_-A PROJECT REPORT

on

"RISC-V 3-stage Pipeline Processor build using RV32I instruction set"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE AND ENGINEERING

BY

ANIRUDDH SINGH	2105769
KHUSHI SINGH	2105803
SADAF SHAHAB	21051590
MEHUL AGARWAL	21051479
RADHIKA MANISH	2105730

UNDER THE GUIDANCE OF

Prof. Abhishek Raj

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA - 751024

May 202

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is to certify that the project entitled

"RISC-V 3-stage pipeline processor
build using RV32I instruction set"

submitted by

ANIRUDH SINGH	2105769
KHUSHI SINGH	2105803
SADAF SHAHAB	21051590
MEHUL AGARWAL	21051479
RADHIKA MANISH	2105730

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work will be done during the year 2024-2025, under our guidance.

Date: 4 /04 /24

(Prof. ABHISHEK RAJ)
Project Guide

Acknowledgements

I would like to express my sincere gratitude to **ABHISHEK RAJ** sir for his invaluable guidance and support throughout my project. His extensive knowledge, insightful feedback, and continuous encouragement have been instrumental in the successful completion of this project. I am grateful for his dedication and patience in answering all my queries and providing me with valuable inputs that helped me to improve my work.
Thank you, **ABHISHEK RAJ** sir, for being an excellent mentor and for sharing your expertise with me. Your guidance has been a source of inspiration to me, and I look forward to applying the knowledge gained from this project in my future endeavors."

ANIRUDH SINGH
KHUSHI SINGH
SADAF SHAHAB
MEHUL AGARWAL
RADHIKA MANISH

ABSTRACT

This project describes the design and development of a RISC-V 3-stage pipeline processor based on the RV32I instruction set architecture. RISC-V is an open-source instruction set architecture that has gained popularity in recent years for its simplicity, extensibility, and modularity. The aim of this thesis is to design an open-source RISC-V processor. SystemVerilog was used for RTL coding. ModelSim was used for RTL simulation. The processor architecture adheres to the standard RISC (Reduced Instruction Set Computing) principles, with a smaller instruction set and a three-stage pipeline consisting of fetch, decode, and execute stages. The RV32I instruction set has a broad collection of instructions for integer arithmetic, logical, and control operations, making it appropriate for a variety of computing applications.

The instruction set RV32IM is the Instruction Set base for this processor. Through simulation, the CPI of this RISC-V processor can be collected while running different benchmark programs developed in two parallel Master theses to this one. To a certain extent, it can reflect the performance of the processor. Industry-standard simulation and synthesis techniques are used to analyze performance parameters such as clock frequency, instruction throughput, and pipeline efficiency. The results show that the RISC-V 3-stage pipeline processor can achieve great performance and efficiency while adhering to the RISC-V architecture's goals of simplicity and openness. Overall, this project adds to the continued development and use of RISC-V-based processors in a variety of computer applications.

Keywords: RISC-V, ISA, SystemVerilog, RTL simulation, RV32IM, CPI

.

Contents

List of Figures

Chapter 1

Introduction

In the constantly changing field of computer architecture, the RISC-V instruction set architecture has emerged as a potential open standard that provides flexibility, scalability, and adaptation to a variety of computing settings. As the need for efficient and customizable processors grows, there is a critical need for novel architectures that make use of the RISC-V ISA's characteristics to handle modern computing difficulties. In answer to this demand, this project intends to develop and construct a RISC-V 3-stage pipeline processor based on the RV32I instruction set that meets the needs of current computing systems.

This project is significant in that it contributes to the evolution of processor design approaches based on the RISC-V ISA's ideals of simplicity, openness, and modularity. This project aims to strike a compromise between speed and complexity by designing a 3-stage pipeline processor that allows for fast instruction execution while being streamlined in design. Furthermore, using the RV32I instruction set improves compatibility with current software ecosystems, providing easy integration and interoperability across a wide range of applications.

The structure of this report includes a thorough examination of the design and implementation of the RISC-V 3-stage pipeline processor. The parts that follow will dive into the processor's architectural aspects, such as pipeline stage organization, instruction set encoding, data forwarding methods, and control logic. Furthermore, performance parameters like as clock frequency, instruction throughput, and pipeline efficiency will be examined using industry-standard simulation and synthesis tools. By providing thorough insights into the development process and performance analysis, this study intends to be a significant resource for academics, engineers, and hobbyists interested in RISC-V-based CPU design.

Chapter 2

Basic Concepts/ Literature Review

RISC-V is an open standard instruction set architecture (ISA) based totally on reduced instruction set computing (RISC) principles. A 3-level pipeline processor is a simplified CPU structure that executes commands in 3 stages: instruction Fetch (IF), instruction Decode (ID), and Execute (EX).

2.1 Concepts associated with RISC-V 3-Stage pipeline processor:

1. RISC-V ISA (RV32I):
- RV32I is a subset of the RISC-V ISA designed for 32-bit architectures.
- It includes 32 general-purpose registers (x0 to x31) and a fixed set of fundamental instructions.
- primary instruction classes encompass arithmetic, logic, memory access, control transfer, etc.
- instructions are encoded into 32-bit words with a fixed-length format.

2. three-stage Pipeline:
- Instruction Fetch (IF): Fetches the instruction from memory pointed to by means of the program counter (computer).
- Instruction Decode (ID): Decodes the fetched instruction and reads any necessary register operands.
- Execute (EX): Executes the operation distinctive by using the instruction. for example, ALU operations, reminiscence accesses, and many others.

3. Registers:
- The processor has a set of general-purpose registers (x0 to x31) for temporary information storage.
- special-purpose registers include the program Counter (laptop), which holds the address of the next instruction to be fetched.

4. memory access:
- RV32I makes use of Load and store instructions for memory access.
- Load instructions transfer information from memory to registers, at the same time as store instructions switch data from registers to memory.

5. Arithmetic and Logic Operations:
- RV32I supports simple arithmetic operations like addition, subtraction, multiplication, and division.
- Logical operations inclusive of AND, OR, XOR, and shift operations also are available.

6. Control transfer instructions:
- branch instructions allow conditional jumps based on certain situations.
- jump (JAL) and jump and link (JALR) commands facilitate unconditional jumps and subroutine calls.

7. Hazard handling:
- In pipelined processors, hazards may also arise because of data dependencies, control dependencies, or structural hazards.
- techniques like forwarding, stalling, and branch prediction are used to mitigate hazards and maintain pipeline efficiency.

8. Instruction Fetch Unit (IFU):
- responsible for fetching commands from memory using the computer.
- Handles branching and jumping instructions by updating the laptop consequently.

9. Instruction Decode Unit (IDU):
- Decodes the fetched instruction, figuring out the operation to be executed and any required operand values.
- Reads the register report for source operands.

10. Execution Unit (EXU):
- plays the actual execution of arithmetic, logic, and control transfer operations.
- ALU operations, memory accesses, and branch decisions are commonly handled at this stage.

Building a RISC-V 3-stage pipeline processor entails designing and enforcing these components, ensuring the right instruction execution, risk handling, and overall performance optimization within the constraints of the RV32I ISA and the pipeline architecture.

2.2 Challenges in building RISC-V 3-stage pipeline processor using RV32I instruction set

1. Pipeline hazards: Pipeline hazards arise when an instruction depends on the result of a preceding instruction that has not yet finished execution. This will cause stalls in the pipeline, lowering overall performance. managing hazards efficiently through techniques like forwarding, stalling, or department prediction is vital to maintaining pipeline throughput.

2. Data Dependencies: Dependencies between instructions, which include data hazards (e.g., read-after-write dependencies), can result in stalls or incorrect outcomes if not dealt with properly. enforcing forwarding logic and scheduling instructions to reduce dependencies is important for efficient pipeline operation.

3. Control hazards: Control risks arise up whilst the pipeline needs to stall because of conditional branches or jumps, impacting throughput. strategies like branch prediction or speculative execution can mitigate the effect of control hazards by predicting branch results or executing instructions ahead of confirmation.

