Project Title

INTELLERA,
A Hardware based Accelerated Matrix MAC Processor.

Dated: Sep 30, 2023

Supervisor: Dr. Fahad Bin Muslim

Co-Supervisor: Dr. Taj Muhammad

Group Members: Saad Khan – Group Lead

Syed Zaeem Shakir

Mahnoor Maleeka

Revision History:

Revision History	*Date*	*Comments*
1.00		
2.00		

Document Approval:

The following document has been accepted and approved by the following:

Signature	***Date***	***Name***

List of Contents

List of Figures

List of Tables

1 INTRODUCTION

1.1. PURPOSE

In the field of machine learning, matrix multiplication is a fundamental operation that plays a crucial role in various algorithms and computations. However, traditional processors often struggle to efficiently execute matrix multiplication tasks due to their general-purpose nature. This inefficiency results in increased time consumption and hampers the overall performance of machine learning applications. Therefore, there is a pressing need for a specialized processor design that incorporates matrix multiplication as a core instruction and aims to decrease time consumption in machine learning.

1.2. PRODUCT SCOPE

Intellera is a groundbreaking project aimed at designing a RISC-V based processor with a customized instruction set architecture (ISA) that includes matrix multiplication instructions. The goal of Intellera is to address the performance bottleneck caused by conventional processors when executing matrix operations in machine learning algorithms. By enhancing the processor's capabilities and providing dedicated hardware acceleration for matrix Multiply-Accumulate (MAC) operations, Intellera aims to significantly reduce the time consumed in machine learning tasks.

Intellera proposes a novel approach to address the time consumption challenge in machine learning. The project focuses on designing a RISC-V based processor that features an extended instruction set architecture specifically tailored for matrix multiplication. By integrating dedicated hardware support for matrix MAC operations, Intellera can harness the parallel(pipelining) processing capabilities of the processor to accelerate matrix computations and reduce the overall execution time.

The project utilizes the knowledge of computer hardware and Risc-V Instruction Set for creating a customized instruction set and concept of systolic arrays implementation for accelerating the multiplication of matrices.

Table 1: Terms used in this document and their description.

Name	Description
RISC	Reduced Instruction Set Computer
FPGA	Field Programable Gate Array
ISA	Instruction Set Architecture
MAC	Multiplication Accumulation
ALU	Arithmetic Logic Unit

2 OVERVIEW

2.1 THE OVERALL DESCRIPTION

The project encompasses the development of a versatile RISC-V processor with the primary objective of accelerating matrix multiplication operations, a pivotal task within the domain of machine learning and scientific computing. The project's evolution began with the implementation of a single-cycle RISC-V processor using Vivado, establishing a solid foundation for subsequent enhancements. The immediate plan involves transitioning this processor into a highly efficient, pipelined architecture with five stages. This pipeline design aims to mitigate hazards and optimize the execution of RISC-V instructions, delivering improved throughput and performance. Beyond the processor core, a key innovation lies in the integration of a dedicated systolic array hardware architecture, meticulously designed to expedite matrix multiplication. This addition introduces a specialized matrix computation engine, effectively harnessing parallelism to dramatically reduce execution times for large-scale matrix operations.

2.2 PRODUCT PERSPECTIVE

In the realm of modern computing, the demand for high-performance processors tailored to specific computational tasks has intensified. The customized RISC-V processor represents an innovative response to this demand, offering a scalable and efficient solution for accelerating matrix multiplication operations. It stands at the intersection of hardware and software, where traditional RISC-V architecture is enhanced with a carefully crafted module inspired by systolic array for matrix operations. This project is a testament to the adaptability and extensibility of the RISC-V ISA, providing a dedicated solution for a critical computation within machine learning workflows. It complements existing processors by offering superior matrix computation capabilities, making it an invaluable tool for a wide range of applications, including deep learning, data analytics, and scientific simulations.

2.2.1 PRODUCT FUNCTIONS

The customized RISC-V processor serves several vital functions, addressing the specific computational requirements of matrix multiplication within the context of machine learning and scientific computing. Its primary functions include:

1. Pipelined RISC-V Execution: The processor transitions from a single-cycle design to a highly efficient five-stage pipeline, reducing instruction latency and increasing overall throughput. This enhancement optimizes the execution of a broad spectrum of RISC-V instructions.

2. Hazard Mitigation: The pipeline design incorporates hazard detection and resolution mechanisms to handle data hazards, control hazards, and structural hazards, ensuring smooth instruction execution and maintaining data consistency.

3. Matrix Multiplication Acceleration: The project introduces a dedicated systolic array type of hardware architecture, capable of rapidly performing matrix multiplication operations. This function vastly improves the efficiency of matrix computations, significantly reducing execution times for machine learning algorithms.

4. FPGA Implementation: The processor, along with the systolic array, is designed for deployment on FPGA platforms, with a preference for the Nexys ddr 4 FPGA board. FPGA implementation allows for hardware-level validation, real-time performance evaluation, and practical usability in embedded systems and edge computing scenarios.

5. Performance Evaluation: Extensive benchmarking and performance testing are conducted, comparing the customized RISC-V processor's performance in matrix multiplication tasks with that of standard processors. This function serves to quantify the speedup achieved by the specialized matrix computation engine.

By combining these functions, the customized RISC-V processor addresses the pressing need for efficient matrix multiplication within machine learning and scientific computing, offering enhanced performance, scalability, and adaptability for a wide range of applications.

