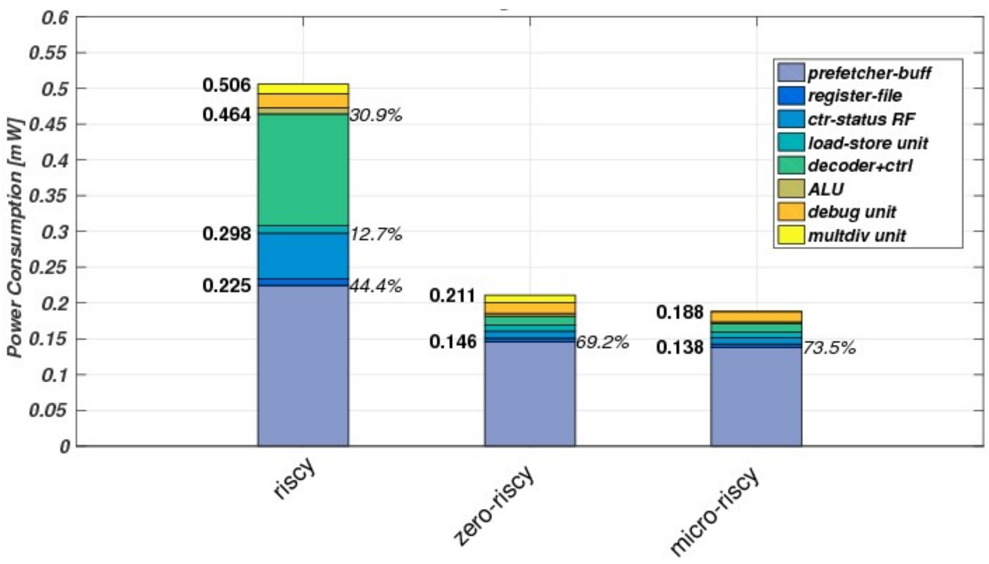


Figure 2.3: The Distribution of Power among the Three Core Units while Running Coremark at a Frequency of 100MHz and a Voltage of 1.2V

Source: Semantic Scholar

As we can see in this graph that they consume power in similar order as they are constructed. zero-riscy consums merely 0.211 mW of maximum power even if its multiplier is running. Now if we compare the area of all the other PULP cores then the figure below shows in detail.

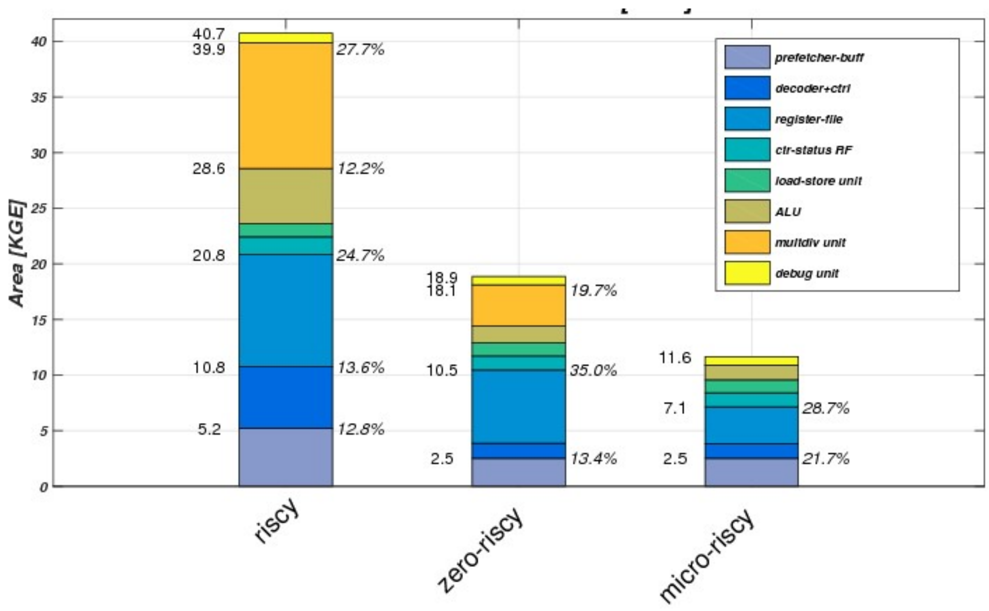


Figure 2.4: The Total Area Occupied by all the Core Units in KGE (thousands of square millimeters)

Source: Semantic Scholar

As we can see in the above graph that the total area covered by zero-riscy is only 18.9 KGE (Kilo Gate Equivalent). Now if we take a look at the figure below of its block diagram.

This block diagram shows different parts/modules of this core. For example pre-fetch buffer has the job of fetching instructions one by one from the instruction memory and either send it to ID stage or store it in a FIFO if exceptional cases occur. Debug unit is used to debug any error or issues related to the hardware of the processor. Decoder decodes the given 32 bit instruction and some part goes to the GPR (General Purpose

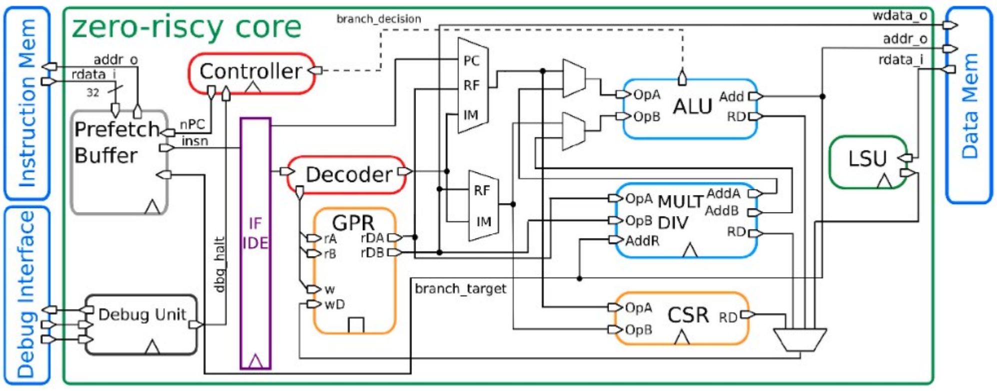


Figure 2.5: Block Diagram of zeroriscy Core

Source: Semantic Scholar

Register) and some goes to Controller to generate control signals. ALU has the capability to perform addition of any type of two operands whether registers, immediates or PCs and it also performs subtraction, shifting and other logical operations. MultDiv block performs, as depicts from the name, multiplication and division. It uses 3 cycle multiplier and 36 cycle divider. CSR (Control and Status Registers are special purpose registers. LSU stands for Load store unit and its job is to either store or load the date to or from memory respectively. This was all for now about zeroriscy core.

Supported Instruction Set:

Zeroriscy supports the following RISC-V instruction sets:

- RV32I Base Integer Instruction Set: It consists of fundamental instructions for performing integer arithmetic, logical operations, load and store operations, as well as control transfer instructions.

- RV32E Base Integer Instruction Set: This instruction set is a subset of the RV32I instruction set, designed for embedded systems with limited resources. It includes a smaller set of instructions and a smaller register file.

- RV32C Standard Extension for Compressed Instructions: This instruction set provides support for compressed instructions, which can reduce code size and improve memory bandwidth.
- RV32M Integer Multiplication and Division Instruction Set Extension: This instruction set provides support for integer multiplication and division operations.
- The RV32M and RV32E extensions can be enabled or disabled using two parameters, allowing for flexibility in configuring the processor core to meet specific requirements. Enabling the RV32M extension adds support for integer multiplication and division instructions, while enabling the RV32E extension switches to the RV32E instruction set, which is a more compact subset of the RV32I instruction set.

Overall, zeroriscy's support for these instruction sets and its configurability make it a versatile and efficient choice for a wide range of applications. [17]

Instruction Fetch:

If the memory or cache can also provide one instruction per cycle, the instruction fetcher can deliver one instruction to the next stage of the processor per cycle. It is important to note that the instruction address must be half-word-aligned to enable the use of compressed instructions. [18]

To further optimize performance, a prefetcher is used to fetch instructions ahead of time so that they are ready when needed.Jumping to an instruction address with the least significant bit set is not feasible.

Overall, these signals and requirements help ensure that instructions are fetched efficiently and correctly, allowing the processor to operate at peak performance.

Signal	**Direction**	**Description**
instr_req_o	output	Request ready, must stay high until instr_gnt_i is high for one cycle
instr_addr_o[31:0]	output	Address
instr_rdata_i[31:0]	input	Data read from memory
instr_rvalid_i	input	instr_rdata_is holds valid data when instr_rvalid_i is high. This signal will be high for exactly one cycle per request.
instr_gnt_i	input	The other side accepted the request. instr_addr_o may change in the next cycle

Table 2.3: Instruction Fetch Signals

Source: zero-riscy User Manual

Load-Store-Unit (LSU):

The Load-Store Unit (LSU) is a component of a processor core that facilitates access to data memory, where the processor stores the data that it needs to operate on. It is specifically designed to handle data of varying sizes, such as 32-bit words, 16-bit half-words, and 8-bit bytes.

A ”load” operation in the LSU retrieves data from memory and stores it in a register within the processor. This operation is used when the processor needs to read data from memory, for example, to read the value of a variable.

A ”store” operation in the LSU writes data to memory. This operation is used when the processor needs to update the value of a variable or store new data in memory.

The LSU supports different data sizes to allow for more efficient memory access. For example, when the processor only needs to read or write 16 bits of data, it can use a half-word load or store operation, which is faster than a full word operation. Similarly, a byte load or store operation can be used when the processor only needs to access 8 bits of data. [18]

Signal	Direction	Description
data_req_o	output	Request ready, must stay high until data_gnt_i is high for one cycle
data_addr_o[31:0]	output	Address
data_we_o	output	Write Enable, high for writes, low for reads. Sent together with data_req_o data_be_o[3:0] output Byte Enable. Is set for the bytes to write/read, sent together with data_req_o
data_wdata_o[31:0]	output	Data to be written to memory, sent together with data_req_o
data_rdata_i[31:0]	input	Data read from memory
data_rvalid_i	input	data_rdata_is holds valid data when data_rvalid_i is high. This signal will be high for exactly one cycle per request.
data_gnt_i	input	The other side accepted the request. data_addr_o may change in the next cycle

Table 2.4: LSU Signals

Source: zero-riscy User Manual

Register File:

The zeroriscy processor core is based on the RISC-V architecture and provides either 31 or 15 32-bit wide registers, depending on whether the RV32E extension is activated. Register x0 is permanently assigned to the value 0 and cannot be modified.

Zeroriscy has two types of register files: a latch-based register file and a flip-flop based register file. The former is suggested for ASICs, while the latter is recommended for FPGA synthesis. However, both types of register files are suitable for either type of synthesis.

When implemented in an ASIC, the flip-flop based register file is considerably larger compared to the latch-based register file. This is because ASICs are optimized for power efficiency and the latch-based register file consumes less power than the flip-flop based register file. On the other hand, in FPGA synthesis, power consumption is not a major concern, and the flip-flop based register file is suggested as it exhibits superior performance and timing characteristics.