4. Instruction Fetch Bandwidth: In a three-level pipeline, the instruction fetch level needs to fetch commands from memory quickly to maintain the pipeline filled and keep high throughput. memory latency, cache misses, and instruction cache size can all have an effect on the fetch bandwidth and average overall performance.

5. Pipeline Balancing: Balancing the pipeline stages to ensure that each stage completes in more or less the same amount of time is vital for optimal performance. Mismatches in level latencies can cause pipeline bubbles or inefficiencies, decreasing basic throughput.

6. Resource contention: In a simplified pipeline layout, useful resource competition can arise whilst more than one commands compete for the same hardware sources simultaneously. dealing with resources like the ALU, memory access units, and register files effectively to limit contention is vital for maintaining pipeline throughput.

7. Instruction Set Extensions: while RV32I gives a basic set of commands, many applications may take advantage of additional instruction set extensions for specialized tasks (e.g., SIMD, cryptography). Integrating and dealing with those extensions whilst retaining compatibility and ease within the pipeline design can be difficult.

8. Testing and Verification: Verifying the correctness and overall performance of a pipeline processor design is a complex mission that requires complete testing and verification methodologies. ensuring that the processor behaves effectively in various situations and meets performance requirements involves rigorous testing, simulation, and debugging efforts.

9. Power and energy performance: at the same time as a 3-level pipeline is rather simple, optimizing power and energy efficiency remains critical, mainly for embedded and cell applications. strategies like clock gating, voltage scaling, and dynamic power management need to be taken into consideration to limit energy intake without sacrificing performance.

Chapter 3

Problem Statement / Requirement Specifications

The project aims to lay out and enforce a three-stage pipeline processor primarily based on the RISC-V architecture, particularly focused on the RV32I instruction set. The processor has to efficiently execute a subset of RISC-V instructions at the same time as adhering to pipeline design concepts to obtain improved performance.

3.1 Project Planning

The project planning phase for the RISC-V 3-stage pipeline processor build using the RV32I instruction set involved understanding RISC-V and HDL(Hardware Description Language- Verilog Programming Language) for RTL coding following which Modelsim platform was used for RTL simulations.
We defined the architecture of the three-stage pipeline processor primarily based on the RV32I instruction set and documented architecture design specifications such as pipeline stages, instruction sets, and data paths. The team enforced the Fetch stage in hardware description language (HDL) (Verilog), the Decode stage, focusing on instruction decoding and register file access and the Execute stage, including arithmetic/logic gadgets and control logic to complete the implementation of all 3 pipeline levels and carry out initial integration testing. We implemented forwarding and hazard detection mechanisms to deal with records hazards and memory access stages for load and store instructions, addressing memory-associated hazards to ensure the right functioning of hazard handling mechanisms and memory access operations. We Optimized the pipeline layout for performance metrics consisting of throughput and latency and developed a complete test suite covering instruction execution, hazard handling, and pipeline
behavior to conduct overall performance evaluation and validate the processor design via rigorous testing.

3.2 Project Analysis

Non-functionally, achieving a balance among overall performance and resource usage is paramount, along meeting timing constraints for correct synchronization and operation. Modularity and scalability are vital for accommodating future improvements or changes. regardless of the project's capability, numerous ambiguities and risks exist. Ambiguities may also arise in the interpretation of certain instruction types, hazard detection/decision mechanisms, and exception handling strategies. mistakes may cause wrong program execution or performance degradation, while compliance issues with the RISC-V specification pose additional risks.

To mitigate those risks, thorough testing, such as unit, integration, and functional tests, is essential. continuous verification against the RISC-V specification and rigorous code reviews can assist identify and rectify deviations or non-compliance problems. performance profiling tools aid in identifying and optimizing essential paths in the processor design, even as comprehensive documentation fosters collaboration and knowledge sharing amongst group members. by conducting a comprehensive project evaluation and implementing effective mitigation strategies, the task can successfully deliver a reliable and efficient RISC-V processor.

3.3 System Design

3.3.1 Design Constraints

The system design for our RISC-V three-stage pipeline processor project, built across the RV32I instruction set architecture, concerned a comprehensive technique integrating Verilog, C code, ModelSim for RTL simulation, Icarus Verilog for compilation, and Python for overall performance evaluation. In Verilog, we meticulously crafted the pipeline stages - fetch, decode, and execute - making ensure adherence to RISC standards of simplicity and efficiency. ModelSim facilitated rigorous RTL simulation, permitting us to validate the functionality and overall performance of our processor design. To similarly ensure robustness, we applied C code for pipeline testing, producing various test cases to evaluate the processor's behavior under diverse scenarios. Additionally, Icarus Verilog performed an important role in compiling the Verilog code into executable simulations, permitting thorough testing and validation of our system design. Python scripts have been employed to analyze simulation traces, profile the performance of the RISC-V processor layout, and identify essential paths, bottlenecks, and areas for optimization. This information guided design changes aimed at enhancing overall performance and efficiency. Via this included approach, we were able to iteratively refine and validate our RISC-V processor design, ensuring its correctness and effectiveness in handling various computing tasks.

3.3.2 System Architecture OR Block Diagram

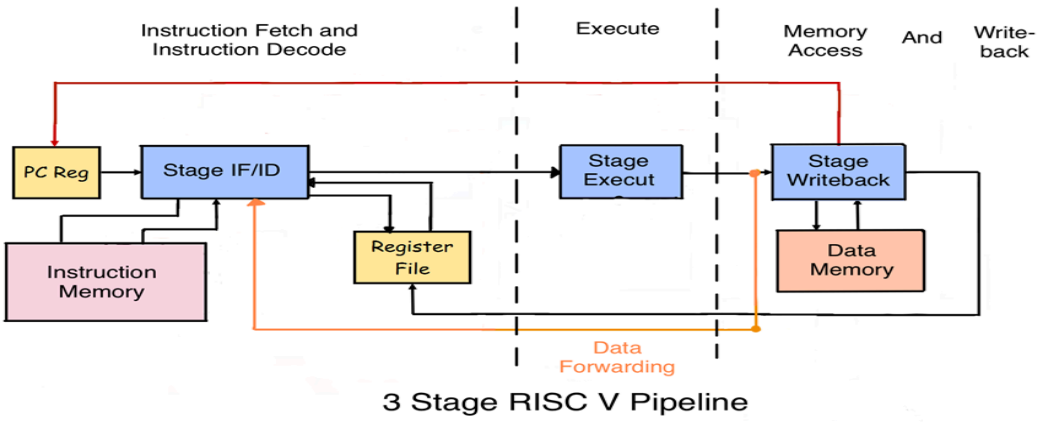

Fig 3.3.2.1 3 stage RISC V Pipeline

The RISC-V three-stage pipeline processor follows a streamlined procedure for executing instructions. within the instruction Fetch (IF) level, instructions are fetched from memory primarily based at the current program counter. The fetched instructions are then decoded within the instruction Decode (id) stage, where control indicators are generated and operand values are fetched from registers. Subsequently, in the Execution (EX) stage, arithmetic, logic, or memory operations are achieved primarily based on the decoded instructions. Throughout the pipeline, forwarding and hazard detection mechanisms are employed to ensure correct instruction execution and maintain pipeline performance. This pipeline design allows efficient instruction processing by overlapping the execution of more than one instruction in different stages of the pipeline.