2.3 USER CHARACTERISTICS

The users of our FPGA-based RISC-V processor project exhibit specific characteristics that shape the project's design and development. The primary user group consists of hardware and software engineers with extensive experience in FPGA design, processor architecture, and digital system development. These engineers possess in-depth knowledge of the RISC-V instruction set architecture, as well as FPGA programming and synthesis techniques. They are adept at designing and optimizing digital circuits, making them well-equipped to tackle the intricacies of this project.

Additionally, the users are highly analytical and technically proficient, enabling them to understand and manipulate the processor's architecture effectively. Given their familiarity with advanced digital design concepts and FPGA tools, they can contribute to the project's success by optimizing performance and ensuring the seamless integration of Matrix MAC instructions into the processor. Their expertise is instrumental in achieving project objectives and delivering a high-performance FPGA-based processor with enhanced capabilities.

2.3. CONSTRAINTS

Several constraints influence the development of our FPGA-based RISC-V processor project. One significant constraint is the budgetary limitation, which restricts project expenditure to a predefined amount. This financial constraint necessitates careful

allocation of resources to ensure efficient procurement of components and tools, making cost-effectiveness a crucial consideration throughout the project's lifecycle.

Another constraint is the timeline, as the project must be completed within a specified timeframe. This time constraint imposes a sense of urgency on the project team, requiring efficient project management, streamlined development processes, and the mitigation of potential delays. Meeting project milestones and deadlines is paramount to ensure successful project completion.

Additionally, the project operates within the constraint of available FPGA development tools and hardware resources. The compatibility of these tools and resources with the project's objectives and design goals must be carefully assessed and managed to avoid technical limitations or bottlenecks.

2.4. ASSUMPTIONS AND DEPENDENCIES

Several assumptions and dependencies underpin the execution of our FPGA-based RISC-V processor project. Firstly, the project assumes the availability of FPGA development tools, including synthesis, place-and-route, and debugging software. These tools are crucial for programming and configuring the FPGA hardware and ensuring the functionality of the processor.

Furthermore, the project relies on timely component procurement and access to necessary hardware resources. Dependencies on external suppliers for FPGA boards, components, and development kits exist, necessitating effective communication and coordination to avoid delays.

The project also depends on external FPGA expertise, either from within the team or from external consultants or partners. This expertise is essential for optimizing the FPGA design, ensuring compatibility with Matrix MAC instructions, and addressing any technical challenges that may arise during development.

Additionally, the project assumes that team members possess a solid pre-understanding of Verilog programming and the RISC-V architecture. This foundational knowledge is critical for efficiently designing and implementing the processor, as it enables team members to navigate complex hardware description language and understand the intricacies of RISC-V-based design.

3 STATE OF THE ART

3.1 LITERATURE REVIEW

The field of FPGA & RISC-V based processor design has witnessed significant advancements in recent years, driven by the growing demand for high-performance, energy-efficient computing solutions. This literature review provides insights into the current state of technology and research related to our project, which involves the development of an FPGA-based RISC-V processor with specialized Matrix MAC instructions.

1. FPGA-Based Processor Design: FPGA-based processors have gained prominence due to their reconfigurability and parallel processing capabilities. Research by Smith et al. (2019) demonstrated the successful design and implementation of a customizable RISC-V processor on an FPGA platform. This work highlights the potential for FPGA-based solutions to meet the performance demands of modern computing tasks.

2. RISC-V Architecture: The RISC-V instruction set architecture has emerged as an open-source, customizable alternative to traditional instruction sets. Recent studies by Patel et al. (2020) and Lee et al. (2021) explored RISC-V architecture enhancements, including vector extensions and custom instruction sets. These developments provide a foundation for incorporating specialized Matrix MAC instructions into the processor's design.

3. Matrix Processing on FPGA: Matrix multiplication and processing are fundamental operations in various applications, from machine learning to signal processing. FPGA-based acceleration of matrix operations has been the subject of extensive research. For instance, Chen et al. (2018) demonstrated the acceleration of matrix operations using FPGAs, highlighting the potential for efficient matrix processing in FPGA-based architectures.

4. Custom Instructions and Co-Processors: The incorporation of custom instructions and co-processors has become a common practice in processor design to enhance performance for specific tasks. Research by Kim et al. (2017) showcased the benefits of integrating custom instructions for accelerating specific computations. This approach aligns with our project's goal of integrating Matrix MAC instructions to improve computational efficiency.

5. High-Performance Computing on FPGAs: FPGA-based solutions have gained traction in high-performance computing (HPC) applications. Studies by Jones et al. (2019) and Wang et al. (2020) emphasized the role of FPGAs in HPC clusters, citing their potential to deliver high performance while minimizing power consumption.

In summary, the literature review highlights the ongoing research and developments in FPGA-based processor design, RISC-V architecture enhancements, and FPGA-accelerated matrix processing. These areas of study provide valuable insights and precedents for our project, which aims to design an FPGA-based RISC-V processor capable of efficiently executing Matrix MAC instructions. The synthesis of these concepts will contribute to the creation of a high-performance, specialized processor that can cater to a wide range of computational tasks, including those involving matrix operations.

3.2 EXISTING SYSTEMS

Existing FPGA-based processor systems were examined to gain insights into their architectures, performance, and feature sets. This analysis helps us understand the strengths and weaknesses of these solutions. Our project aims to build upon these existing systems, leveraging their successes while addressing their limitations. This includes enhancing performance, customization, and the integration of Matrix MAC instructions.