In general, the selection of register file type relies on the target platform and implementation details. Even though both register file types work with either synthesis target, the latch-based register file is more power-efficient for ASICs. In contrast, the flip-flop based register file has better timing characteristics and performance, which makes it a suitable option for FPGA synthesis. [18]

2.8 RISC-V Accelerator for Post-Quantum Cryptography

Post-Quantum Cryptography (PQC) introduces new mathematical elements and operations that are challenging to implement on standard processors, particularly for low-cost and resource-constrained devices. Hardware acceleration is typically required to support PQC on these devices. Additionally, the ongoing standardization process of PQC requires maintaining flexibility in the design. Although co-design techniques that incorporate hardware and software have been utilized to create intricate and personalized post-quantum cryptography (PQC) solutions, the development of tightly integrated accelerators and Instruction Set Architecture (ISA) extensions for PQC has not been extensively investigated.

The proposal of the authors is the RISQ-V, which is an improved version of the RISC-V architecture. It integrates tightly coupled accelerators that enhance the performance of lattice-based PQC. The RISQ-V design optimizes processor resources and reduces the memory access needed, resulting in increased performance with minimal silicon area overhead. [13]

2.9 RISC-V Accelerator designed for Ascon

ASCON-p is a cryptographic module that has a low weight and can perform various cryptographic tasks such as hashing, authenticated encryption, and pseudorandom number generation. It is based on a permutation, which is a mathematical operation that shuffles the input data in a way that is difficult to reverse.ASCON-p is a cryptographic building block that has multiple applications, including authenticated encryption, hashing, and pseudorandom number generation. One of its implementations is ASCON, which is a lightweight authenticated encryption algorithm and won the lightweight category of the CAESAR competition. Another implementation is ISAP, an AEAD (Authenticated Encryption with Associated Data) scheme that uses ASCON-p and is currently a competitor in the NIST lightweight cryptography standardization project.

The researchers have implemented ASCON-p as an instruction extension for the RISC-V processor, which is a popular open-source processor architecture used in many embedded devices. This instruction extension is tightly coupled to the processor's register file, which means it does not require any dedicated registers. With this implementation, cryptographic computations can be performed with high performance, and the performance is even better if protection against fault attacks and power analysis is desired.

The researchers have also shown that their implementation is very efficient and requires only a small amount of area, making it suitable for low-end embedded devices like 32-bit ARM Cortex-M or RISC-V microprocessors. The authors conducted a performance evaluation of their implementation, utilizing the AEAD modes and hashing algorithms of the ISAP and ASCON families. Their results demonstrated that cryptographic computations can be executed at a performance rate of approximately 2 cycles per byte. However, if protection against fault attacks and power analysis is required, the performance rate increases to about 4 cycles per byte.

Finally, the researchers have analyzed the implementation security of ISAP when implemented using their instruction extension. They have shown that their implementation provides strong protection against a large class of implementation attacks such as DPA, DFA, SFA, and SIFA. This means that the cryptographic computations performed using their implementation are secure against these attacks, which are commonly used to break cryptographic systems. [14]

Chapter 3

RISC-V Processor

3.1 Introduction

RISC-V is a contemporary Instruction Set Architecture (ISA) with a clean and modular design that has the potential to become a significant player in the era of Artificial Intelligence (AI), Machine Learning (ML), and the Internet of Things (IoT) due to its open nature. Although it is not the first time a processor implementation or ISA has been declared open for public use, RISC-V is associated with two key concepts - freedom and innovation. Some users seek freedom to use cores, while others desire the freedom to modify them as needed. The increasing popularity of RISC-V coincides with the slowing of Moore's Law and the remarkable growth of machine learning. Is this just a fortuitous coincidence?

According to Simon Davidmann, the CEO of Imperas Software, the RISC-V architecture has gained popularity due to the demand for greater freedom in the hardware design process community. In today's electronic products, which heavily rely on software running on processors, machine learning is essential. This has led to a need for numerous processors, and the ability to configure them in a customized manner. People realized that off-the-shelf technologies were not sufficient, and they required more freedom to design chips and their processors. Therefore, there has been a shift in the electronic product marketplace that seeks greater freedom in architecting chips and the processors that operate within them.

Various academic institutions have contributed to the development and education of the RISC-V ISA. Some have created open-source cores and are incorporating RISC-V into their courses. The University of California, Berkeley, for instance, has designed Rocket cores based on RISC-V and is teaching the ISA to its EECS students. Similarly, ETH Zurich has developed power-efficient cores such as ibex, zero-riscy, micro-riscy, and others through its PULP platform. Collaboration groups involving both academia and industry, such as CHIPS Alliance, OpenHW Group, and SiFive, are also working on building open-source cores and making them available to the wider community.

3.2 VexRiscv

VexRiscv is an open-source RISC-V processor implemented in the SpinalHDL language, which can be synthesized for various FPGA platforms. The version of VexRiscv processor used in this project is MyVexRiscv, with some modifications and additions to VexRiscv's default configuration. [1]

MyVexRiscv has a simple, yet powerful architecture. It is a 32-bit RISC-V processor that supports the RV32IMC instruction set, which includes the basic integer arithmetic and logic operations, load and store instructions, control transfer instructions, multiplication and division capability, compressed instructions, and system calls. The processor also includes additional instructions provided by the VexRiscv plugins, such as barrel shifter instructions, early branching etc.

The processor is highly configurable, with a set of CPU plugins that can be added or removed depending on the user's needs. These plugins include a cached instruction bus, a cached data bus, a CSR (Control and Status Register) plugin, a simple decoder plugin, a memory translator plugin, a register file plugin, an integer ALU plugin, a source plugin, a full barrel shifter plugin, a hazard plugin, a multiplication plugin, a division plugin, a branch plugin, and a YAML plugin.

The processor also includes peripherals such as an APB (Advanced Peripheral Bus) controller, an AXI (Advanced eXtensible Interface) controller, a JTAG (Joint Test Action Group) controller, a timer, a prescaler, and a DDR (Double Data Rate) SPI (Serial Peripheral Interface) master controller.

The processor is designed to be synthesized for FPGAs, with a simple interface that allows it to be easily integrated into larger systems. The processor's clock domain can be configured to run at a maximum frequency of 166 MHz. It works on little endian memory storage with 8K of instruction and data cache.

Overall, MyVexRiscv is a flexible and powerful RISC-V processor that can be customized to meet a wide range of embedded system design requirements. It provides a robust platform for developing FPGA-based systems with RISC-V processors, and is well-suited for applications such as robotics, machine learning, and embedded vision.

3.2.1 Performance Score Comparison

MyVexRiscv processor has achieved benchmark scores of 1.38 DMIPS/Mhz and 2.57 Coremark/Mhz. These scores are with configuration of single cycle barrel shifter, debug module, catch exceptions, dynamic branch prediction in the fetch stage, branch and shift operations done in the Execute stage These scores indicate the performance of the processor in terms of instructions executed per second (DMIPS) and its ability to handle real-world workloads (Coremark).

While these scores may not be the highest in the industry, they still demonstrate that MyVexRiscv is a capable processor that can handle a wide range of tasks. Additionally, it's important to note that benchmarks are just one measure of a processor's performance, and real-world performance can vary based on a variety of factors such as the specific workload, memory configurations, and software optimizations.

Following is the table which includes benchmark scores [22] of other

RISC-V and ARM cores for comparison with VexRiscv:

Processors:	**Coremark/MHz**	**DMIPS/MHz**
MyVexRiscv (RV32IMC ISA)	2.57	1.38
Cortex-M0+ (ARMv6-M ISA)	1.8	0.95
Cortex-M4 (ARMv7-M ISA)	2.76	1.25
SiFive E21 (RV32IMAC ISA)	3.1	1.38
SiFive E20 (RV32IMC ISA)	2.4	1.1

Table 3.1: Benchmark Score Comparison of Different Processors

Source: RISC-V Foundation

Chapter 4

RISC-V Accelerator for ML Applications

4.1 Introduction

Ztachip is a hardware accelerator designed for low-end FPGA devices or custom ASICs, which aims to provide significant performance improvements for vision and AI edge applications. It has the ability to speed up various tasks, such as executing TensorFlow AI models and performing common vision tasks, such as motion detection, color conversion, optical flow, and edge detection.

Ztachip's acceleration is achieved through the implementation of an innovative tensor processor hardware, which allows for massive processing and data parallelism. This tensor processor hardware is capable of performing a range of tasks, and it is one of the key differences between ztachip and other accelerators that typically only accelerate a narrow range of applications, such as convolution neural networks.

To help programmers leverage the processing power of ztachip's tensor processor, a new tensor programming paradigm has been introduced. This programming paradigm enables programmers to write code that takes advantage of the parallelism provided by the tensor processor, making it easier to develop high-performance applications.

Overall, ztachip is a powerful hardware accelerator that is designed to provide significant performance improvements for vision and AI edge applications. It is capable of accelerating a wide range of tasks and includes an innovative tensor processor hardware and a new tensor programming paradigm to help programmers take full advantage of its processing power. [19]

4.2 Ztachip as DSA

Ztachip is an open-source DSA (Domain-Specific Architecture) architecture that is designed to be versatile and support a wide range of applications beyond just AI. It is a novel architecture that provides significant performance improvements for applications that can be expressed as a sequence of tensor operations.

- One of the primary objectives of ztachip is to make DSA programming simple and intuitive. To achieve this, ztachip includes a wide range of tensor operations, including data operations and computing operations. These tensor operations can be used to perform complex tasks, such as tensor transpose, tensor dimension resize, and data remapping, in addition to more traditional tensor operations.

- By supporting a broad range of tensor operations, ztachip aims to be a highly flexible and versatile DSA architecture that can be used to accelerate a wide range of applications. This makes it an attractive option for developers and researchers looking to build high-performance computing systems that can handle a diverse set of workloads.

- Overall, ztachip is a novel and versatile DSA architecture that is designed to provide significant performance improvements for a broad range of applications. Its support for a wide range of tensor operations makes it an attractive option for developers and researchers looking to build high-performance computing systems that can handle complex workloads beyond just AI.

- An important benefit of this method is the streaming-based memory transfer between external memory and the system, which involves prefetching and lacks round trip delays. As a result, system latency is reduced and hardware resources are utilized more effectively.

- A benefit of tensor data operations is that they provide specific data requirements for later execution, removing the need for caching. As a result, memory usage can be reduced and system performance can be improved.
- The ztachip utilizes tensor operators to represent computing operations, which enables algorithm parallelism and efficient mapping of multiple hardware threads to numerous parallel tasks. This approach improves the overall efficiency of the system.
- In ztachip, tensor computing is designed to only use internal memory, which results in a simpler hardware design. Additionally, this approach eliminates memory stall cycles, enabling the system to operate more efficiently and provide faster processing times.

Overall, the decoupling of data plane and computing operations in ztachip provides many advantages for the hardware design, including improved performance, reduced memory usage, and simplified design. [20]

4.2.1 Limitations

There are several challenges associated with using Domain-Specific Architectures (DSA), despite the fact that they can provide higher efficiencies for specific sets of applications. One of the main challenges is achieving diversity within the DSA domain while still maintaining efficient hardware implementation. One of the challenges in introducing DSA concepts to users is avoiding the need for cross-disciplinary knowledge, as it can be challenging to find individuals who possess both hardware design and software engineering skills.