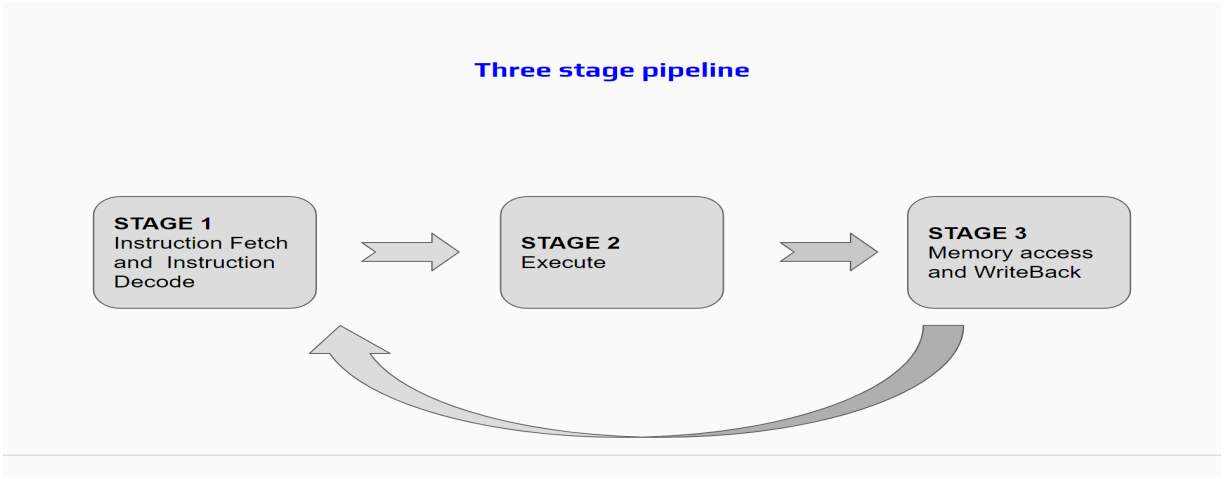

Fig 3.3.2.2 Three stage pipeline

The 3-stage pipeline architecture for a processor streamlines instruction execution into distinct stages: instruction fetch and decode, execute, and memory access and writeback. In the first stage, instruction fetch and decode, the processor retrieves commands from memory primarily based on the current program counter (computer). concurrently, in the decode phase, these fetched instructions are decoded to determine their types and required operations. control signals are generated primarily based on the opcode and instruction fields, preparing the instruction for execution. As the decoded instructions flow to the second stage, the execution stage, arithmetic, logic, or memory operations are finished according to the decoded instructions. ALU operations, memory accesses (load/store), and branch calculations occur here. This stage executes the core functionality of the commands. Finally, in the third stage, memory access and writeback, the outcomes of the finished commands are accessed from memory, and if necessary, they may be written and returned to the register file. This stage completes the instruction cycle by ensuring that data is effectively stored or retrieved from memory. Moreover, the third level is connected back to the primary level, developing a feedback loop. This connection permits for efficient instruction processing by permitting the processor to fetch the subsequent instruction while concurrently performing memory access and write back operations for the current instruction, thereby maximizing pipeline throughput and standard performance.

Chapter 4

Implementation

The implementation of the "RISC-V three-stage pipeline processor construct the usage of RV32I instruction set" project revolves round Verilog coding, simulation, compilation, and testing using numerous tools. Verilog serves as the primary language for developing the processor core, where modules are meticulously crafted to execute features aligned with the RV32I instruction set architecture. ModelSim comes into play for simulation purposes, permitting us to rigorously test the Verilog code. Through the advent of testbench modules, we can confirm that the processor core behaves as expected, executing commands accurately and generating preferred outputs. post-simulation, Icarus Verilog helps code compilation, changing the Verilog code into an executable illustration of the processor core. In the end, Imperas OVP emerges as an essential device for testing the performance and correctness of the designed processor core. In the OVP environment, comprehensive testing is performed, making sure that the processor core meets performance expectations, complies with the RV32I instruction set specification, and is prepared for further improvement or deployment in real-world packages. This holistic approach to implementation guarantees thorough testing and validation of the processor core, laying the foundation for its successful integration into future projects or systems.

4.1 Methodology OR Proposal

Method for building a RISC-V 3-level Pipeline Processor with the usage of RV32I preparation Set:

1. Understanding RISC-V structure and RV32I Instruction Set:
- Thoroughly understanding the RISC-V architecture specs, particularly focusing on the RV32I base integer instruction set.
- familiarizing with the distinct instruction formats, instruction types, register file organization, memory addressing modes, and fundamental arithmetic/logic operations.

2. Design Specification and requirements gathering:
- outline the specs of the three-stage pipeline processor inclusive of clock frequency, instruction set architecture (ISA) compliance, pipeline stages, and overall performance expectancies.
- gather requirements from stakeholders concerning supported instructions, overall performance goals, and any precise functions required.

3. Pipeline structure design:
- designing a three-stage pipeline structure which includes Fetch, Decode, and Execute stages.
- define the control signals for each level and their interactions.
- determine the data paths for instruction fetch, register document access, ALU operations, and memory access.
- design hazard detection and determination mechanisms together with forwarding and stall mechanisms to handle data risks.

4. RTL design and Implementation:
- implement the pipeline stages in RTL (register transfer level) the usage of a hardware description language inclusive of Verilog or VHDL.
- design and implement the important control logic for each stage and the pipeline interconnections.
- increase modules for the register file, ALU (arithmetic logic Unit), memory interface, and different components as per the RISC-V RV32I specification.

5. Functional Verification:
- develop a complete testbench to confirm the functionality of the pipeline processor.
- Create test cases overlaying a huge range of instructions, addressing modes, and corner instances to ensure the correctness of the implementation.
- perform simulation-based testing using enterprise-standard tools like ModelSim.

6. Overall performance Verification and Optimization:
- perform overall performance analysis to measure the throughput and latency of the pipeline processor.
- identify ability bottlenecks and areas for optimization which includes important paths, resource usage, and instruction dependencies.
- inspect techniques like pipelining optimizations, branch prediction, and instruction-stage parallelism to enhance performance.

7. Synthesis and physical design:
- Synthesize the RTL code to acquire gate-level netlists and use of synthesis tools like the Icarus Verilog compiler.
- Carry out floorplanning and physical synthesis to fulfill timing constraints and achieve the desired clock frequency.
- address issues related to power intake, area, and timing closure through iterative optimization.

8. Publish-Synthesis Verification:
- behavior submit-synthesis simulations to confirm the capability and timing behavior of the synthesized design.
- ensure that the synthesized netlist matches the expected behavior from RTL simulations.

9. RTL-to-GDSII flow (optional for ASIC Implementation):
- If targeting ASIC implementation, continue with the RTL-to-GDSII (Graphical data system) flow.
- Carry out logic synthesis, placement, routing, and timing closure with the use of ASIC design equipment like the Icarus Verilog compiler.

11. Maintenance and further development:
- maintain the design documentation and codebase for future reference and updates.
- address any post-deployment problems, insects, or overall performance upgrades as necessary.
- discover opportunities for extending the processor design with additional functions or custom instructions based on software necessities.

By following this technique, we can systematically develop a RISC-V three-stage pipeline processor compliant with the RV32I instruction set, ensuring functional correctness, overall performance optimization, and scalability for future enhancements.

4.2 Testing OR Verification Plan

In the subsection, the simulation work is mentioned. the principle goal is to examine the designed processor with a RISC-V processor produced utilizing the digital platform tool Imperas OVP to test if the designed processor works efficiently.