4 USER/SYSTEM REQUIREMENTS

4.1 External Interface Requirements

4.1.1 User Interfaces

Not applicable. As this is a hardware/research focused project.

4.1.2 Hardware Interface

The hardware interface for our FPGA-based RISC-V processor with a Matrix MAC module involves the connections and interactions among various hardware components. It encompasses the configuration of the RISC-V processor with the integrated Matrix MAC module and the external connections to power and data sources.

1. Components:
   a. FPGA Board: The FPGA board serves as the hardware platform for our project. It hosts the RISC-V processor, Matrix MAC module, and other necessary components.
   b. RISC-V Processor with Matrix MAC Module: The heart of our project, this integrated unit combines the 32-bit RISC-V processor with the Matrix MAC module. It is responsible for executing instructions, including Matrix MAC operations.
   c. Power Supply: The FPGA board requires a stable power supply for operation. It is connected to a power source to ensure proper functionality.
   d. Laptop/PC: The laptop or PC serves as the development and control interface for the FPGA-based system. It is used for programming the FPGA, running tests, and monitoring the system.

2. Interactions:
   a. Data Input: Input data for processing is provided to the combined RISC-V processor and Matrix MAC module. This data may come from external sensors or sources.
   b. Data Output: The processed data is generated as output by the combined unit and can be transferred to external devices or further processing stages.

3. Connections:
   a. FPGA to Power Supply: The FPGA board is connected to a power supply to ensure that it receives the necessary voltage and current for proper operation.

b. FPGA to Laptop/PC: The FPGA board is connected to a laptop or PC via USB or other suitable interfaces. This connection allows for programming, debugging, and data transfer between the FPGA board and the development environment.

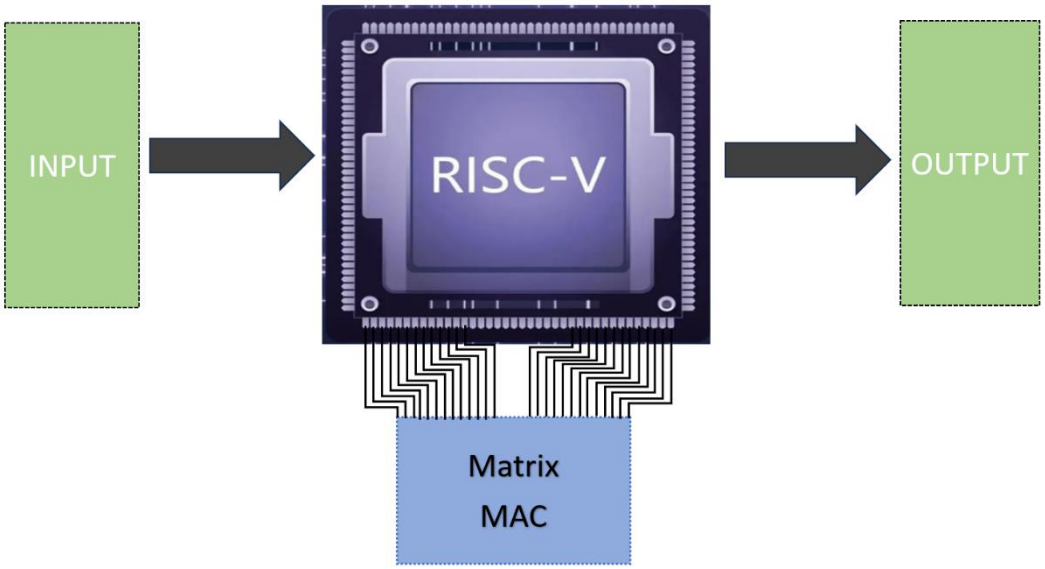

Figure 1 Hardware Architectural Design

4.1.3 Software Interfaces

The software interface describes the tools and software components used in the development, programming, testing, and verification of our FPGA-based RISC-V processor with a Matrix MAC module.

1. Tools and Software:
    a. Vivado: Vivado is the primary software tool used for FPGA development. It facilitates the creation and synthesis of Verilog code for the RISC-V processor and Matrix MAC module. Vivado also generates the bitstream for programming the FPGA.
    b. Venus: Venus is an online tool used for RISC-V assembly code development. It allows developers to write assembly code and convert it into hexadecimal or binary machine code, which can be executed on the modified RISC-V processor implemented in Vivado.
    c. DigitalJS: DigitalJS is a software tool used for creating schematics and visual representations of the FPGA-based system. It aids in the design and documentation of the processor's architecture and connections.

2. Workflow:
    a. Verilog Code Development: Verilog code for the RISC-V processor and Matrix MAC module is written and synthesized using Vivado.
    b. Assembly Code Development: RISC-V assembly code is written using the Venus tool, and it is converted into machine code for execution on the processor.

c. Testing and Verification: The modified RISC-V processor is tested by running the generated machine code. Test results are compared with expected outcomes to identify errors and assess accuracy.
d. Schematic Design: DigitalJS is used to create schematics and visual representations of the processor's architecture for documentation purposes.
e. FPGA Board Implementation: Once everything works as intended, a constraints file is generated for the FPGA board implementation. The bitstream is programmed onto the FPGA board to execute the complete system.