To address these challenges, it is important to provide clear and concise documentation that explains the concepts and benefits of DSA, as well as providing examples and tutorials that demonstrate how to use DSA in practice. Additionally, collaboration between hardware and software engineers can be useful in developing DSA, as it can help to bridge the gap between the two disciplines and ensure that both hardware and software components are optimized for performance.

The Systolic-Array (SA) is a widely utilized DSA architecture that is specifically suited for numerous fundamental mathematical operations required in artificial intelligence, including convolution, dot product, and matrix multiplication. SA can be implemented in hardware to achieve high efficiency, and can also be integrated with software tools to make it easier for software engineers to use in their applications. [20]

4.3 Ztachip Software Stacks

ztachip provides several DSA components that make it easier for users to take advantage of its hardware acceleration capabilities.

The RTL source codes for ztachip's hardware stack are provided, allowing for easy portability to various FPGA and ASIC platforms. This allows developers to customize the design and optimize it for their specific use cases.

The second aspect of ztachip's approach is to offer a compiler that supports the implementation of a Domain Specific Language (DSL), which hides the intricacies of the hardware from users. This means that software developers do not need to possess extensive knowledge of the hardware aspects, and the same software can be easily ported to various hardware with different capacities by simply recompiling it. This makes it easier to develop software that can take advantage of the hardware acceleration provided by ztachip.

In addition, ztachip offers a software stack that incorporates various vision and AI algorithms. This software stack includes support for TensorFlow without the need for retraining. This makes it easier for developers to use the hardware acceleration provided by ztachip to accelerate their applications without having to implement the algorithms from scratch. [20]

Overall, the combination of the hardware stack, compiler, and software stack provided by ztachip makes it easier for developers to take advantage of the hardware acceleration capabilities of ztachip, without requiring detailed knowledge of the underlying hardware.

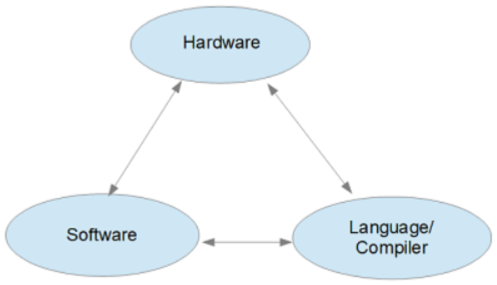


Figure 4.1: Methodology

Source: GitHub

4.4 Hardware Architecture

Ztachip is an accelerator designed specifically for accelerating machine learning workloads. It features a unique hardware architecture that incorporates a matrix of processing elements, with each element optimized for performing MAC (Multiplication and Accumulation) operations efficiently. The accelerator utilizes systolic array architecture, which enables parallel and pipelined execution of 1x8 MAC operations.

The processing elements in ztachip are arranged in a two-dimensional array, and the data flows through them in a pipeline fashion, resulting in high throughput and low latency. Additionally, ztachip has a dedicated memory system that minimizes the data movement required to perform MAC operation, further enhancing its performance.

The ztachip architecture also supports various data types and precision, making it versatile enough to handle a wide range of machine learning workloads. The accelerator can be connected to a host processor through standard interfaces, such as AXI, allowing for easy integration into existing systems. Overall, ztachip provides a highly efficient hardware solution for accelerating machine learning workloads, which can significantly speed up the training and inference phases of machine learning applications. [23]

Given below is the block diagram of its hardware architecture:

Now following is the detailed explanation of each of its component: The ztachip's hardware architecture is designed to accelerate tensor operations and consists of several subcomponents, such as the axilite, sram_core, ddr_rx, ddr_tx, core, and dp_core. The axilite_* interface connects the dp_core with RISC-V processor via the AXILite bus protocol, while the axi_* interface is used by ztachip to initiate DMA (Direct Memory Access) memory transfer to or from external memory.

The sram_core subcomponent acts as a temporary memory block to hold data during tensor data transfer. The ddr_rx and ddr_tx subcomponents are responsible for handling DMA transfer from external DDR memory to the core's internal memory and vice versa. The core is the tensor arithmetic execution unit and comprises an array of lightweight VLIW processors, while the dp_core is the central tensor processor unit

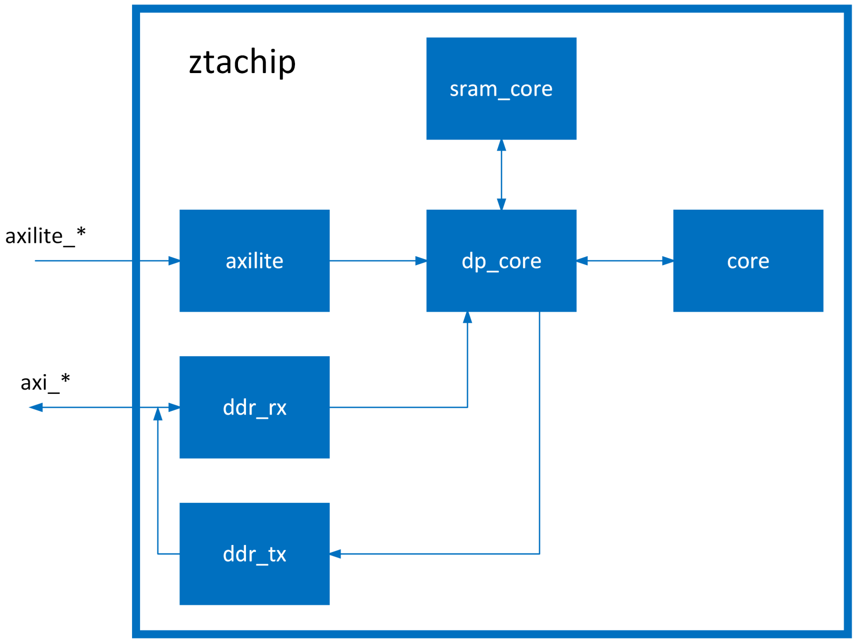


Figure 4.2: ztachip Hardware Architecture

Source: GitHub

that coordinates all activities within ztachip, including memory transfer and launching execution on the VLIW processor array.

The dp_core receives tensor instructions from RISC-V processor via the axilite_* interface and then coordinates tensor data operations to transfer data between the sram_core, core's internal memory, and external DDR memory. Additionally, the dp_core dispatches tensor operator execution requests to the core, which then dispatches the execution to the VLIW processor array. Overall, the ztachip hardware architecture is designed to efficiently execute tensor instructions, transfer data between memory and processors, and coordinate the execution of tensor operators. Now we will further explain each of the sub-components within ztachip. [23]

dp_core:

The dp_core is a central tensor processor unit within ztachip that coordinates all activities, including memory transfer and launching execution on the VLIW processor array. It receives tensor instructions from RISC-V processor via the axilite_* interface and executes them by coordinating tensor data operations, dispatching tensor operator execution requests to the core, and performing other complex functions.

To perform these functions, dp_core uses various interfaces and subcomponents. For instance, it uses bus_* to receive tensor instructions from RISC-V processor and task_* to send tensor operator execution instructions to the core. It also has readmaster1*, readmaster2*, and readmaster3* buses to receive DMA data transfers from different sources, and writemaster1*, writemaster2*, and writemaster3* buses to send DMA data transfers to different destinations.

Additionally, dp_core has several subcomponents, including dp_fetch, dp_gen_core, dp_source, and dp_sink. These subcomponents work together to decode tensor instructions, generate memory addresses for data transfer, generate DMA transfer read requests to the source of tensor data operations, and generate DMA transfer write requests to the destination point of tensor data operations.

Overall, dp_core is the main tensor processor of ztachip, and it performs a range of functions, including receiving tensor instructions from RISC-V processor, coordinating tensor data operations, dispatching tensor operator execution requests to the core, and ensuring that all memory transfers with the core's internal memory are completed before tensor operator execution can be issued. By using hardware threads, it can execute multiple data operations at the same time, and it can also execute tensor instructions out-of-order while still enforcing application-enforced order if needed. [23]

core:

The core's internal architecture consists of interfaces and subcomponents that work together to execute tensor operator tasks. The dp_write* interface receives DMA data transfer to the core's internal memory, while the dp_read* interface sends DMA data transfer from the internal memory. The task* interface is used to receive tensor operator execution commands from the dp_core component.

There are several subcomponents that make up the core internal architecture. The stream processor is responsible for data mapping between input and output. Two stream processors are used to perform data mapping before data is written to the core's internal memory and as data is retrieved from the memory.

The cell subcomponent groups 4 pcores together to improve the fan-out performance of bus signals. The pcores are multi-threaded processors with 16 hardware threads executing in a round-robin fashion. The instr subcomponent is the master processor for all pcore's VLIW processor cores that are simply ALUs running in locked step mode with each other.

The instr.instr_fetch component performs thread scheduling of execution and fetches VLIW instruction code from the instr.rom component. All VLIW cores share the same instruction code, and only one instruction is fetched for all the VLIW processors at every clock. [23]

The core internal architecture is responsible for all tensor operator execution tasks, and requests for tensor operator execution are received from the dp_core component via the task* interface. The instr subcomponent controls the execution of all pcore's VLIW processors, which are lightweight processors that are mostly just ALU with memory running in lock-step mode with each other. The VLIW instructions executed in these VLIW processors are 128 bit long with the following format:

Parameter	**Length**	**Description**
MU	N/A	pcore.alu instruction
oc	5	opcode
save	1	Save alu. y_out=>xregister_file
x3	N/A	x3 parameter; map to alu.xscalar_in
vector	1	1 if parameter is vector, 0 if scalar
addr	12	Internal memory address of x3
attr	4	Attribute
x1	N/A	x1 parameter; map to alu.x1_in
vector	1	1 if parameter is vector, 0 if scalar
addr	12	Internal memory address of x1
attr	4	Attribute
x2	N/A	x2 parameter; map to alu.x2_in
vector	1	1 if parameter is vector, 0 if scalar
addr	12	Internal memory address of x2
attr	4	Attribute
y	N/A	y parameter; map to alu.y*_out
vector	1	1 if parameter is vector, 0 if scalar
addr	12	Internal memory address of y
attr	4	Attribute
IMU	N/A	pcore.ialu instruction
oc	5	opcode
x1	4	x1 parameter; map to ialu.x1_in
x2	4	x2 parameter; map to ialu.x2_in
y	4	y parameter; map to ialu.y_out
const	13	Constant field
CTRL	N/A	CTRL
oc	5	Branching opcode based on IMU.y value
addr	11	Code address to jump to

Table 4.1: VLIW Instruction Format

Source: GitHub

The main ALU block of the ztachip's core component resides in the pcore subcomponent. It comprises various interfaces and functions. The interfaces include x1_in, x2_in, xreg_in, xscalar_in, y_out, y3_out, and y2_out. These interfaces serve to transfer data to and from the ALU block, such as input values, accumulator input, and results.