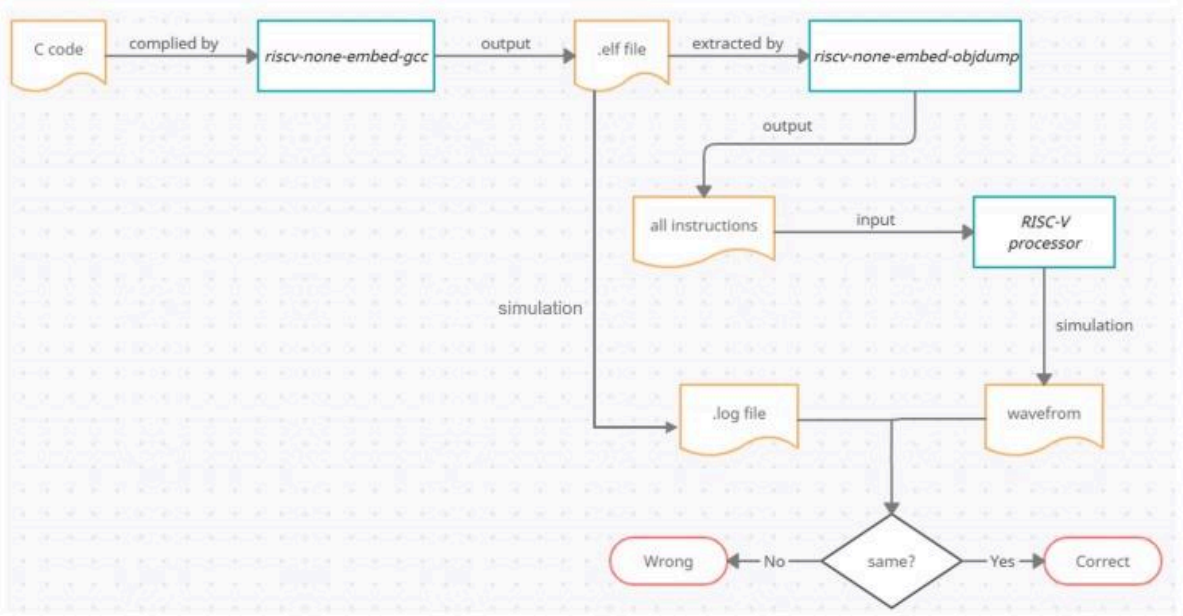

Fig 4.2.1 Block Diagram of Simulation

C codes used for testing the pipeline:

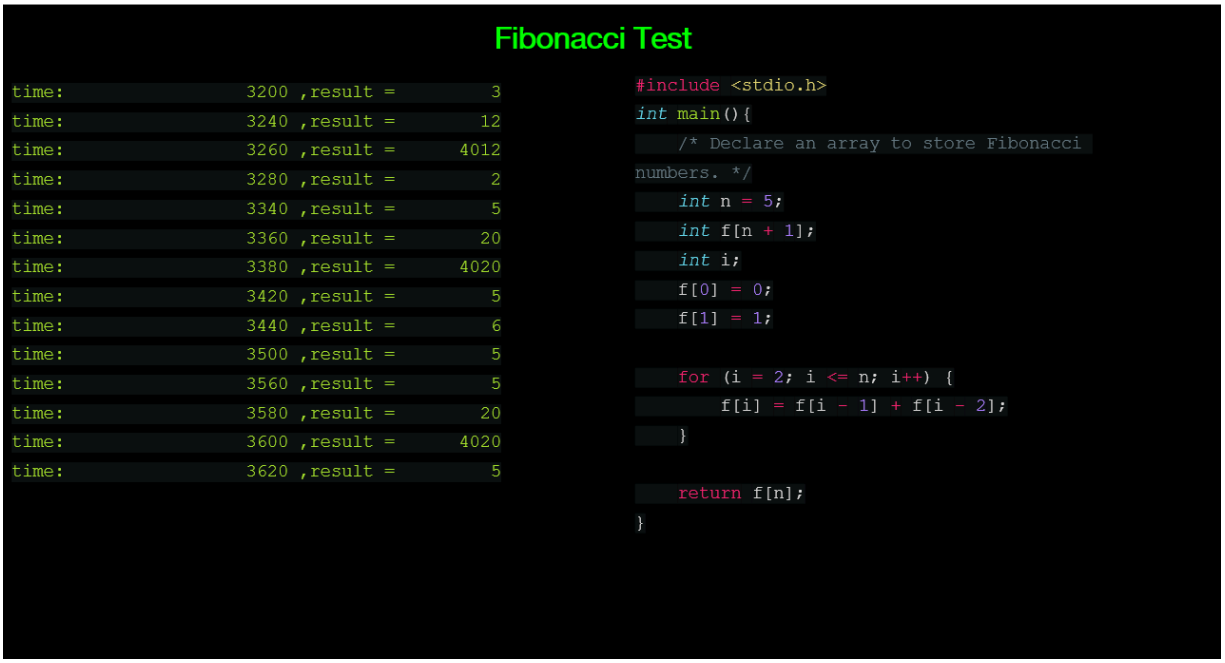

Fig 4.2.2 Fibonacci test

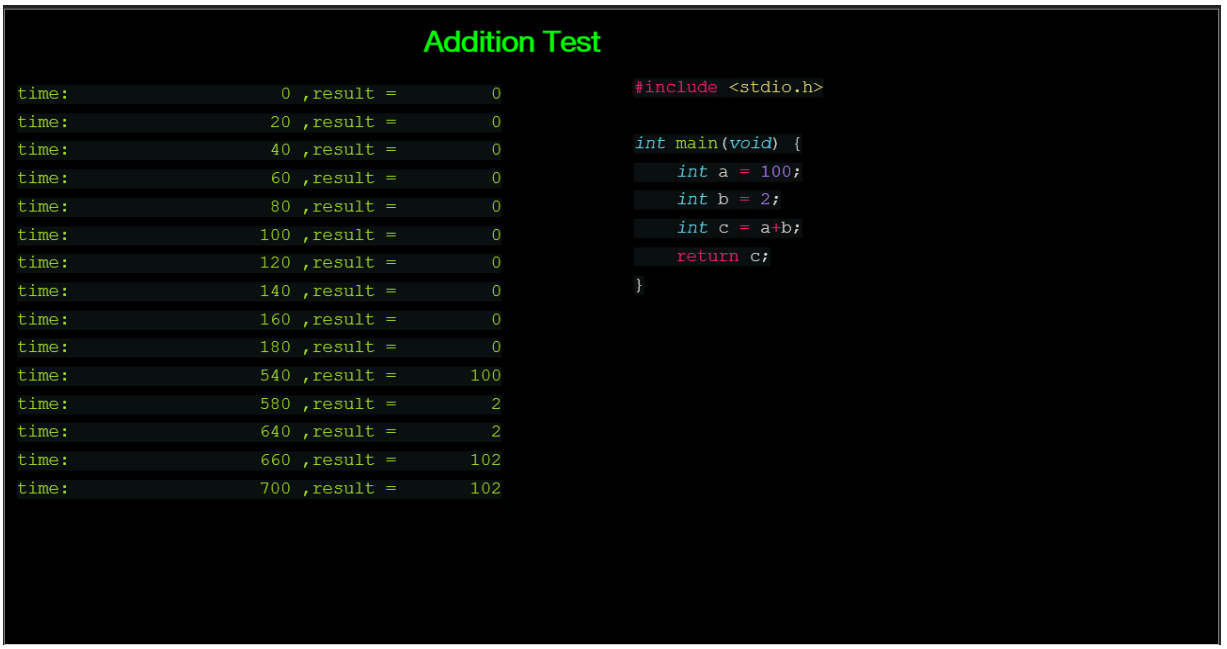

Fig 4.2.3 Addition test

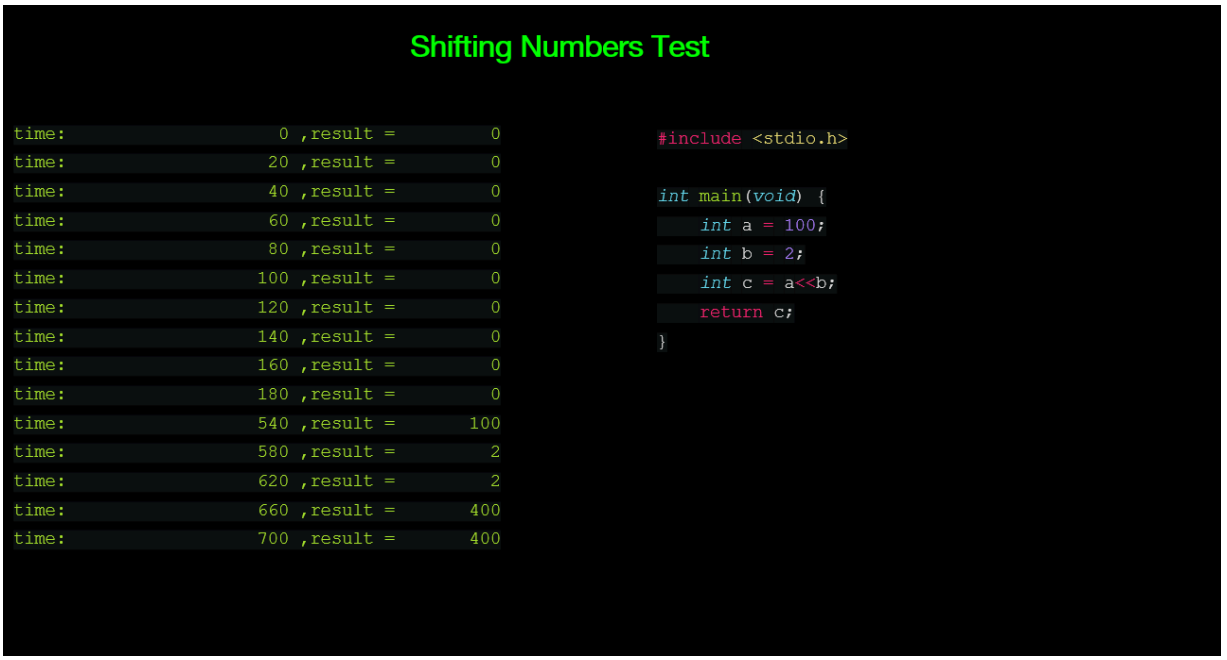

Fig 4.2.4 Shifting Numbers test

```
Sorting Test

time:            2760 ,result =       4031
time:            2800 ,result =          0
time:            2820 ,result =          1
time:            2840 ,result =          4
time:            2880 ,result =       4035
time:            2900 ,result =          6
time:            2980 ,result =          0
time:            3000 ,result =          1
time:            3040 ,result =          2
time:            3060 ,result =          1
time:            3120 ,result =          2
time:            3140 ,result =          1
time:            3200 ,result =          1
time:            3240 ,result =          3
```

```c
#include <stdio.h>
int main(){
int swapped;
int n, i, k, j, m;
n = 3;
int arr[] = {6,3,9};
m =n;
do{
    swapped =0;
    for(j=0; j<n-1; j++){
        if (arr[j]> arr[j+1]){
            int temp;
            temp= arr[j];
            arr[j]= arr[j+1];
            arr[j+1] = temp;
            swapped = 1;
        }
    }
    n = n-1;
}while(swapped==1);
return arr[0];
}
```

Fig 4.2.5 Sorting test

4.3 Result Analysis OR Screenshots

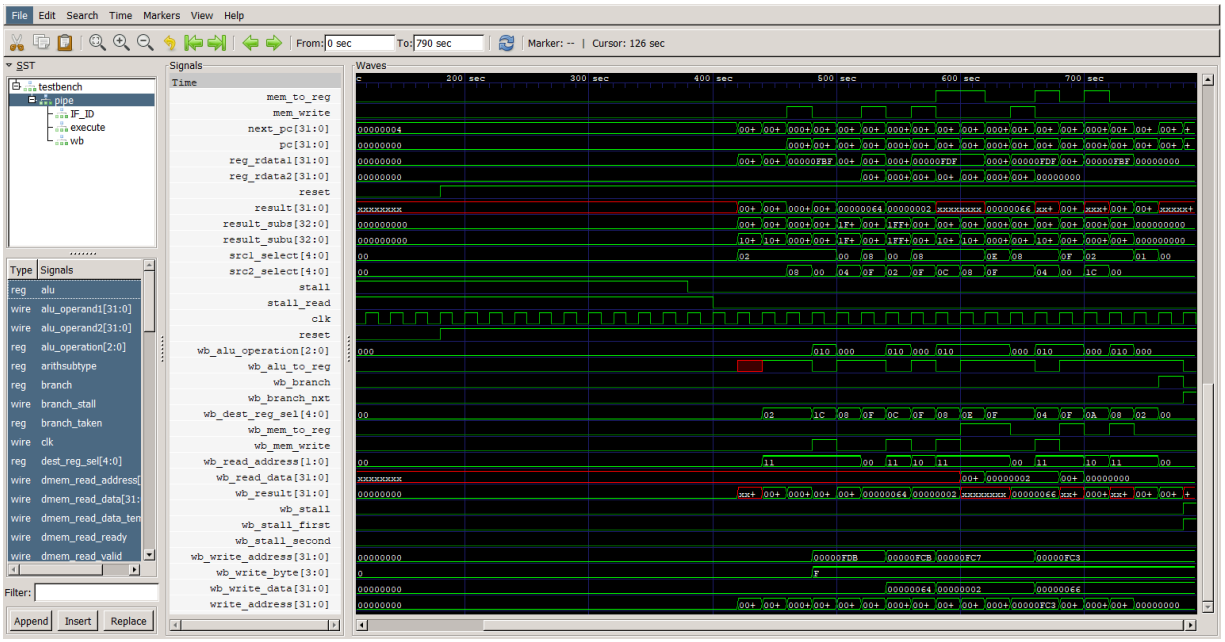

Fig 4.3.1 Waveform for all the wires

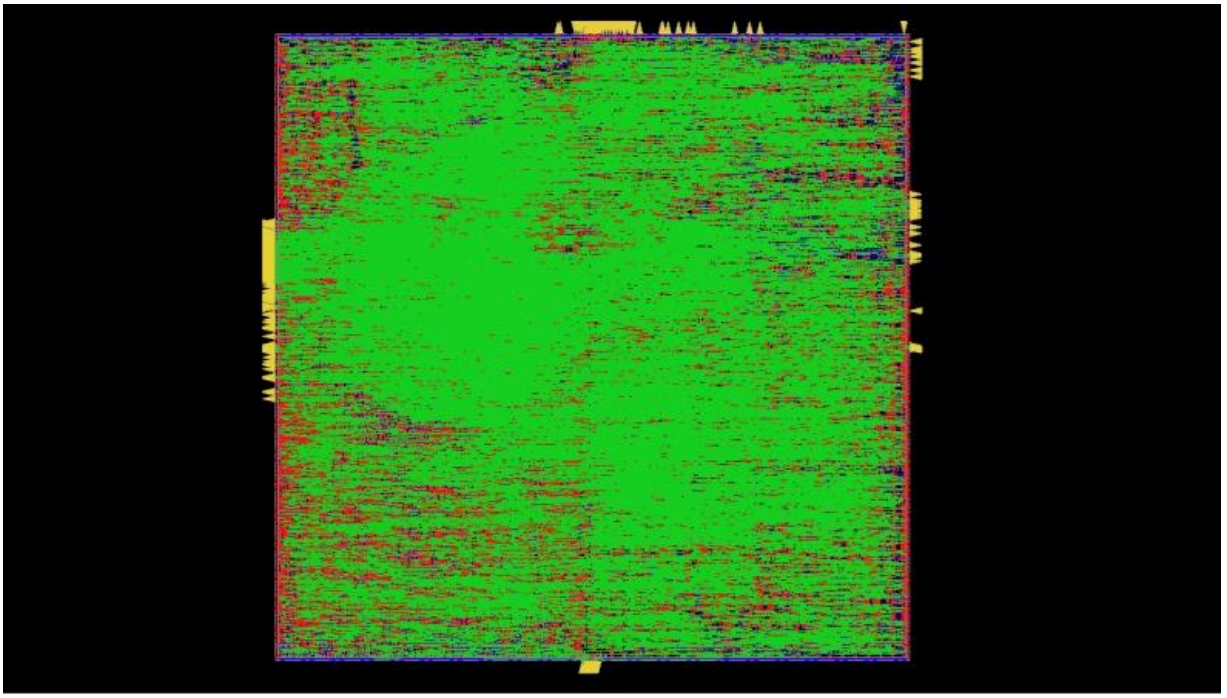

Fig 4.3.2 Final routed layout

The final layout is proven in Fig 4.3.2. The final core usage is quite high. The I/O pins are allotted on every side. A more precise and clean picture is available on Innovus.

```
innovus 1> report_area
    Hinst Name       Module Name          Inst Count          Total Area
-----------------------------------------------------------------------
risccore                                      26384         28308528.000
 i_clint             clint                      672           902016.000
 i_csr_reg           csr_reg                   1797          2351376.000
 i_div               div                       2042          2486160.000
```

Fig 4.3.3 Area report of final routed layout

Fig 4.3.3 indicates the area report of the final format. the entire area is round 28.31 mm2, which is greater than the synthesized one but still less than 49 mm2. this is due to the fact the optimization requires the insertion of additional cells like buffers.