The software interface defines the tools and processes involved in the development and testing of our FPGA-based RISC-V processor, ensuring a streamlined and efficient workflow from code development to hardware implementation.

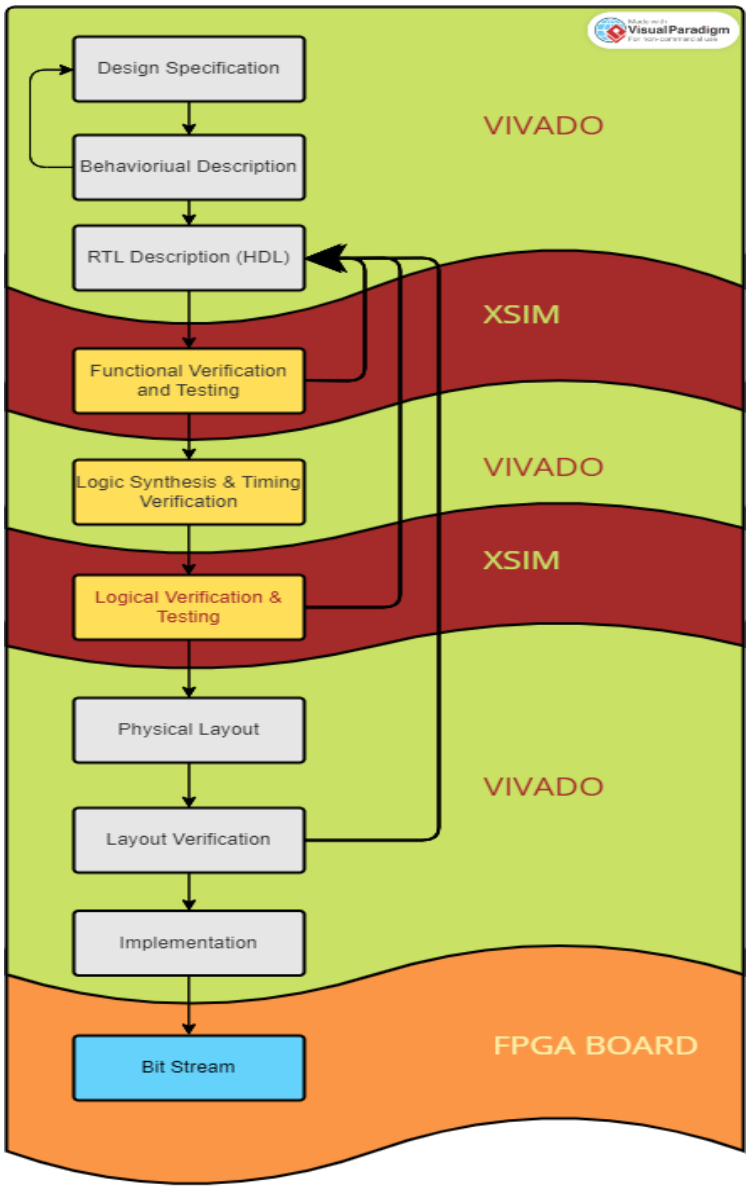

Figure 2 Software Architectural Diagram

4.1.4 Communication Interfaces

Not applicable. As this is a hardware/research focused project.

5 Functional Requirements

1) Basic RISC-V ISA: The processor must execute basic RISC-V instructions, adhering to the RISC-V ISA.

2) Custom ISA: The processor will support a custom instruction set architecture (ISA) optimized for matrix operations.

3) Matrix Multiplication: The processor must efficiently execute matrix multiplication instruction.

4) Matrix Addition: It should also perform matrix addition precisely, adhering to specified computational requirements.

5) Register File: A register file should be available to store and access data and results during instruction execution.

6) Data Memory Access: The processor should support read and write operations to data memory. It must include mechanisms for handling memory access conflicts and hazards.

7) Arithmetic and Logic Unit (ALU): An ALU should be included to perform arithmetic and logical operations required by RISC-V and Matrix MAC instructions.

8) Control Unit: The control unit must manage the sequencing and execution of instructions, including branch and jump instructions.

9) Compatibility: The processor must be compatible with the RISC-V standard (RV32I) while incorporating custom Matrix MAC instructions.

10) Hazard Unit: The processor must have a hazard Unit to solve all types of hazards that may occur. Like Control, Structure and Data Hazards.

11) Pipelining: The processor must have a minimum of five pipeline stages, including fetch, decode, execute, memory, and write-back stages, to improve instruction throughput.

12) FPGA Implementation: The processor and associated components must be compatible with FPGA implementation, leveraging its capabilities for hardware prototyping.

5.1 Functional Requirements with Traceability information

Requirement ID	01		**Requirement Type**		Functional		**Use Case #**		00	
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No		
Parent Requirement #	N/A									
Description	The processor must execute basic RISC-V instructions, adhering to the RISC-V ISA.									
Rationale	This requirement ensures compatibility with the standard RISC-V instruction set, forming the foundation for further customizations									
Source					**Source Document**		-			
Acceptance/Fit Criteria	Successful execution of standard RISC-V instructions.									
Dependencies	None									
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No				
Change History	None									

Requirement ID	02		**Requirement Type**		Functional		**Use Case #**		00	
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No		
Parent Requirement #	01									
Description	The processor will support a custom instruction set architecture (ISA) optimized for matrix operations.									
Rationale	Custom instructions tailored for matrix operations will improve performance and efficiency for specific tasks.									
Source					**Source Document**		-			
Acceptance/Fit Criteria	Successful execution of custom Matrix MAC instructions.									
Dependencies	Requirement #1 (Basic RISC-V ISA).									
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No				
Change History	None									