The ALU block is designed to be simple and to perform linear arithmetic operations for AI and vision processing tasks. Other non-linear operations are taken care of by stream. Together, ALU and stream cover a wide range of AI and vision processing. [23]

The ALU block includes arithmetic blocks such as multiplication of two 12 bit values, adding the result of multiplication to a 32-bit accumulator value, performing shift operation, performing boolean comparison of the result, and casting the 32-bit result to a 12 bit result. Following opcodes are supported in order to execute different operations through ALU block:

- COMPARE_LT : y2_out=(y_out < 0)?1:0
- COMPARE_LE : y2_out=(y_out <= 0)?1:0
- COMPARE_GT : y2_out=(y_out > 0)?1:0
- COMPARE_GE : y2_out=(y_out >= 0)?1:0
- COMPARE_EQ : y2_out=(y_out==0)?1:0
- COMPARE_NE : y2_out=(y_out!=0)?1:0
- MULTIPLY : y_out=x1*x2
- ADD : y_out=x1+x2
- SUBTRACT : y_out=x1-x2
- FMA : xreg_in += x1_in*x2_in

- FMS : xreg_in -= x1_in*x2_in
- ASSIGN : y_out=x1_in
- ACCUMULATOR_SHL : y_out=(xreg_in << x_scalar_in)
- ACCUMULATOR_SHR : y_out=(xreg_in >> x_scalar_in)
- INT12 SHR : y_out=(x1_in >> x_scalar_in)
- INT12 SHL : y_out=(x1_in << x_scalar_in)

Chapter 5

Integration of RISC-V Processor with ML Accelerator on FPGA

5.1 Introduction

FPGAs are a type of semiconductor device that come equipped with programmable interconnects and configurable logic blocks (CLBs), which can be used to create customized digital circuits.

Unlike Application Specific Integrated Circuits (ASICs), which are designed for specific tasks and are manufactured to meet those requirements, FPGAs can be reprogrammed after manufacturing to meet changing design needs. This makes them a flexible option for a wide range of applications, from digital signal processing to artificial intelligence and machine learning.

FPGAs are typically based on volatile memory technologies such as Static Random Access Memory (SRAM), which allows for reprogramming of the device during operation. This is in contrast to one-time programmable (OTP) FPGAs, which can only be programmed once and are more limited in their flexibility.

FPGAs have a range of advantages over ASICs, including faster time-to-market, lower development costs, and greater flexibility in design. However, they also have some drawbacks, such as higher power consumption and lower performance compared to ASICs for certain applications.

5.2 Advantages of using FPGA forIimplementation

FPGAs are often used to implement systems that need to be updated or reprogrammed to support new functionality or algorithms. The use of FPGAs allows manufacturers to create hardware that can be easily reconfigured to support different applications or algorithms, without having to create entirely new hardware designs.

For instance, Microsoft using FPGAs in its data centers for Bing search is a great example of this. Bing search algorithms require high-speed search capabilities to quickly and efficiently search through massive amounts of data. FPGAs are well-suited for this task because they can be reprogrammed to support new search algorithms or optimizations as they are developed.

By using FPGAs, Microsoft can make updates to its Bing search algorithms without having to modify or replace the underlying hardware. This provides a significant advantage in terms of speed and flexibility, as FPGAs can be reconfigured much faster than developing and manufacturing entirely new hardware.

In addition, FPGAs can be more power-efficient than general-purpose processors for certain tasks, such as digital signal processing or high-speed data processing. This can lead to cost savings and better performance in certain applications.

Overall, the use of FPGAs allows for a high degree of flexibility and performance in a wide range of applications, making them a popular choice in many industries, including data centers, aerospace, telecommunications, and more.

5.2.1 Genesys 2

The FPGA board used for this project is Genesys 2 by Digilent. The Genesys 2 FPGA kit is a development board designed for digital system designers and engineers who are looking to prototype and implement high-speed digital systems. The board is built around the Xilinx Kintex-7 FPGA, which is a powerful and versatile field-programmable gate array. The FPGA provides a flexible and reconfigurable platform for implementing a wide range of digital systems.

The Genesys 2 FPGA kit includes a number of peripherals and interfaces that make it easy to integrate with other systems. These include gigabit Ethernet, USB, HDMI, and a variety of analog inputs and outputs. The board also includes a number of memory resources, including DDR3 SDRAM, QSPI flash memory, and a microSD card slot.

The kit is designed to be used with the Xilinx Vivado development environment, which provides a comprehensive set of tools for designing and implementing FPGA-based digital systems. The development environment includes a hardware design language, simulation tools, and a synthesis tool that converts the design into a configuration file that can be loaded onto the FPGA.

One of the key features of the Genesys 2 FPGA kit is its high-speed interfaces. The board is capable of handling data rates of up to 10 Gbps, making it ideal for applications that require high-speed data transfer. This includes applications in the telecommunications, networking, and high-performance computing industries. [24]

Overall, the Genesys 2 FPGA kit is a powerful and versatile development board that provides designers and engineers with a flexible platform for prototyping and implementing high-speed digital systems. With its high-speed interfaces and range of peripherals, it is well-suited to a wide range of applications in the digital design space.

The reason for choosing this board is that it includes a VGA connector, which is a standard video connector used to connect a display to a computer. The VGA connector is capable of transmitting analog video signals and can support resolutions up to 1920x1200 at 60Hz. The VGA interface is controlled by the onboard FPGA, which can generate the necessary video signals to display images or video on a connected monitor or display. It can be used to display results of running different ML applications.

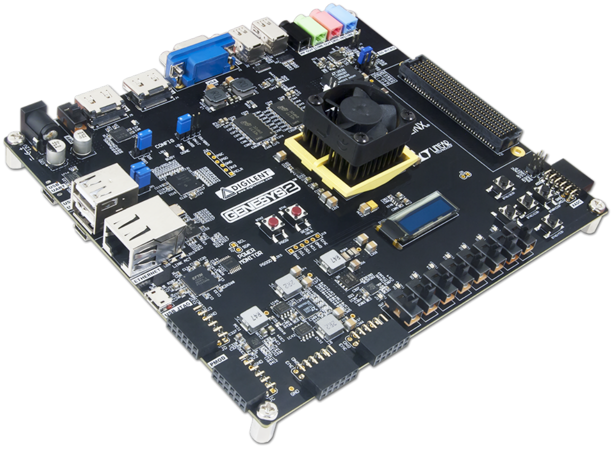


Figure 5.1: Genesys 2 FPGA Board

Source: Genesys 2 Reference Manual

In addition to this, the Genesys 2 FPGA kit also includes high-speed DDR3 memory which is another reason for choosing this board. DDR3 is a type of synchronous dynamic random-access memory (SDRAM) that offers high bandwidth and low power consumption. The onboard DDR3 memory has a capacity of 1GB and can operate at speeds up to 1800 Mbps data rate. The high-speed DDR3 memory is essential for applications that require high bandwidth data processing such as digital signal processing, image and video processing, and machine learning. The onboard DDR3 memory can be accessed and controlled by the onboard FPGA, allowing for efficient and high-speed data transfers between the FPGA and external memory.

Overall, the combination of the VGA connector and high-speed DDR3 memory on the Genesys 2 FPGA kit make it a powerful platform for developing high-performance video and ML processing applications.

5.3 Steps of Integration

Integrating both hardware components on an FPGA kit can be a complex task, but here are the steps that are followed for this purpose:

1. Selecting an FPGA development board that supports high data rate for video processing and fast memory transfer. Also make sure the kit has enough resources to accommodate both designs, and that it supports the necessary communication interfaces between the processor and the accelerator. Genesys 2 is selected for this purpose.

2. Selecting the RISC-V core which will support necessary functions like in this case sending tensor instructions to ztachip, supporting debugging through JTAG, 32 bit integer, multiplication and compressed instructions etc. VexRiscv is selected for this purpose

3. Implementing and testing the RISC-V processor. This can be done using an HDL (hardware description language) such as Verilog or VHDL in Vivado Design Suite. SpinalHDL is used as HDL to generate MyVexRiscv core written in Verilog. For testing purpose, different algorithms in C language are run on the processor. For compiling and debugging purpose different open-source tools are used including RISC-V GNU toolchain, GDB Debugger, and OpenOCD. More about it in later section.

4. Understanding and implementing ztachip accelerator.

5. Integrating the RISC-V processor and ztachip accelerator. This involves connecting the two designs together using the appropriate communication interfaces, such as AXI or Wishbone. AXI4 bus is used for this purpose.

6. Testing and verifying the design. This involves visualizing the results using different tools such as mentioned in step 3 to ensure that the processor and accelerator are communicating correctly.

7. Optimizing the design. Once the design is working, we can optimize it for performance, power consumption, or other metrics by adjusting the design parameters, such as clock frequency, pipeline depth, or resource usage.

.

5.4 Integrated Design Architecture

The design includes a RISC-V processor based on the VexRiscv implementation, DDR memory controller, ztachip accelerator, VGA, camera, and GPIO.

The components are interconnected over various AXI buses such as memory-mapped, streaming, and advanced peripheral buses. The DDR memory controller is connected via a 64-bit AXI memory-mapped bus, while the RISC-V processor has two AXI buses for instruction and data. Ztachip accelerator has two AXI buses, one for receiving tensor instructions from RISC-V processor and the other for initiating memory DMA transfers.

The VGA receives screen data from DDR memory over an AXI streaming bus, while the camera sends captured data to DDR memory over an AXI streaming bus. Both VGA and camera use Xilinx VDMA to manage DDR memory blocks within the AXI crossbar. The GPIO is connected via an AXI advanced peripheral bus. Given below is the diagram of its hardware architecture:

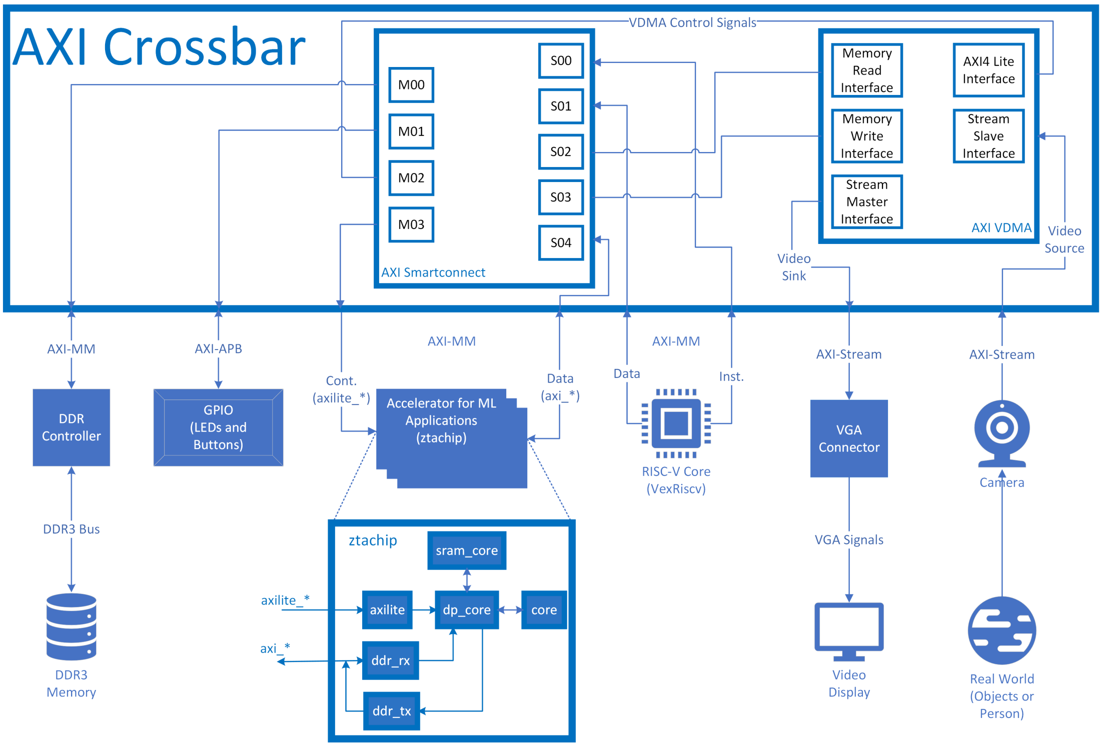


Figure 5.2: Integrated System Block Diagram

Following is the explanation for each of these modules, except for ztachip and MyVexRiscv which are explained in previous sections, used in this integrated hardware design:

DDR3 Memory:

MIG (Memory Interface Generator) 7 Series is a tool provided by Xilinx, which allows the creation of custom memory controllers for various memory interfaces. MIG can be used to generate customized memory controllers for DDR2, DDR3, and DDR4 SDRAM memory interfaces in Xilinx 7 series FPGAs.

MIG provides a wizard-based user interface to guide the user through the process of configuring and generating the memory controller IP. The tool generates a complete memory controller design, including RTL code, constraints files, and testbench files. The generated RTL code includes all the necessary logic to interface with the target memory, handle timing and protocol requirements, and provide a standard interface to the rest of the system.

In this design MIG is used to generate the DDR3 SDRAM memory controller. It is configured to support the MT41J256m16xx-107 memory part and is set to operate at a data rate of 1600 Mbps. The data width of the memory chip is 16 bits, so two chips are used in parallel to provide a total of 32 bits of data width. The input clock frequency is 200 MHz.

The MIG IP core handles the complex memory interface protocol and generates the necessary control signals to interface with the DDR3 memory. It provides several interfaces, including AXI4 memory mapped, AXI4 streaming, and AXI4-Stream Video. In this design, the AXI4 memory mapped interface is used to connect the memory controller to other peripherals.

The AXI4 data width in this design is set to 64 bits, which allows for efficient data transfer between the memory controller and the ztachip accelerator. The MIG IP core also provides advanced features such as read and write data strobes, data mask signals, and support for burst transactions to optimize the data transfer performance.

VGA Connector and Module:

Genesys 2 development board includes a VGA (Video Graphics Array) connector, which is a type of analog video interface commonly used to connect a computer or other video source to a monitor or display.

The VGA connector on the Genesys 2 board consists of three rows of 5 pins each, for a total of 15 pins. These pins are used to transmit analog video signals, as well as to provide connections for ground and other necessary signals. It is compatible with standard VGA cables, which are readily available from a variety of sources. It supports a wide range of video resolutions, including up to 1600x1200 at 60Hz with RGB656 color scheme.

The VGA module in this design is a VHDL implementation of a VGA interface used to display a 24-bit pixel on a VGA monitor. This module receives a 32-bit data stream at the input port. The output ports include the horizontal and vertical synchronization signals, and the red, green, and blue pixel values.

The VGA interface is structured to display frames of resolution 640x480 with a refresh rate of 60 Hz. The input data stream is 32 bits wide, and the output pixel values are 24 bits wide, with each color component represented by 8 bits. This resolution is chosen because of the camera selection for this design.

Camera:

The camera used to capture video and images is OV7670. The OV7670 is a low-cost, small form-factor camera module designed for use in a wide range of embedded applications, such as mobile phones, toys, security systems, and robotics. It is manufactured by OmniVision Technologies, a leading supplier of image sensors and processing technologies.

The OV7670 features a 1/6-inch optical format CMOS (Complementary Metal-Oxide-Semiconductor) image sensor with a resolution of 640 x 480 pixels (VGA) and 10-bit RGB (Red Green Blue) output. The camera module also includes an on-chip analog-to-digital converter (ADC), a programmable gain control (AGC), and a built-in image processing pipeline with color interpolation, gamma correction, and edge enhancement.

The OV7670 communicates with a host microcontroller using a parallel interface, and it supports multiple data formats, including YUV, RGB, and JPEG. The camera module can operate at a frame rate of up to 30 frames per second (fps) at VGA resolution, and it requires a single 3.3V power supply.

The camera module used in this design has several output ports, including SIOC, SIOD, RESET, PWDN, XCLK, tdata_out, and tvalid_out etc. These ports are used to communicate with the external environment and transmit the captured image data.

The module uses a state machine to capture the image data from the OV7670 sensor. The state machine waits for the vertical sync (VS) signal from the sensor and then starts capturing image data on the rising edge of the pixel clock (PCLK) signal. The captured data is stored in a shift register and then transferred to an output register. The transfer is triggered by the ready signal from the external environment. Once the transfer is complete, the state machine waits for the next vertical sync signal to start capturing the next frame.

GPIO:

For the purpose of general purpose input output leds and buttons of FPGA board are used. The leds have various functions which includes to show whether the overall system is working well or not. A test can be conducted for ztachip in which blinking of leds show that the hardware is processing the data and generating the output correctly. The buttons are used for choosing between different ML applications running. Pressing the button will run and show the next application on to the display. Both leds and buttons are connected to the other components via AXI lite bus protocol.

AXI Crossbar:

AXI crossbar is a component in the Vivado Design Suite that allows for the interconnection of multiple AXI (Advanced eXtensible Interface) masters and slaves within a single design. It provides a centralized point of control for the communication between multiple AXI-based IP cores, simplifying the interconnection and management of these components.

The AXI crossbar component operates by arbitrating access to shared resources between multiple AXI masters, ensuring that data transfers are managed efficiently and without contention. It also supports the routing of data between AXI-based IP cores through a range of different transfer protocols, including memory-mapped, streaming, and peripheral AXI interfaces.

In this design AXI crossbar is responsible for connecting all the components which includes the following:

- DDR3 memory controller connected via AXI memory mapped interface
- General purpose input outputs (GPIOs) connected via AXI advanced peripheral bus protocol
- ztachip connected via AXI memory mapped interface
- MyVexRiscv processor connected via AXI memory mapped interface
- VGA module connected via AXI stream bus
- Camera module connected via AXI stream bus

Within the AXI crossbar two IPs (intellectual properties) are used which are AXI smartconnect and AXI VDMA.

AXI Smartconnect:

AXI SmartConnect is an IP (intellectual property) block that can be used to simplify the interconnection of different AXI (Advanced eXtensible Interface) peripherals. AXI is a widely-used interconnect protocol for connecting multiple IP blocks in a system-on-chip (SoC) design. It simplifies the process of connecting multiple AXI peripherals by automatically generating the necessary interconnect logic. It supports a wide range of AXI interfaces, including AXI4, AXI4-Lite, AXI4-Stream, and AXI4-Stream Data FIFO.

The AXI SmartConnect IP block can be configured with a graphical user interface (GUI) to specify the number and type of AXI interfaces to connect, as well as the arbitration scheme for resolving conflicts between multiple AXI masters. It also provides optional address decoding logic to enable routing of transactions to the appropriate peripheral. This address decoding can be configured with the GUI or through a set of AXI registers that are accessible through a JTAG interface. It simplifies the process of interconnecting multiple AXI peripherals in a Vivado design, reducing design time and minimizing errors.

The AXI SmartConnect IP is responsible for ensuring that the data transfer between the masters and slaves is done smoothly and efficiently. It handles the arbitration and routing of data between the different AXI interfaces. The AXI SmartConnect IP also provides the ability to configure the system to meet specific requirements, such as changing the priority of different masters or adding new slaves to the system. In this design, AXI SmartConnect IP is used to connect 4 AXI masters and 5 AXI slaves.

The 4 AXI masters are:

1. DDR3 memory - This is the main memory of the system, which is

used to store the data and instructions required by the system.

2. GPIO - This is used for general-purpose input/output operations. It provides the ability to interface with external devices such as LEDs and buttons.

3. Control signals for ztachip - This is used to control the various functions of the ztachip.

4. AXI VDMA control signals - This is used to control the AXI VDMA, which is used for video processing operations such as video capturing and display.

The 5 AXI slaves are:

1. Instruction signals for RISC-V processor - This is used to provide instructions to MyVexRiscv processor, which is the main processor of the system.

2. Control signals for RISC-V processor - This is used to control the various functions of the RISC-V processor.

3. AXI VDMA memory read interface - This is used by the AXI VDMA to read data from the memory.

4. AXI VDMA memory write interface - This is used by the AXI VDMA to write data to the memory.

5. ztachip data signals - This is used to transfer data to and from the ztachip.

AXI VDMA:

AXI VDMA (Video Direct Memory Access) IP is an Intellectual Property (IP) core that is available in Vivado Design Suite, and it is used to enable high-speed data transfer between video devices and memory. This IP core provides efficient direct memory access (DMA) data transfers between AXI4 memory-mapped interface and a video stream interface.

The AXI VDMA IP can support multiple independent video channels with different resolutions and refresh rates. It contains a programmable Genlock feature that enables synchronization between multiple video streams. The IP core also includes advanced features such as frame buffer read and write arbitration, programmable video timing generation, and burst transfers to enhance system performance. It has several configurable parameters, including the number of channels, the width and height of the video stream, the memory interface data width, and the burst transfer length. It can operate in two modes: Read-Only and Write-Only. In Read-Only mode, the IP core transfers data from the memory to the video stream, while in Write-Only mode, the IP core transfers data from the video stream to the memory.

The AXI VDMA IP core is commonly used in various video processing applications such as video surveillance, machine vision, and augmented reality. By utilizing this IP core, designers can easily implement high-performance video processing systems with efficient memory access and data transfer capabilities. In this design it is used for similar purpose. The camera and VGA are connected to the video source and sink of VDMA interfaces respectively. The memory read and write interfaces are also connected to DDR3 memory via AXI smartconnect. This IP will ensure that the video processing will run smoothly.

5.5 Process Flow

The overall process flow includes the following four steps or processes happening simultaneously:

1. The Camera sends the signals to AXI VDMA Video Source interface. The VDMA transfers to to the DDR3 memory for storage through its memory write interface via AXI Smartconnect.

2. The RISC-V processor MyVexRiscv reads the tensor instructions from the DDR3 memory and sends them to ztachip control interface all via AXI Smartconnect

3. Then after ztachip reads those tensor instructions it initiates DMA (direct memory access) and reads the video data (sent by the camera) from DDR3 memory and do required computations (which are previously mentioned in ztachip section) and then stores the processed data back to DDR3 memory and again all of this via AXI Smartconnect.