Chapter 5

Standards Adopted

5.1 Design Standards

In the event of developing a RISC-V 3-stage pipeline processor with an RV32I instruction set, it is vital to meet certain design standards and best practices that would ensure that the processor is reliable, efficient, and maintainable. These are some recommended design standards for such a project:

1. RISC-V Specification Compliance: Make sure the design of the processor follows RV32I instruction set specification as outlined by the RISC-V Foundation. Adopt standard instruction format, opcode encodings, as well as behavior prescribed in RISC-V ISA specification.

2. Modularity and Abstraction: Use modular and hierarchical designing strategies when coming up with this processor. Splitting the processor into logical modules such as instruction fetch, instruction decode, execution units, memory access, and write-back stages facilitates abstraction which allows functionality encapsulation inside each module thereby increasing reuse and maintainability.

3. Pipeline Architecture: The architecture should be designed in such a way that three stages including the fetch stage, decode stage, and execute stage should be used for implementing pipelining architecture. Proper handling of pipeline hazards like data hazards, control hazards as well as structural hazards should be done. Ensure that some stall mechanisms can help to maximize pipeline efficiency through forwarding units and branch prediction methods being incorporated.
4. Synchronization of Time by a Clock and Timing Aspects: This is where you establish rules that define how time synchronization should be done and force the use of timing constraints between different parts of pipelines. Such constraints are put in place to meet performance standards as well as ensure dependable operation within the range of clock rates specified.

5. Designing Data Path and Control Logic: It involves designing bits for RV32I instructions that can be executed efficiently including arithmetic operations, logical operations, memory accesses, and control flow instructions. At this stage, we build up control logic which can interpret instruction opcodes, and generate control signals to enable communication between pipeline stages.

6. The Cache Design with Memory Hierarchy: We could improve our system by introducing cache memory into the memory hierarchy to reduce latency during memory access. About addressing translation, data caching, and protection against memory-related problems, such things as cache controllers, Cache Coherence Protocols (CCPs), and Memory Management Units (MMUs).

7. Error Handling and Fault Tolerance: Incorporate mechanisms that ensure error detection and error correction into the processor design, which will guarantee its reliability and fault tolerance. Also include memory error detection codes, parity checks, and Error Correction Codes (ECC) in the data path.

8. Testing and Verification: These test plans reveal most of the processor design's functionality, performance, and correctness to avoid any mistakes while all these could be confirmed through simulation runs, emulations, or formal verification techniques as it makes certain whether the architectural designs are aligned with RISC-V's RV32I ISA spec.

9. Documentation and Reporting: Keep comprehensive documentation about the processor architecture, design specifications, implementation details, testing methodologies as well as verification results for future reference purposes such that reports that present a clear report on design reasoning good enough to convince stakeholders regarding its suitability towards the intended application scenarios can easily be created

10. Compliance with Industry Standards: This is also inclusive of IEEE standards regarding hardware description languages like VHDL/Verilog; ISO about quality management systems; and RISC-V foundation among others which regulate how computer chips should be configured because technology keeps changing so developers must keep pace with new industry trends or they will soon become obsolete.
By following these design standards and excellent practices, you may develop a RISC-V three-stage pipeline processor to build the use of the RV32I instruction set that meets performance, reliability, and scalability requirements while adhering to industry standards and specifications.

5.2 Coding Standards

Coding standards for RISC-V three-stage Pipeline Processor (RV32I):
general principles:
minimize lines: Prioritize concise and efficient code.
meaningful Names: Use descriptive variables and signal names.
Logical Grouping: organize code blocks into well-defined sections with clear comments.
Indentation: Indent consistently to enhance code readability for control flow structures (if/else, loops).

function design:
single obligation: each function has to handle a particular task.
Modularity: break down complex functionalities into smaller, reusable functions.
limited size: strive for functions that fit easily within a single screen.

specific considerations:
Pipeline stages: outline separate functions for each pipeline stage (Fetch, Decode, Execute).
instruction decoding: Use a case declaration or comparable construct for efficient instruction decoding primarily based on opcode.
data Forwarding: implement data forwarding logic inside the Execute stage using committed functions or combinational logic.
Hazard detection: employ functions to identify data hazards (raw) and control hazards (branches) inside the Decode stage.
Stalls and Bubbles: utilize separate functions or logic for handling pipeline stalls and bubble insertion.

Additional recommendations:
comments: include concise comments to explain non-obvious code sections or design selections.
register Naming: Adhere to RISC-V register naming conventions (e.g., rd, rs1, rs2).
signal Naming: Use clear and descriptive names for signals inside the datapath (e.g., pc_next, reg_data_out).

By following those guidelines, we will ensure a nicely structured, easy-to-understand, and maintainable codebase for our RISC-V three-stage pipeline processor.

5.3 Testing Standards

Testing standards for RISC-V 3-stage Pipeline Processor (RV32I)

Even as there isn't a single standard specifically for testing RISC-V processors, we will leverage installed practices and resources for effective verification:

1. Functional Verification:
Random instruction era: Use a random instruction generator to feed the processor with a diffusion of instructions, testing its ability to handle various situations.
corner Case testing: design test cases focused on particular corner cases, along with boundary conditions for arithmetic operations or memory access styles.

2. Pipeline Verification:
Pipeline Stalls and Bubbles: enforce tests that create data hazards (raw) and control hazards (branches) to confirm the processor correctly handles stalls and inserts bubbles while important.
data Forwarding: Create test instances that exercising the data forwarding logic in the Execute stage. make sure correct forwarding of results from preceding stages to avoid useless stalls.

4. Register file testing:
read/Write Operations: Write test cases that carry out numerous read and write operations to all registers. verify the processor effectively reads and writes data to the particular registers.
Exception handling: test the behavior of the processor whilst encountering exceptions, which includes illegal commands or memory access violations.

5. Simulation and Verification tools:
RTL Simulation: make use of hardware verification tools like Verilator or Icarus Verilog to simulate the processor's register-switch degree (RTL) code. This lets in for detailed inspection of internal signals and verification of data flow.
instruction Set Simulator (ISS): consider the use of an RISC-V ISS (coaching Set Simulator) to run a wider variety of test programs and confirm processor conduct at a better stage of abstraction.

6. Documentation and Reporting:
test Plan: develop a complete test plan outlining the testing methodologies and specific test cases to be accomplished.

test coverage report: document the carried out test coverage, indicating the proportion of commands and functionalities exercised through the test suite.
bug tracking: implement a bug monitoring system to report any problems encountered in the course of testing and track their resolution.

By following those testing standards, we can make sure an intensive verification of our RISC-V three-level pipeline processor, promoting confidence in its capability and reliability.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

In conclusion, the improvement of the "RISC-V three-stage pipeline processor build using RV32I instruction set" project marks a considerable achievement within the realm of processor design and implementation. Leveraging Verilog coding for implementation, alongside simulation tools like ModelSim and compilation using Icarus Verilog, the project correctly delivered to life a 32-bit core with a three-stage pipeline, supporting the RV32I base instruction set with the "M" standard extension.

During the task, several changes were made to pipeline devices, substantially in the division unit, to ensure assistance for all divisions and the rest commands. additionally, the incorporation of new modules, inclusive of CSR_REG and INTu, increased the functionality of the processor. despite some non-standard strategies, like having ROM as part of the core and limitations in assisting multiple slaves, the processor core demonstrated predicted functionality throughout verification and simulation.

Upon synthesis as an ASIC using the SoI_STD library, the RISC-V core exhibited an area of about 25.177 mm², with a slack of approximately 0.5ns. even though the vital data path allowed for a clock frequency of up to 10MHz, post-region and route estimations indicated an area of around 28.31 mm².