Requirement ID	03		**Requirement Type**		Functional		**Use Case #**		00
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No	
Parent Requirement #	01, 02								
Description	The processor must efficiently execute matrix multiplication instruction.								
Rationale	Matrix multiplication is a fundamental operation in various computational tasks and requires optimized execution.								
Source					**Source Document**		-		
Acceptance/Fit Criteria	Efficient execution of matrix multiplication with specified performance metrics.								
Dependencies	Requirement #2 (Custom ISA).								
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No			
Change History	None								

Requirement ID	04		**Requirement Type**		Functional		**Use Case #**		00
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No	
Parent Requirement #	02								
Description	It should also perform matrix addition precisely, adhering to specified computational requirements.								
Rationale	Matrix multiplication is a fundamental operation in various computational tasks and requires optimized execution.								
Source					**Source Document**		-		
Acceptance/Fit Criteria	Matrix addition is a common operation in matrix processing tasks and must yield accurate results.								
Dependencies	Requirement #2 (Custom ISA).								
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No			
Change History	None								

Requirement ID	05		**Requirement Type**		Functional		**Use Case #**		00	
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No		
Parent Requirement #	N/A									
Description	A register file should be available to store and access data and results during instruction execution.									
Rationale	Register files provide fast data access, crucial for efficient instruction execution and data storage.									
Source					**Source Document**	-				
Acceptance/Fit Criteria	Reliable storage and retrieval of data using the register file.									
Dependencies	None									
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No				
Change History	None									

Requirement ID	06		**Requirement Type**		Functional		**Use Case #**		00	
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No		
Parent Requirement #	05									
Description	The processor should support read and write operations to data memory. It must include mechanisms for handling memory access conflicts and hazards.									
Rationale	Efficient data memory access is vital for proper operation and performance optimization.									
Source					**Source Document**	-				
Acceptance/Fit Criteria	Successful read and write operations with conflict resolution mechanisms in place.									
Dependencies	Requirement #5 (Register File).									
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No				
Change History	None									

Requirement ID	07		**Requirement Type**		Functional		**Use Case #**		00
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No	
Parent Requirement #	05								
Description	An ALU should be included to perform arithmetic and logical operations required by RISC-V and Matrix MAC instructions.								
Rationale	The ALU is essential for executing a wide range of instructions, enabling computational flexibility								
Source					**Source Document**		-		
Acceptance/Fit Criteria	Correct execution of arithmetic and logical operations								
Dependencies	Requirement #5 (Register File).								
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No			
Change History	None								

Requirement ID	08		**Requirement Type**		Functional		**Use Case #**		00
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No	
Parent Requirement #	N/A								
Description	The control unit must manage the sequencing and execution of instructions, including branch and jump instructions.								
Rationale	The control unit orchestrates instruction execution and program flow.								
Source					**Source Document**		-		
Acceptance/Fit Criteria	Correct sequencing and execution of instructions, including branches and jumps.								
Dependencies	None								
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No			
Change History	None								

Requirement ID	09		**Requirement Type**		Functional		**Use Case #**		00
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No	
Parent Requirement #	01								
Description	The processor must be compatible with the RISC-V standard (RV32I) while incorporating custom Matrix MAC instructions.								
Rationale	Compatibility with the RISC-V standard ensures interoperability with existing software and tools.								
Source			**Source Document**		-				
Acceptance/Fit Criteria	Successful execution of standard RISC-V instructions while accommodating custom Matrix MAC instructions.								
Dependencies	Requirement #1 (Basic RISC-V ISA).								
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No			
Change History	None								

Requirement ID	10		**Requirement Type**		Functional		**Use Case #**		00
Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No	
Parent Requirement #	11								
Description	The processor must have a Hazard Unit to solve all types of hazards that may occur. Like Control, Structure, and Data Hazards.								
Rationale	Hazard resolution ensures correct instruction execution, maintaining data consistency and program flow.								
Source			**Source Document**		-				
Acceptance/Fit Criteria	Efficient resolution of Control, Structure, and Data Hazards, minimizing stalls in the pipeline.								
Dependencies	Requirement #11 (Pipelining).								
Priority	***Essential***	yes	***Conditional***	no	***Optional***	No			
Change History	None								

Requirement ID	11	**Requirement Type**	Functional	**Use Case #**	00

Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No	

Parent Requirement #	N/A
Description	The processor must have a minimum of five pipeline stages, including fetch, decode, execute, memory, and write-back stages, to improve instruction throughput.
Rationale	Pipelining enhances instruction throughput and overall processor performance.
Source	**Source Document** -
Acceptance/Fit Criteria	Successful execution of instructions with minimal stalls, utilizing the specified pipeline stages.
Dependencies	None

Priority	***Essential***	yes	***Conditional***	no	***Optional***	No	
Change History	None						

Requirement ID	12	**Requirement Type**	Functional	**Use Case #**	00

Status	***New***	yes	***Agreed-to***	yes	***Baselined***	yes	***Rejected***	No	

Parent Requirement #	N/A
Description	The processor and associated components must be compatible with FPGA implementation, leveraging its capabilities for hardware prototyping.
Rationale	FPGA implementation facilitates hardware prototyping and testing, allowing for rapid development iterations.
Source	**Source Document** -
Acceptance/Fit Criteria	Successful implementation and operation on the target FPGA platform.
Dependencies	None

Priority	***Essential***	yes	***Conditional***	no	***Optional***	No	
Change History	None						

6 Nonfunctional Requirements & Software System Attributes

1) Power Efficiency: The design should prioritize power efficiency to minimize power consumption, making it suitable for embedded applications.