4. After this the VDMA reads the processed data from DDR3 memory through its memory read interface via AXI Smartconnect and sends to the VGA module through its video sink interface.

5.6 Hardware Setup

The picture below show the overall hardware setup for this project, including the FPGA board and the camera module. The FPGA board used in the project is the Digilent Genesys 2, which features a Xilinx Kintex-7 FPGA.

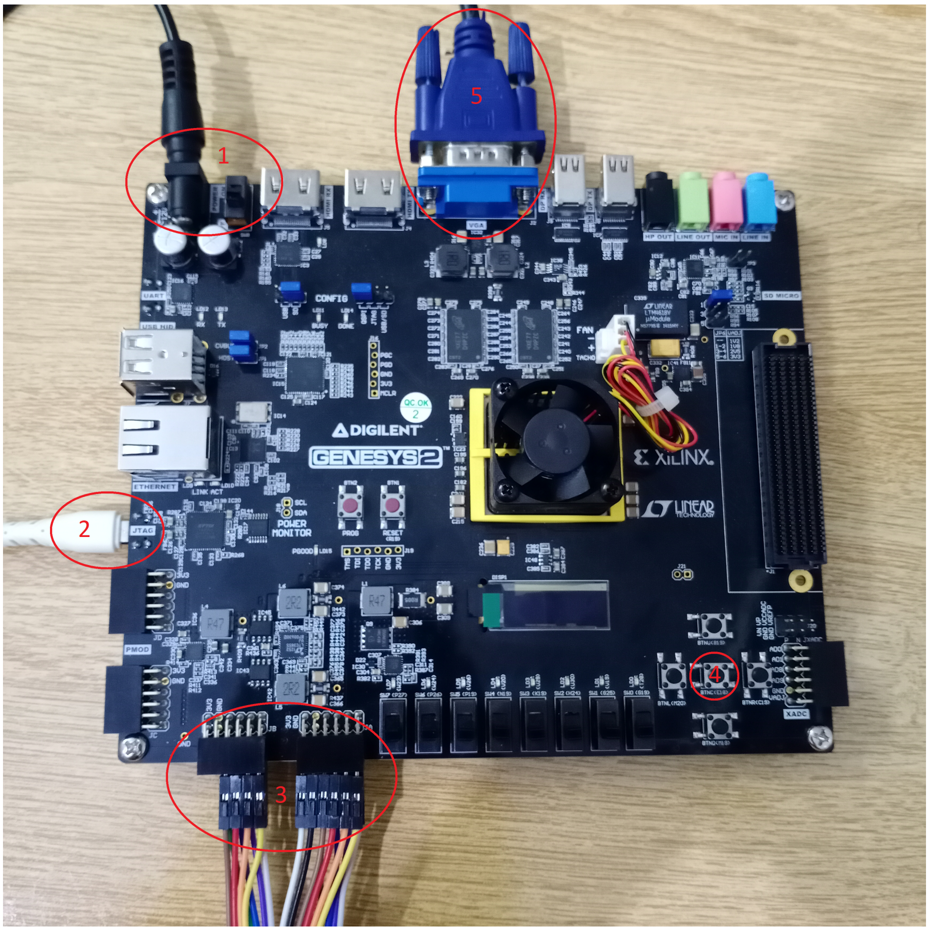


Figure 5.3: Hardware System

The above picture illustrate the overall system architecture and the physical layout of the hardware components. Following is the description of each of the labelled component:

1. This is a Power jack with voltage of 12 V DC. The Genesys 2 board can receive power from an external power supply through the center-positive barrel jack (J27). The external supply voltage must be 12 V ±5 %. The Genesys 2 board cannot be powered from the USB bus. This external power helps in running hardcore circuits on this

development board.

2. The JTAG port allows the FPGA board to be connected with Xilinx Vivado and other tools. In case of Xilinx Vivado, the Digilent USB-JTAG circuitry can be utilized to program the FPGA device. However, the programming method used in this case is different from the Simple JTAG programming method. A memory configuration device is used for indirect programming which is a two-step process controlled by Vivado. Firstly, a design that can program flash devices is programmed into the FPGA, and then the data is transferred to the flash device through the FPGA circuit. The flash device can configure the FPGA automatically during a subsequent power-on or reset event as per the mode jumper setting. The Spansion's s25fl256xxxxxx0 part is used for programming files storage in the flash device, which will remain until they are overwritten, irrespective of power-cycle events.

3. This is the camera module connectivity. The camera is connected to the Pmod JA and JB connectors via male to female wire bundle. Total wires used are 18 in number including 2 for power supply and ground, 8 data wires and 8 control wires. The Pmod connectors provide fast connectivity from FPGA to the camera with 3.3 V supply.

4. This is the button BTNC (button center) which is when pressed allows the next ML algorithm to run.

5. This is the VGA connector which is responsible for sending video signals to a display screen. This VGA cable can be connected to any LCD or monitor which has VGA port and supports at least 640 x 480 display resolution.

Chapter 6

Conclusion

6.1 Summary and Implications

This project aimed to implement an integrated hardware system containing open source RISC-V processor and a hardware accelerator for machine learning applications. Other features of this project includes the use of camera and displaying the results on to the screen with the help of video processing IPs in Vivado Design Suite.

During the course of this project we first of all started with a simple RISC-V processor named zeroriscy and it's reverse engineering helped us build a strong understanding of a RISC-V processor architecture. Even before that understanding of a RISC-V ISA including its assembly language, set of instructions, machine code conversion, memory allocation, and many other parameters and important points to remember about designing a RISC-V processor. We may not be able to build our own processor but doing an in-depth back analysis of a sophisticated architecture which was designed using an unfamililar ISA and then doing its implementation as well as testing using open-source tools on FPGA is also not at all simple and effortless.

That rigorous and intensive understanding of zeroriscy made it easier to further understand other RISC-V cores. Then replacing zeroriscy with VexRiscv is lead by a reason that VexRiscv supports connectivity of JTAG interface which makes it easier for debugging. Then for the second part of the project open-source ztachip accelerator was used. It used because of its open-source nature and ability to support a wide range of ML applications.

Next step was to utilize ztachip accelerator by the help of RISC-V processor. Their integration and the displaying of the results required many other modules to be integrated with the system as well. The modules include camera, VGA, and DDR memory. The video processing required use of AXI VDMA IP and for the fast interconnection of all these components over AXI bus required the use of AXI Smartconnect IP.

In conclusion, this project was successful in designing and implementing a RISC-V ML accelerating system on an FPGA board. The system was also able to process video data from the camera in real-time with the help of AXI VDMA, ztachip, and DDR memory. This project serves as a proof of concept for the use of FPGAs in running ML applications efficiently with low memory usage and can be extended in future work to include additional functionalities and optimizations.

6.2 Abilities

Ztachip's domain-specific language (DSL) has undergone extensive testing across various vision preprocessing and AI tasks. [20] These tasks include TensorFlow model-based tasks such as the following:

- Image Classification
- Object Detection
- Edge Detection using Canny Algorithm
- Color Space Conversion
- Contrast Enhancement
- Image Blurring using Gaussian Convolution
- Harris Corner Detection for Robotics SLAM
- Optical Flow for Motion Detection
- Image Resizing.

But for this demo only Object Detection, Edge Detection, Motion Detection, and Harris Corner are run on the hardware.

6.3 Result

This section demostrates the results shown by this ML accelerating system. The values displayed next to the identified objects or persons represent the probability or likelihood of the algorithm's detection accuracy for each of the identified entities.

The below figure shows that the Object Detection algorithm is succussfully detecting a person with a confidence score of 0.63 and keyboard with probability 0.58 at the same time.

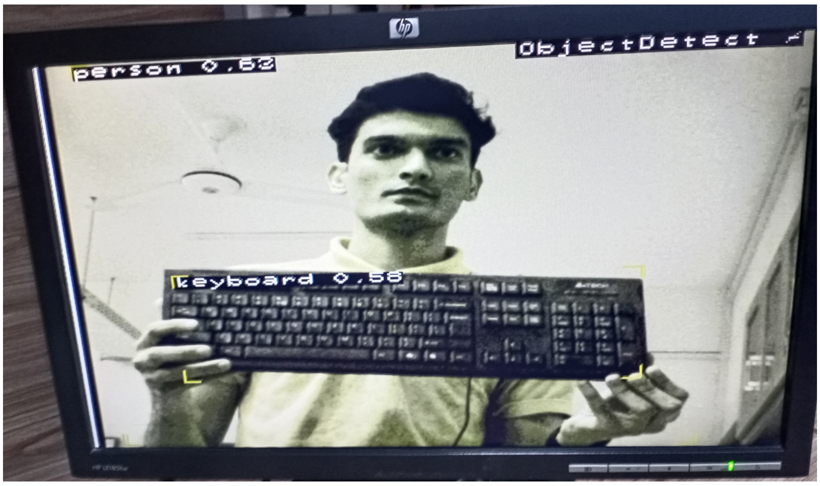


Figure 6.1: Object Detection of Person and Keyboard

The below figure shows that the Object Detection algorithm is succussfully detecting an Air Conditioner Remote with a confidence score of 0.65.

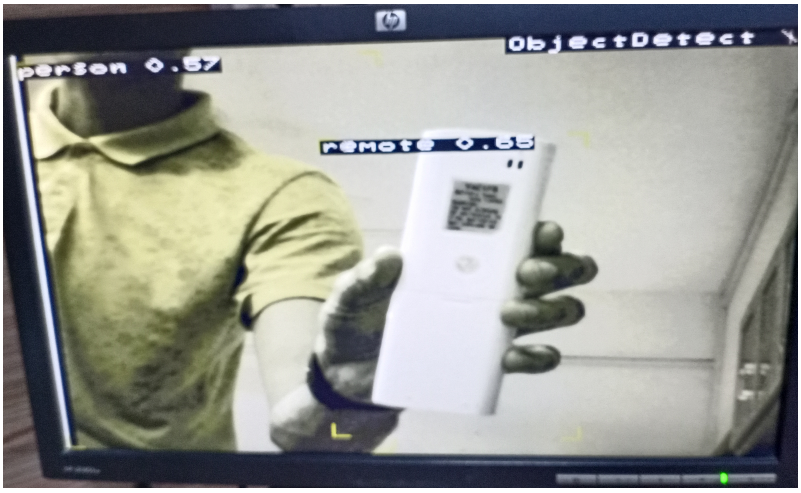


Figure 6.2: Object Detection of A.C. Remote

The below figure shows that the Object Detection algorithm is succuss-fully detecting an Umbrella with a confidence score of 0.75.