Despite those achievements, it's crucial to observe that the core is not presently available for manufacturing purposes. future endeavors may additionally contain refining the design, addressing non-standard strategies, and improving aid for additional functionalities and scalability. overall, the challenge showcases a complete understanding of processor structure and design principles, laying a stable foundation for further exploration and development within the area.

6.2 Future Scope

The future scope for the "RISC-V 3-stage pipeline processor build using RV32I instruction set" project is sizeable, encompassing several essential areas for enhancement. at the RTL stage, the integration of additional modules like a JTAG debug module and peripherals which include UART or timers can bolster the processor's functionality. Optimizing algorithms, growing pipeline stages, and mitigating pipeline bubbles at some stage in specific instructions are imperative for enhancing efficiency. furthermore, addressing missing components like I/O pads and RAM layouts, optimizing the SoI_STD library, and increasing the bus unit to support greater than four slaves are vital for overall system enhancement. Embracing standard bus designs just like the AXI4 protocol and enforcing strategies like department predictors and speculative execution can similarly optimize performance and compatibility. By focusing on those areas, the project can evolve into a more versatile and efficient RISC-V processor design, poised to satisfy the needs of numerous computing applications.

References

[1] Mezger, B.W., Santos, D.A., Dilillo, L., Zeferino, C.A., & Melo, D.R. (2022). A Survey of the RISC-V Architecture Software Support. *IEEE Access, 10*, 51394-51411.

[2] Zhang, W., Zhang, Y., & Zhao, K. (2021). Design and Verification of Three-stage Pipeline CPU Based on RISC-V Architecture. *2021 5th Asian Conference on Artificial Intelligence Technology (ACAIT)*, 697-703.

[3] Aletan, S.O. (1992). An overview of RISC architecture. *ACM Symposium on Applied Computing*.

[4] Takai, Y., Nagase, M., Kitamura, M., Koshikawa, Y., Yoshida, N., Kobayashi, Y., Obara, T., Fukuzo, Y., & Watanabe, H. (1993). 250 Mbyte/sec synchronous DRAM using a 3-stage-pipelined architecture. *Symposium 1993 on VLSI Circuits*, 59-60.

[5] Gokhale, M.B., & Stone, J.M. (1998). NAPA C: compiling for a hybrid RISC/FPGA architecture. *Proceedings. IEEE Symposium on FPGAs for Custom Computing Machines (Cat. No.98TB100251)*, 126-135.

[6] Li, Z., Hu, W., & Chen, S. (2019). Design and Implementation of CNN Custom Processor Based on RISC-V Architecture. *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 1945-1950.

[7] Poli, L., Saha, S., Zhai, X., & Mcdonald-Maier, K.D. (2021). Design and Implementation of a RISC V Processor on FPGA. *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, 161-166.

[8] Palmiero, C., Guglielmo, G.D., Lavagno, L., & Carloni, L.P. (2018). Design and Implementation of a Dynamic Information Flow Tracking Architecture to Secure a RISC-V Core for IoT Applications. *2018 IEEE High Performance extreme Computing Conference (HPEC)*, 1-7.

INDIVIDUAL CONTRIBUTION REPORT:
RISC-V 3-Stage Pipeline Processor build using RV32I Instruction Set

ANIRUDDH SINGH
2105769

Abstract: This project gives the design and development of an open-source RISC-V 3-stage pipeline processor primarily based on the RV32I instruction set architecture. the usage of SystemVerilog for RTL coding and ModelSim for simulation, the processor adheres to RISC standards with a focus on simplicity, modularity, and extensibility. Benchmarking with RV32IM instruction set and industry-standard strategies exhibit high performance, efficiency, and adherence to RISC-V's goals, contributing to the advancement of RISC-V-based total processors in various computing applications.

Individual contribution and findings: As Aniruddh Singh, my main role in the project centered on crafting and executing the Fetch Stage of the RISC-V 3-Stage Pipeline Processor, utilizing the RV32I Instruction Set. This entailed crafting the interface for instruction memory, programming PC increment logic, and ensuring accurate handling of control signals for branching and jumping instructions. I meticulously planned each phase, commencing with a comprehensive grasp of the instruction set architecture and pipeline design principles. I devised the interface meticulously, considering data width, address structure, and timing nuances, and implemented PC increment logic while accommodating diverse execution scenarios. Throughout the process, I encountered hurdles such as ensuring synchronous operation among pipeline stages and optimizing memory access techniques. Nonetheless, through diligent testing and Verilog experimentation, I successfully tackled these challenges, yielding valuable insights into processor architecture and digital design principles. Overall, my pivotal role in establishing the foundation for the Fetch Stage significantly contributed to the project's overall success.

Individual contribution to project report preparation: Prepared the Chapter 3 Problem Statement / Requirement Specifications and chapter 6 Future Scope.

Individual contribution for project presentation and demonstration:
During the project presentation and demonstration, I focused on showcasing the Fetch Stage, which I developed as part of the project team. I led the explanation of the Fetch Stage's design and functionality, emphasizing its pivotal role in the RISC-V 3-Stage Pipeline Processor. I effectively communicated the complexities of the instruction memory interface, PC increment logic, and the management of control signals for branching and jumping instructions. Throughout the demonstration, I provided a thorough walkthrough of how the Fetch Stage operates within the pipeline, highlighting its importance in fetching instructions and initializing their execution. Furthermore, I addressed any queries raised by the audience regarding the Fetch Stage implementation, ensuring a comprehensive understanding of its functionality.

In summary, my individual contribution during the project presentation and demonstration centered on presenting and elucidating the Fetch Stage, which I developed as part of the project team.

Full Signature of Supervisor:

Full signature of the student
Aniruddh Singh

RISC-V 3-Stage Pipeline Processor build using RV32I Instruction Set

KHUSHI SINGH
2105803

Abstract: This project gives the design and development of an open-source RISC-V 3-stage pipeline processor primarily based on the RV32I instruction set architecture. the usage of SystemVerilog for RTL coding and ModelSim for simulation, the processor adheres to RISC standards with a focus on simplicity, modularity, and extensibility. Benchmarking with RV32IM instruction set and industry-standard strategies exhibit high performance, efficiency, and adherence to RISC-V's goals, contributing to the advancement of RISC-V-based total processors in various computing applications.

Individual contribution and findings: In our project aimed at constructing a RISC-V 3-Stage Pipeline Processor using the RV32I Instruction Set, I assumed the pivotal role of designing and implementing the Decode Stage. This task necessitated a thorough grasp of the RISC-V instruction set architecture, particularly RV32I, and meticulous attention to detail during the decoding process. My responsibilities included accurately deciphering fetched instructions, extracting crucial fields like opcode, source/destination registers, and immediate values, and generating essential control signals for subsequent pipeline stages. Collaborating closely with my team members, I ensured seamless integration with both the Fetch and Execute stages. While executing this phase, challenges arose in managing intricate instruction formats and ensuring compatibility with the overall pipeline design. However, through collaborative problem-solving efforts, we effectively surmounted these obstacles. My involvement significantly deepened my understanding of processor architecture and pipeline design principles, instilling confidence in the successful culmination of our project..

Individual contribution to project report preparation: Prepared the Chapter 4 Implementation.

Individual contribution for project presentation and demonstration: During the project presentation and demonstration of the RISC-V 3-Stage Pipeline Processor with the RV32I Instruction Set, I played a significant role due to my involvement in the Design and Implementation of the Decode Stage. In the presentation, I focused on elucidating the Decode Stage's importance within the pipeline's overall functioning. I described how this stage deciphers fetched instructions, emphasizing the process of extracting critical fields like opcode, source/destination registers, and immediate values. Moreover, I showcased how we generate vital control signals based on the instruction types, illustrating how this stage enables smooth instruction execution in subsequent pipeline phases. Additionally, I shared insights into the challenges we faced during implementation and detailed our collaborative problem-solving approach to overcome them.