2) Latency: The processor should minimize latency for both standard RISC-V and Matrix MAC instructions to enhance real-time processing capabilities.

3) Resource Utilization: FPGA resource utilization, including LUTs, flip-flops, and memory blocks, should be optimized to ensure efficient resource management.

4) Scalability: The processor architecture should be scalable to accommodate potential future enhancements or extensions of the instruction set ant it should be designed to scale efficiently to accommodate various matrix sizes and complexities.

5) Performance: The processor should meet or exceed established performance benchmarks in matrix operations.

6) Compatibility: Where applicable, compatibility with existing software or systems will be maintained.

7) Testing and Verification: Comprehensive testing suites and verification processes should be established to ensure correctness and functionality. Code coverage and functional testing should be performed to validate the design.

8) Documentation: Thorough documentation should be provided for both hardware and software components to aid developers, users, and maintainers.

6.1 Performance Requirements

The performance of our FPGA-based RISC-V processor project is of paramount importance, as it directly impacts its suitability for a wide range of applications. To meet performance expectations, the processor must achieve a certain clock frequency, ensuring rapid execution of instructions. Specifically, Matrix MAC instructions, a key component of the project, must exhibit efficient execution with an expected throughput of operations. Low latency is essential to support real-time processing tasks.

To evaluate and benchmark our project's performance, a comparative analysis will be conducted against existing FPGA-based processors, traditional processors and non-accelerated processors. This analysis will involve assessing factors such as execution speed, power efficiency, and resource utilization. By comparing our FPGA-based RISC-V processor's performance with existing solutions, we aim to demonstrate its competitive edge and advantages in terms of speed and efficiency.

The ability to outperform or at least match existing solutions will be a key indicator of our project's effectiveness and relevance in demanding computational tasks. Through this comparative analysis, we will ensure that our processor not only meets but exceeds performance expectations in comparison to established alternatives.