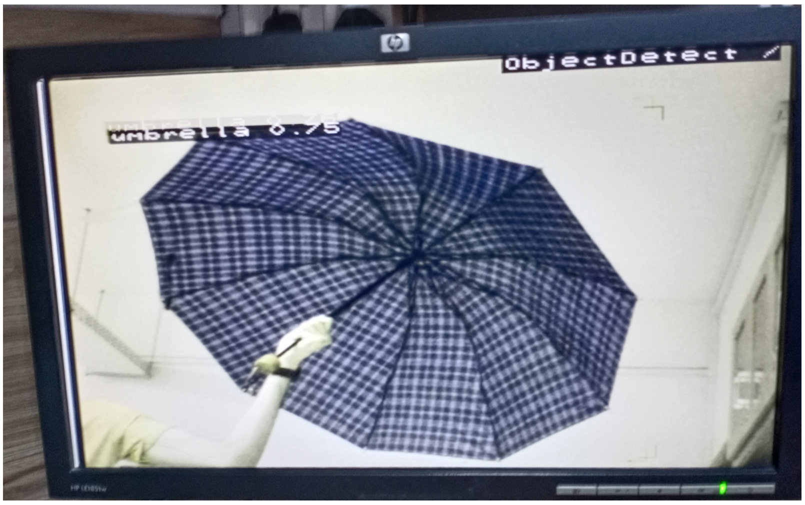


Figure 6.3: Object Detection of Umbrella

The below figure shows the Edge Detection algorithm succussfully detecting the edges of a Person and a Keyboard at the same time.

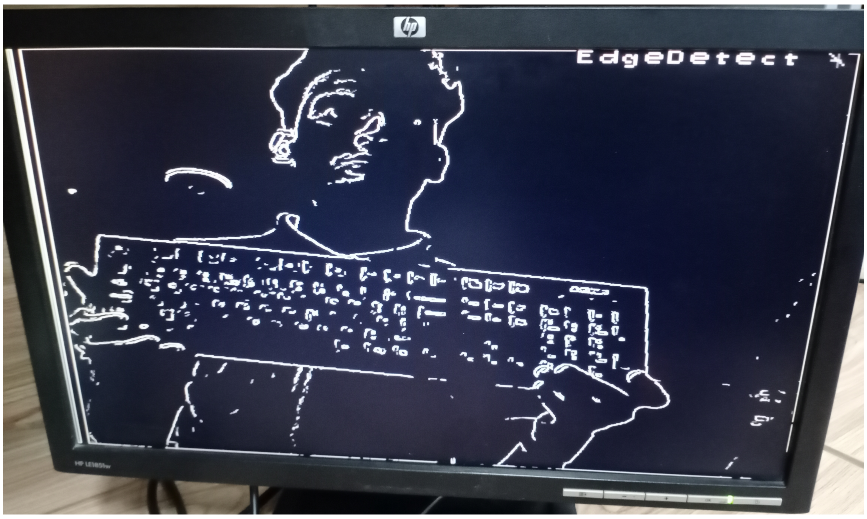


Figure 6.4: Edge Detection of Person and Keyboard

The below figure shows the Point of Interest using Harris Corner algorithm succussfully detecting points of interest on a Person and a Keyboard at the same time and highlighting them with yellowish dots.

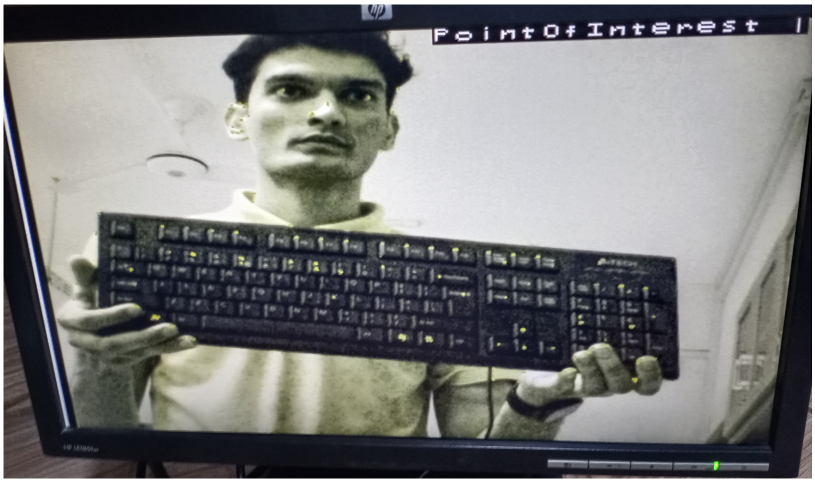


Figure 6.5: Point of Interest on Person and Keyboard

The below figure shows the Motion Detection algorithm succussfully detecting a Keyboard and highlighting it with vivid bluish colors.

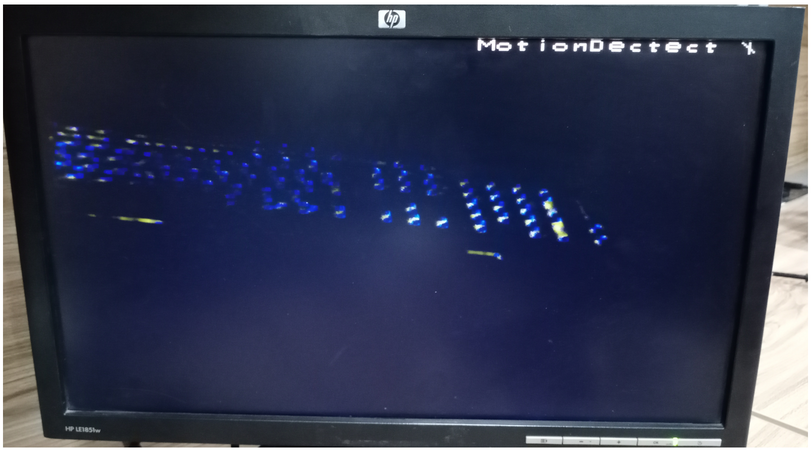


Figure 6.6: Motion Detection on Keyboard

Finally this last below figure shows the all these four algorithms running in parallel as a multitask to show the efficiency and performance of this ML accelerator.

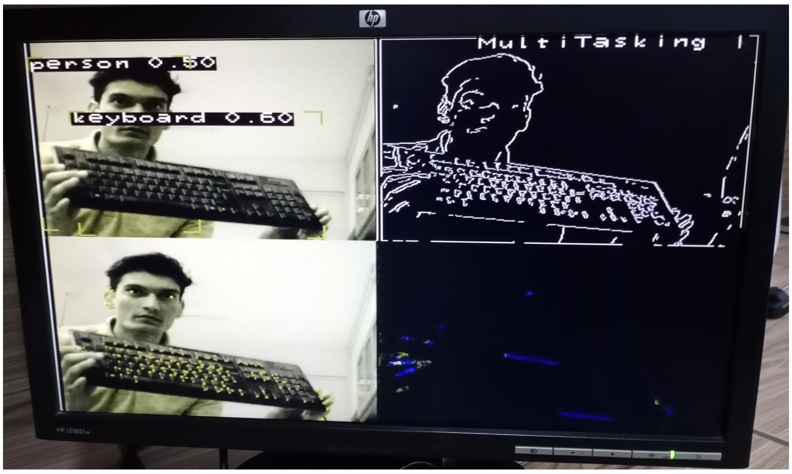


Figure 6.7: All Four ML Algorithms running in Parallel

6.4 Performance

It's important to note that performance and resource utilization can vary greatly depending on the specific application and its requirements. However, it's encouraging to see that ztachip is able to achieve competitive performance while utilizing less computing resources than other platforms, resulting in potentially lower power consumption. Efficient use of memory is also an important factor to consider in DSA design, as it can greatly impact overall system performance and cost.

The performance of ztachip is impressive and is tested for benchmarking on the popular Mobinet-SSD AI model. MobileNet SSD is an artificial intelligence (AI) model for object detection. It is a single-shot detection (SSD) framework that uses a MobileNet architecture as a feature extractor. The model is designed to run on mobile and embedded devices with limited computational resources while achieving high accuracy in object detection tasks. [20]

The MobileNet architecture is optimized for mobile devices and is made up of depthwise separable convolutions that reduce the number of parameters and operations required. This allows the model to achieve high accuracy in object detection while maintaining fast inference speeds on devices with limited resources. The SSD framework used in MobileNet SSD divides the input image into a grid of cells and predicts the presence and location of objects within each cell. The predictions are made at multiple scales, allowing the model to detect objects of different sizes. Following is the table showing benchmark score comparison with Nvidia Accelerator:

Hardware Accelerators:	FPS (Frames per Second)	GOPS (Giga Operations per Second)
Ztachip ML Accelerator	10	20
Nvidia Jetson Nano	40	500

Table 6.1: Benchmark Score Comparison of ztachip with Nvidia

Source: GitHub

Resultantly, ztachip's computing resource utilization is six times better than that of Nvidia, which leads to a considerable reduction in power consumption. This makes ztachip an ideal solution for power-sensitive applications, such as edge devices, where low power consumption is critical.

Furthermore, ztachip has much lower memory requirements compared to other solutions, due to its efficient use of memory. This means that ztachip can be deployed in devices with limited memory, making it a cost-effective solution for a wide range of applications.

References

[1] SpinalHDL/VexRiscv. GitHub, 2021, https://github.com/SpinalHDL/VexRiscv

[2] Smart2Zero. (2022, January 5). NASA selects RISC-V CPU for next-gen spaceflight processor. [Online article]. Retrieved from https://www.smart2zero.com/en/nasa-selects-risc-v-cpu-for-next-gen-spaceflight-processor/

[3] SiFive. (n.d.). About us. Retrieved from https://www.sifive.com/about

[4] Alibaba.com. (n.d.). DC-ROMA RISC-V Development Laptop. [Product listing]. Retrieved February 23, 2023, from https://www.alibaba.com/product-detail/DC-ROMA-RISC-V-Development-Laptop_1600610157163.html

[5] The Register. (2022, September 23). Google reportedly using SiFive RISC-V cores in Tensor Processing Units. [Online article]. Retrieved from https://www.theregister.com/2022/09/23/google_using_sifive_riscv_cores/

[6] The Register. (2022, October 5). China switches from ARM to RISC-V for top-level data center chip. [Online article]. Retrieved from https://www.theregister.com/2022/10/05/china_riscvv_arm_datacenter/

[7] Islamabad Post. (2022, October 20). Ntiny-e: NUST realizes Pakistan's first truly indigenous embedded microprocessor. [Online article]. Retrieved from https://islamabadpost.com.pk/ntiny-e-nust-realizes-pakistans-first-truly-indigenous-embedded-microprocessor/

[8] Asanovic, Krste, et al. "The rocket chip generator." EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17 4 (2016).

[9] Zhao, Jerry, et al. "Sonicboom: The 3rd generation berkeley out-of-order machine." Fourth Workshop on Computer Architecture Research with RISC-V. Vol. 5. 2020.

[10] Y. Lee et al., "A 45nm 1.3GHz 16.7 double-precision GFLOPS/W RISC-V processor with vector accelerators," ESSCIRC 2014 - 40th European Solid State Circuits Conference (ESSCIRC), 2014, pp. 199-202, doi: 10.1109/ESSCIRC.2014.6942056.

[11] Keller, Ben, et al. "A RISC-V processor SoC with integrated power management at submicrosecond timescales in 28 nm FD-SOI." IEEE Journal of Solid-State Circuits 52.7 (2017): 1863-1875.