Through clear explanations, supported by visual aids and potential live demonstrations, I effectively conveyed the intricacies of the Decode Stage, ensuring a thorough understanding of its role in the project's success..

Full Signature of Supervisor:

Full signature of the student:
Khushi Singh

RISC-V 3-Stage Pipeline Processor build using RV32I Instruction Set

SADAF SHAHAB
21051590

Abstract: This project gives the design and development of an open-source RISC-V 3-stage pipeline processor primarily based on the RV32I instruction set architecture. the usage of SystemVerilog for RTL coding and ModelSim for simulation, the processor adheres to RISC standards with a focus on simplicity, modularity, and extensibility. Benchmarking with RV32IM instruction set and industry-standard strategies exhibit high performance, efficiency, and adherence to RISC-V's goals, contributing to the advancement of RISC-V-based total processors in various computing applications.

Individual contribution and findings: In our project to construct a RISC-V 3-Stage Pipeline Processor utilizing the RV32I Instruction Set, I played a critical role in designing and implementing the Execute Stage. This phase is essential as it carries out arithmetic, logical, and data transfer operations based on previously decoded instructions. My planning involved thorough preparation, reviewing the RV32I instruction set and Execute Stage specifications, and collaborating closely with team members to outline necessary components and functionalities. During implementation, I focused on developing efficient algorithms for executing various instruction types, such as implementing the Arithmetic Logic Unit (ALU), managing data dependencies, handling branch instructions, and calculating memory addresses for load/store operations. Challenges included optimizing branch instruction execution and ensuring accurate memory address calculations. Rigorous testing and collaboration with team members ensured seamless integration with other pipeline stages, leading to a functional RISC-V processor.

Individual contribution to project report preparation: Prepared the Chapter 2 2 Basic Concepts/ Literature Review.

Individual contribution for project presentation and demonstration: In the project presentation and demonstration of the RISC-V 3-Stage Pipeline Processor built using the RV32I Instruction Set, my primary contribution stemmed from my role in the Design and Implementation of the Execute Stage. This stage plays a pivotal role as it executes arithmetic, logical, and data transfer operations based on the instructions decoded in earlier stages. My responsibilities encompassed implementing crucial functionalities such as Arithmetic Logic Unit (ALU) operations, managing data dependencies, executing branch instructions, and calculating memory addresses for load/store operations. During the presentation, I effectively communicated the significance of the Execute Stage in facilitating the core processing tasks of the processor. I demonstrated the ALU operations, showcasing the execution of arithmetic and logical instructions, and highlighted the efficient handling of data dependencies to ensure smooth operation and avoid hazards. Additionally, I explained the execution of branch instructions and the importance of accurate memory address calculations for load/store operations.

Through clear explanations and possibly live demonstrations, I elucidated the integral role of the Execute Stage in the overall functionality and performance of the RISC-V proces

Full Signature of Supervisor:

Full signature of the student:
Sadaf Shahab

RISC-V 3-Stage Pipeline Processor build using RV32I Instruction Set

MEHUL AGARWAL
21051479

Abstract: This project gives the design and development of an open-source RISC-V 3-stage pipeline processor primarily based on the RV32I instruction set architecture. the usage of SystemVerilog for RTL coding and ModelSim for simulation, the processor adheres to RISC standards with a focus on simplicity, modularity, and extensibility. Benchmarking with RV32IM instruction set and industry-standard strategies exhibit high performance, efficiency, and adherence to RISC-V's goals, contributing to the advancement of RISC-V-based total processors in various computing applications.

Individual contribution and findings:.In our project to construct a RISC-V 3-Stage Pipeline Processor utilizing the RV32I Instruction Set, I assumed the role of designing and implementing the Memory Stage. This phase involved accessing memory for load/store instructions and managing data hazards. My planning encompassed understanding memory access operations and potential hazards, collaborating closely with team members to ensure seamless integration. During implementation, I focused on designing the data memory interface, handling data forwarding from the Execute Stage, and executing memory accesses for load/store instructions. Despite encountering technical challenges, particularly in mitigating data hazards, collaborative problem-solving and rigorous testing facilitated successful resolution. This experience deepened my understanding of memory management in pipelined processors and emphasized the importance of efficient hazard resolution.

Individual contribution to project report preparation: Prepared the Chapter 5 Standard Adopted.

Individual contribution for project presentation and demonstration: In the context of our project on the development of a RISC-V 3-Stage Pipeline Processor utilizing the RV32I Instruction Set, my assigned task pertained to the Design and Implementation of the Memory Stage. This phase encompassed crucial responsibilities such as facilitating memory access for load/store instructions, managing data hazards, and ensuring smooth data forwarding from the execute stage. Specifically, my contributions involved designing and implementing the data memory interface, executing memory accesses for load/store instructions, and proactively addressing any memory-related hazards encountered during the development process. As we prepare for the project presentation and demonstration, my role will be to effectively showcase the functionality and efficiency of the Memory Stage. This entails demonstrating seamless memory access, efficient data handling, and robust hazard resolution mechanisms. Additionally, during the presentation, I will provide detailed insights into the design choices, implementation strategies, and performance optimizations employed in the Memory Stage.

Through clear and concise communication, I aim to elucidate the significance of this stage in the overall functionality and performance of our RISC-V processor..

Full Signature of Supervisor:

Full signature of the student:
Mehul Agarwal

RISC-V 3-Stage Pipeline Processor build using RV32I Instruction Set

RADHIKA MANISH

2105730

Abstract: This project gives the design and development of an open-source RISC-V 3-stage pipeline processor primarily based on the RV32I instruction set architecture. the usage of SystemVerilog for RTL coding and ModelSim for simulation, the processor adheres to RISC standards with a focus on simplicity, modularity, and extensibility. Benchmarking with RV32IM instruction set and industry-standard strategies exhibit high performance, efficiency, and adherence to RISC-V's goals, contributing to the advancement of RISC-V-based total processors in various computing applications.

Individual contribution and findings:In the project of building a RISC-V 3-Stage Pipeline Processor using the RV32I Instruction Set, my role primarily centered around Integration, Testing, and Optimization. Throughout this phase, I was responsible for the seamless integration of individual stage modules, ensuring their interoperability and coherence within the overall pipeline structure. I meticulously conducted unit tests for each stage, meticulously analyzing their functionality and identifying any discrepancies. Furthermore, I orchestrated comprehensive functional and performance tests on the complete pipeline to validate its correctness and efficiency. My efforts were also directed towards optimizing the design to enhance throughput and efficiency, employing various strategies such as pipeline stall reduction and instruction scheduling. Through diligent planning and collaboration with other team members, I ensured that the integrated processor met the project objectives effectively.

Individual contribution to project report preparation: Prepared the Chapter 1 Introduction and chapter 6 conclusion.

Individual contribution for project presentation and demonstration: For the project presentation and demonstration, my primary involvement centers around the Integration, Testing, and Optimization phase of crafting the RISC-V 3-Stage Pipeline Processor utilizing the RV32I Instruction Set. In this role, I'll exhibit the seamless integration of individual stage modules, highlighting their functionalities and interactions within the pipeline framework. During the presentation, I'll demonstrate the meticulous unit testing conducted for each stage, emphasizing the thoroughness in ensuring accuracy and efficiency. Additionally, I'll showcase the comprehensive functional and performance tests carried out on the complete pipeline, illustrating its effectiveness and adherence to project specifications. Furthermore, I'll discuss the optimization techniques utilized to boost throughput and efficiency, offering insights into the design refinements made during the project's progression. Regarding the demonstration, I'll guide through the operation of the integrated pipeline, showcasing its capability to execute instructions precisely and effectively. I'll also underscore any optimizations applied, illustrating their impact on key performance metrics like throughput and latency.

Overall, my role in the presentation and demonstration will focus on effectively communicating the accomplishments of the Integration, Testing, and Optimization phase, while showcasing the functionality and performance of the developed pipeline processor.

Full Signature of Supervisor:

Full signature of the student:
Radhika Manish

TURNITIN PLAGIARISM REPORT