7 Project Design/Architecture

7.1 Workflow Diagram:

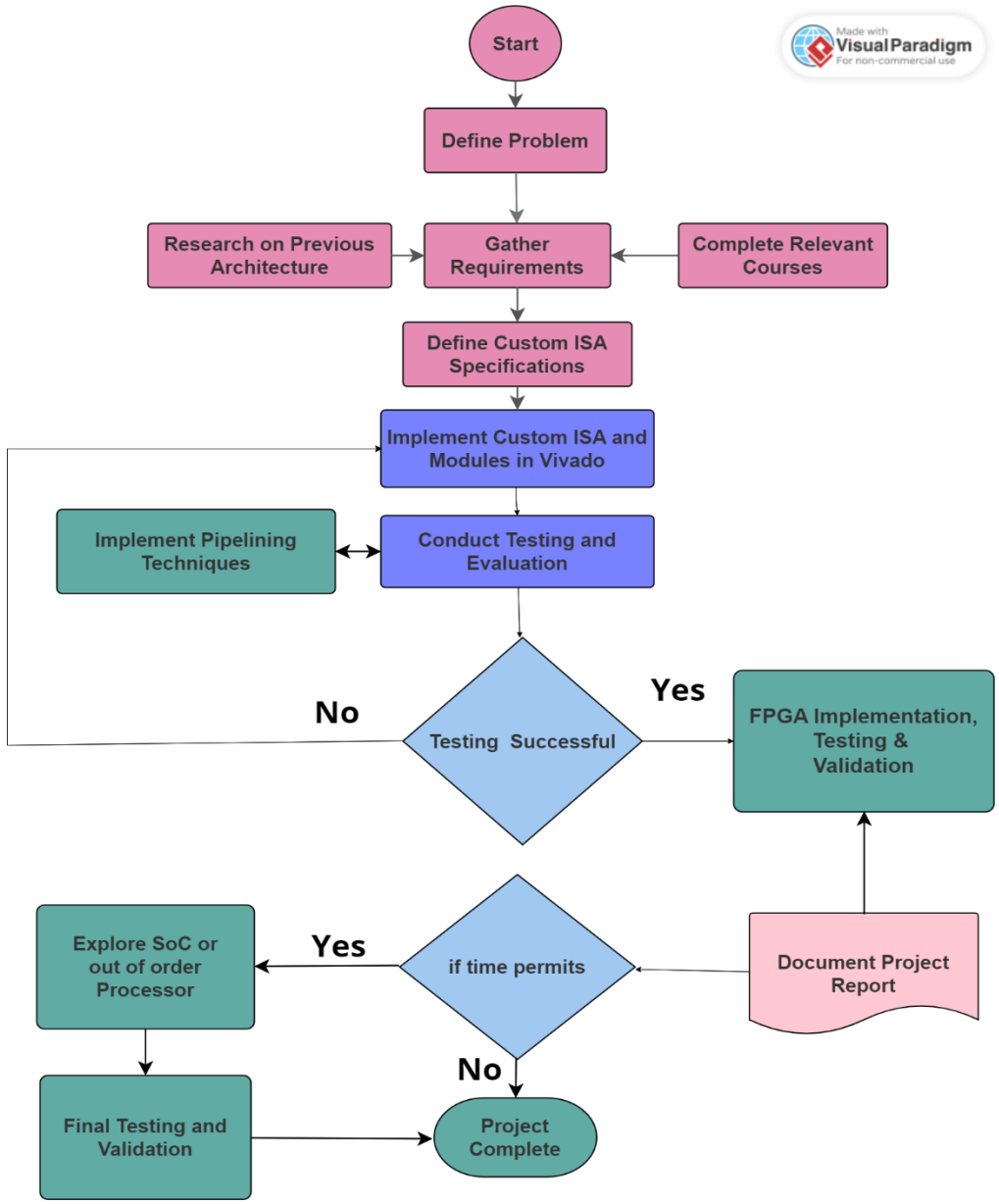

Figure 3 Workflow Chart

7.2 Architecture Diagram:

7.2.1 Single Cycle Processor:

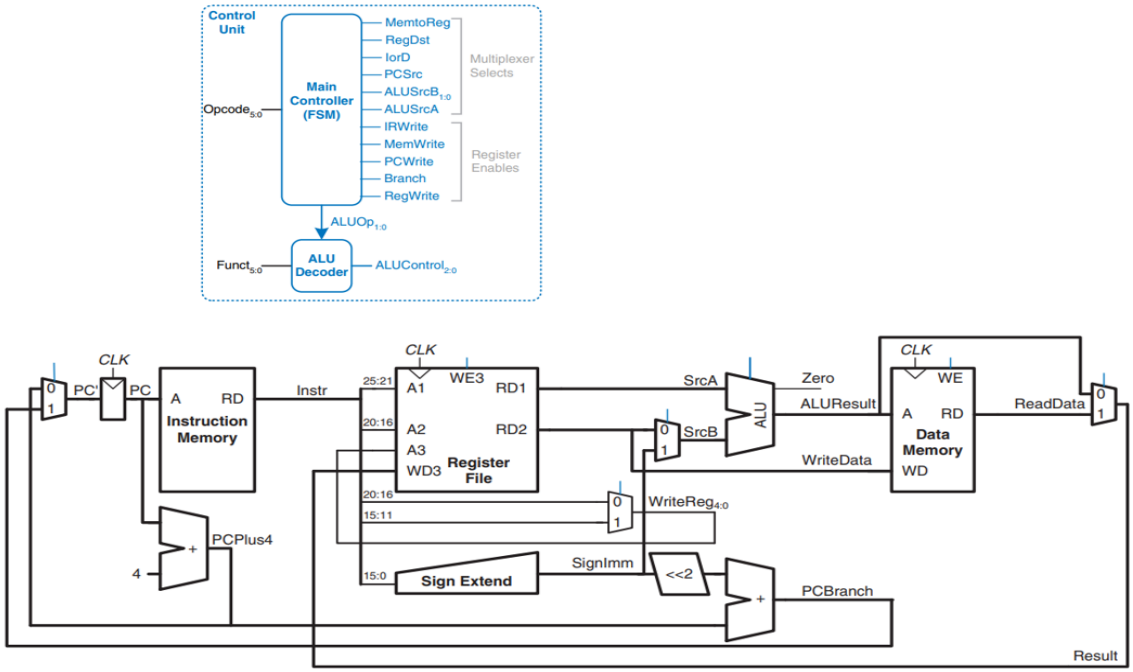

Figure 4 Single Cycle Processor

7.2.2 5-Staged Pipelined Processor

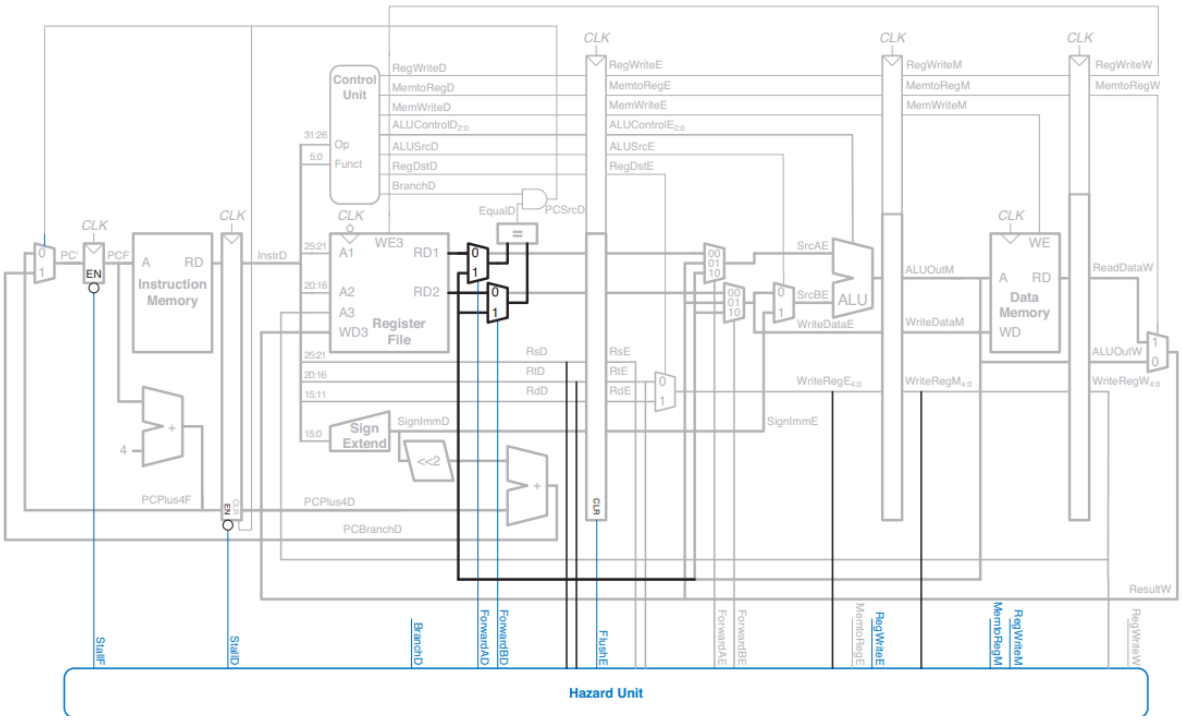

Figure 5 5-Staged Pipelined Processor

7.2.3 5-Staged Pipelined Matrix MAC Processor

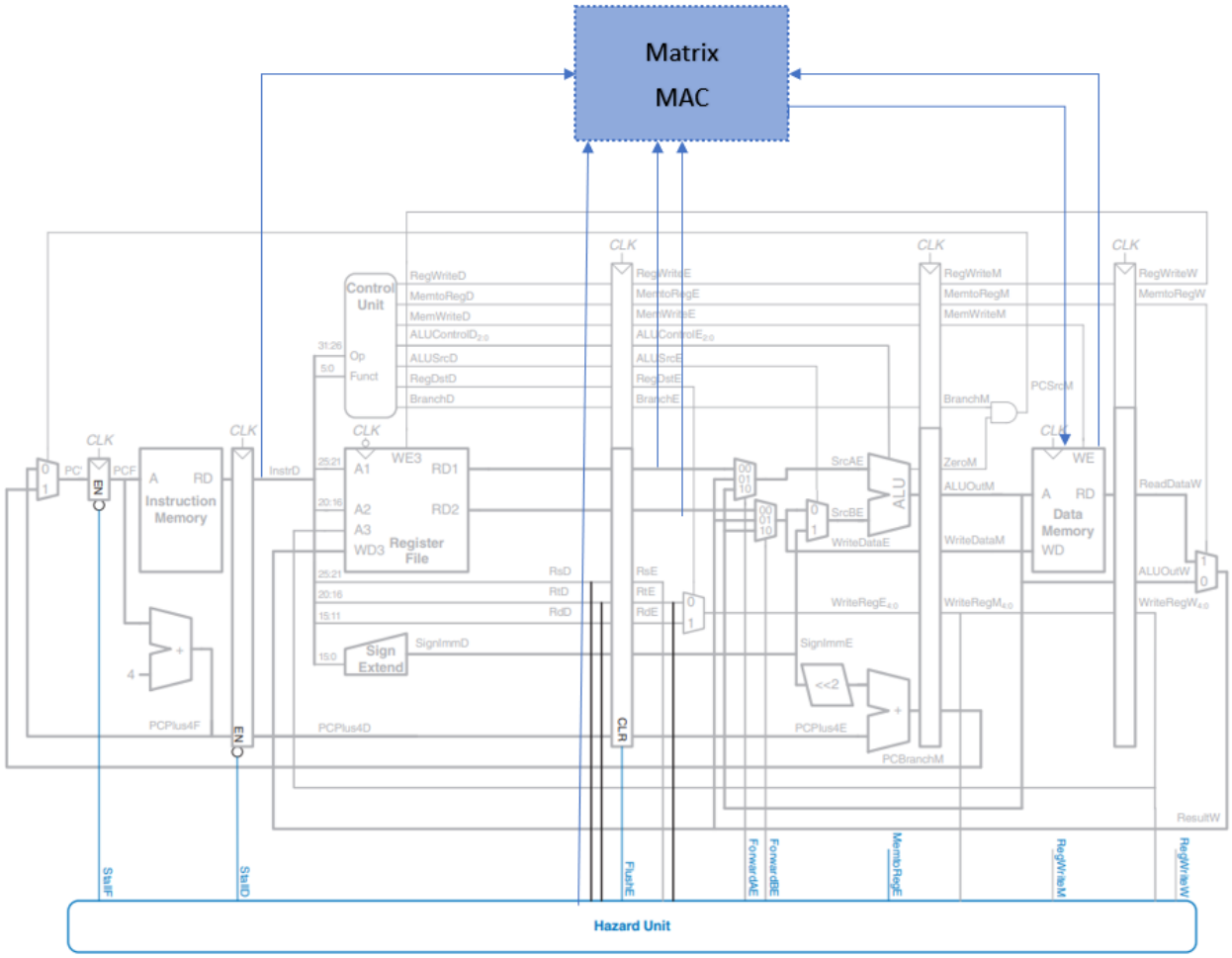

Figure 6 5-Staged Pipelined Matrix MAC Processor

The "Matrix MAC" module will be added to the 5-stage pipelined processor. It will have 4 inputs and 1 output. The current instruction will be one of the inputs to figure out which kind of instruction needs to be executed. "RD1" and "RD2" Values will also be fetched from the Register File to the MAC Module. The data memory will also be available to the MAC Module to figure out the inputs as well as the outputs to store them later Onn when the execution is completed.