[12] D. A. Santos, L. M. Luza, C. A. Zeferino, L. Dilillo and D. R. Melo, "A Low-Cost Fault-Tolerant RISC-V Processor for Space Systems," 2020 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS), Marrakech, Morocco, 2020, pp. 1-5, doi: 10.1109/DTIS48698.2020.9081185.

[13] Fritzmann, Tim, Georg Sigl, and Johanna Sepúlveda. "RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography." IACR Transactions on Cryptographic Hardware and Embedded Systems (2020): 239-280.

[14] Steinegger, Stefan, and Robert Primas. "A fast and compact RISC-V accelerator for ascon and friends." Smart Card Research and Advanced Applications: 19th International Conference, CARDIS 2020, Virtual Event, November 18–19, 2020, Revised Selected Papers. Cham: Springer International Publishing, 2021.

[15] PULP Platform. (n.d.). Project Info. [Webpage]. Retrieved from `https://pulp-platform.org/projectinfo.html`

[16] lowRISC. (n.d.). ibex. [GitHub Repository]. Retrieved from `https://github.com/lowRISC/ibex`

[17] P. Davide Schiavone et al., "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2017, pp. 1-8, doi: 10.1109/PATMOS.2017.8106976.

[18] Schiavone, P.D. (2018). zero-riscy: User Manual, Revision 0.2, January 2018. Retrieved from `https://www.pulp-platform.org/docs/user_manual.pdf`

[19] ztachip. (n.d.). ztachip. [GitHub Repository]. Retrieved from `https://github.com/ztachip/ztachip`

[20] ztachip. (n.d.). ztachip overview. [GitHub markdown file]. Retrieved from `https://github.com/ztachip/ztachip/blob/master/Documentation/Overview.md`

[21] Tom01h. (n.d.). zero-riscy [GitHub repository]. Retrieved from `https://github.com/tom01h/zero-riscy`

[22] Barbier, D. (2018). RISC-V: Open-Source Silicon [PDF]. SiFive. Retrieved from `https://riscv.org/wp-content/uploads/2018/07/DAC-SiFive-Drew-Barbier.pdf`

[23] Ztachip. "Ztachip Documentation - Hardware Design." GitHub, 2021, `https://github.com/ztachip/ztachip/blob/master/Documentation/HardwareDesign.md`

[24] Digilent. (n.d.). Genesys 2 Reference Manual. Retrieved from `https://digilent.com/reference/programmable-logic/genesys-2/reference-manual`

FPGA

ORIGINALITY REPORT

8%	7%	4%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	link.springer.com Internet Source	<1%
2	www.farnell.com Internet Source	<1%
3	Submitted to University of Southern Mississippi Student Paper	<1%
4	people.eecs.berkeley.edu Internet Source	<1%
5	elib.dlr.de Internet Source	<1%
6	github.com Internet Source	<1%
7	tches.iacr.org Internet Source	<1%
8	www.researchgate.net Internet Source	<1%
9	en.wikipedia.org Internet Source	<1%
10	Submitted to University of Sussex Student Paper	<1%

11	aaltodoc.aalto.fi Internet Source	<1 %
12	libraetd.lib.virginia.edu Internet Source	<1 %
13	www.eeworldonline.com Internet Source	<1 %
14	2015.aeroconf.org Internet Source	<1 %
15	Sarah L. Harris, David Harris. "Microarchitecture", Elsevier BV, 2022 Publication	<1 %
16	webthesis.biblio.polito.it Internet Source	<1 %
17	Pranav S. Mutha, Yogita M. Vaidya. "FPGA reconfiguration using UART and SPI flash", 2017 International Conference on Trends in Electronics and Informatics (ICEI), 2017 Publication	<1 %
18	boston.i3investor.com Internet Source	<1 %
19	pdfs.semanticscholar.org Internet Source	<1 %
20	csrc.nist.gov Internet Source	<1 %
21	docs.askives.com Internet Source	<1 %

22	hal-lirmm.ccsd.cnrs.fr Internet Source	<1 %
23	readthedocs.org Internet Source	<1 %
24	"Smart Card Research and Advanced Applications", Springer Science and Business Media LLC, 2021 Publication	<1 %
25	www.digitalcamerawarehouse.com.au Internet Source	<1 %
26	Andrew Elbert Wilson, Nathan Baker, Ethan Campbell, Jackson Sahleen, Michael Wirthlin. "Post-Radiation Fault Analysis of a High Reliability FPGA Linux SoC", Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2023 Publication	<1 %
27	Sallar Ahmadi-Pour, Vladimir Herdt, Rolf Drechsler. "MircoRV32", Proceedings of the Workshop on Design Automation for CPS and IoT, 2021 Publication	<1 %
28	www2.eecs.berkeley.edu Internet Source	<1 %
29	www.arxiv-vanity.com Internet Source	<1 %

30	Benjamin W. Mezger, Douglas A. Santos, Luigi Dilillo, Cesar A. Zeferino, Douglas R. Melo. "A survey of the RISC-V architecture software support", IEEE Access, 2022 Publication	<1 %
31	repository.tudelft.nl Internet Source	<1 %
32	Submitted to University of Warwick Student Paper	<1 %
33	www.semanticscholar.org Internet Source	<1 %
34	Ben Keller, Martin Cochet, Brian Zimmer, Jaehwa Kwak, Alberto Puggelli, Yunsup Lee, Milovan Blagojevic, Stevo Bailey, Pi-Feng Chiu, Palmer Dabbelt, Colin Schmidt, Elad Alon, Krste Asanovic, Borivoje Nikolic. "A RISC-V Processor SoC With Integrated Power Management at Submicrosecond Timescales in 28 nm FD-SOI", IEEE Journal of Solid-State Circuits, 2017 Publication	<1 %
35	Submitted to Sikkim Manipal University Student Paper	<1 %
36	Submitted to University of Rome Tor Vergata Student Paper	<1 %
37	www.coursehero.com Internet Source	<1 %

38	www.supplychain247.com Internet Source	<1 %
39	Submitted to Higher Education Commission Pakistan Student Paper	<1 %
40	www.dsprelated.com Internet Source	<1 %
41	Douglas Almeida Santos, Lucas Matana Luza, Cesar Albenes Zeferino, Luigi Dilillo, Douglas Rossi Melo. "A Low-Cost Fault-Tolerant RISC-V Processor for Space Systems", 2020 15th Design & Technology of Integrated Systems in Nanoscale Era (DTIS), 2020 Publication	<1 %
42	Matthew Naylor, Simon W. Moore, Alan Mujumdar. "A consistency checker for memory subsystem traces", 2016 Formal Methods in Computer-Aided Design (FMCAD), 2016 Publication	<1 %
43	citeseerx.ist.psu.edu Internet Source	<1 %
44	cs.uwaterloo.ca Internet Source	<1 %
45	cv32e40p.readthedocs.io Internet Source	<1 %
	www.esri.com	

46	Internet Source	<1%
47	Görkem Nişancı, Paul G. Flikkema, Tolga Yalçın. "Symmetric Cryptography on RISC-V: Performance Evaluation of Standardized Algorithms", Cryptography, 2022 Publication	<1%
48	www.mobilitytechzone.com Internet Source	<1%
49	Atul Prasad Deb Nath, Kshitij Raj, Swarup Bhunia, Sandip Ray. "SOCCOM: Automated Synthesis of System-on-Chip Architectures", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2022 Publication	<1%
50	F. Renzini, D. Rossi, E. Franchi Scarselli, C. Mucci, R. Canegallo. "A Fully Programmable eFPGA-Augmented SoC for Smart-Power Applications", 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2018 Publication	<1%
51	Zimmer, Brian, Yunsup Lee, Alberto Puggelli, Jaehwa Kwak, Ruzica Jevtic, Ben Keller, Stevo Bailey, Milovan Blagojevic, Pi-Feng Chiu, Hanh-Phuc Le, Po-Hung Chen, Nicholas Sutardja, Rimas Avizienis, Andrew Waterman, Brian Richards, Philippe Flatresse, Elad Alon, Krste Asanovic, and	<1%

	Borivoje Nikolic. "A RISC-V vector processor with tightly-integrated switched-capacitor DC-DC converters in 28nm FDSOI", 2015 Symposium on VLSI Circuits (VLSI Circuits), 2015. Publication	
52	cis.cihe.edu.hk Internet Source	<1 %
53	homes.cs.washington.edu Internet Source	<1 %
54	vtechworks.lib.vt.edu Internet Source	<1 %
55	web.archive.org Internet Source	<1 %
56	www.filibeto.org Internet Source	<1 %
57	"Cognitive Radio Oriented Wireless Networks", Springer Science and Business Media LLC, 2018 Publication	<1 %
58	D.D. Gajski. "A retargetable, ultra-fast instruction set simulator", Design Automation and Test in Europe Conference and Exhibition 1999 Proceedings (Cat No PR00078) DATE-99, 1999 Publication	<1 %
59	Parhami, Behrooz. "Computer Architecture", Oxford University Press Publication	<1 %

60	Roland Holler, Dominic Haselberger, Dominik Ballek, Peter Rossler, Markus Krapfenbauer, Martin Linauer. "Open-Source RISC-V Processor IP Cores for FPGAs — Overview and Evaluation", 2019 8th Mediterranean Conference on Embedded Computing (MECO), 2019 Publication	<1 %
61	cryptography.gmu.edu Internet Source	<1 %
62	docplayer.net Internet Source	<1 %
63	islamabadpost.com.pk Internet Source	<1 %
64	kth.diva-portal.org Internet Source	<1 %
65	nebula.wsimg.com Internet Source	<1 %
66	oops.uni-oldenburg.de Internet Source	<1 %
67	riscv.org Internet Source	<1 %
68	www.springerprofessional.de Internet Source	<1 %
69	www.st.com Internet Source	<1 %

70	Corrado De Sio, Sarah Azimi, Andrea Portaluri, Luca Sterpone. "SEU Evaluation of Hardened-by-Replication Software in RISC- V Soft Processor", 2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2021 Publication	<1 %
71	Rogerio Paludo, Leonel Sousa. "NTT Architecture for a Linux-Ready RISC-V Fully-Homomorphic Encryption Accelerator", IEEE Transactions on Circuits and Systems I: Regular Papers, 2022 Publication	<1 %
72	Vaibhav Verma, Tommy Tracy II, Mircea R. Stan. "EXTREM-EDGE—EXtensions To RISC-V for Energy-efficient ML inference at the EDGE of IoT", Sustainable Computing: Informatics and Systems, 2022 Publication	<1 %
73	"The Fourth Terminal", Springer Science and Business Media LLC, 2020 Publication	<1 %
74	Maha S. Diab, Esther Rodriguez-Villegas. "Embedded Machine Learning Using Microcontrollers in Wearable and Ambulatory Systems for Health and Care Applications: A Review", IEEE Access, 2022 Publication	<1 %

75	P. Nannipieri, S. Di Matteo, L. Zulberti, F. Albicocchi, S. Saponara, L. Fanucci. "A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to speed-up CRYSTALS Algorithms", IEEE Access, 2021 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		