रा.इ.सू.प्रौ.सं
NIELIT
CHENNAI

NATIONAL INSTITUTE OF ELECTRONICS AND INFORMATION TECHNOLOGY, CHENNAI

ONLINE INTERNSHIP IN RTL DESIGN USING VERILOG HDL

PROJECT REPORT

on

5-STAGE PIPELINED RV32I RISC-V PROCESSOR

SUBMITTED BY

VIDYADHEESHA M PANDURANGI

at

FEBRUARY 2026

ABOUT NIELIT

National Institute of Electronics and Information Technology (NIELIT), headquartered in New Delhi is the capacity building arm of the Ministry of Electronics and Information Technology (MeitY), mandated to carry out HR development and related activities in Information, Electronics and Communication Technology (IECT).

Since its inception in 2010, NIELIT Chennai has established itself as a premier institution providing affordable quality education as per the job market requirements for candidates primarily from Tamil Nadu. As of date NIELIT Chennai has imparted skill training to more than 30, 000 youth of this region and conducted digital literacy certification for more than 25,000 candidates. NIELIT Chennai handles the activities of the NIELIT in Tamil Nadu, Andhra, Telangana, and the Andaman and Nicobar Islands.

Presently NIELIT Chennai is implementing the following projects for Tamil Nadu: ISEA (Information Security Education and Awareness), Future Skills PRIME capacity building projects (3D Printing and Additive Manufacturing, Robotic Process Automation, IoT - technology streams), and the aspirational district projects for SC/ST candidates at Ramanathapuram and Virudhunagar funded by MeitY, Government of India. The centre also received funding through the electronics product design and product testing capacity building project funded by MeitY.

Other than training, the centre also provides services like; Private Cloud Setup using Citrix, vSphere, Red Hat & FOSS Cloud, Hyper-converged Infrastructure (HCI) and Virtual Desktop Infrastructure (VDI) Server Setup, High-Performance Computing (HPC) Setup with 100G/200G connectivity, Server and Virtual Lab Setup for BigData, AI & Information Security, and Digital & Mobile Forensics Service.

NIELIT has also set up a Virtual Academy, a common End-to-End Platform to conduct Online training courses including a virtual lab environment in the areas of IECT from foundation to expert level targeting from school students to working professionals.

INDEX

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who have supported and guided me throughout the completion of this project on **5-STAGE PIPELINED RV32I RISC-V PROCESSOR** as part of the **Online Internship in RTL Design using Verilog HDL**, conducted by the National Institute of Technology and Applied Research (NIELIT), Chennai.

First and foremost, I would like to thank my course instructor, Mr. Naresh Raja, Project Engineering - VLSI for his invaluable guidance, constructive feedback, and constant encouragement throughout the project. His expertise in the field of Chip Design has been a key factor in the successful completion of this report.

I would also like to extend my appreciation to the NIELIT Chennai for offering this comprehensive online course, providing us with high-quality learning resources and support despite the challenges of remote learning.

A special thanks to my fellow participants in the online internship for the engaging discussions and knowledge-sharing that enriched my learning experience. The collaborative nature of this internship has significantly enhanced the depth and quality of my work.

Finally, I would like to thank all the authors, researchers, and resources referenced in this report for their contributions to the field of Chip Design, which formed the foundation of my project.

Thank you to everyone who has been part of this learning journey.

ABSTRACT

With the rapid advancement of high-performance embedded computing systems, modern processors are required to deliver faster execution while ensuring efficient utilization of hardware resources. One of the most prevalent strategies to enhance processor throughput is pipelining, a technique that enables multiple instructions to execute concurrently across different stages of the processor's datapath.

This project focuses on designing and implementing a 5-stage pipelined RV32I RISC processor based on the RISC-V instruction set architecture using the Verilog Hardware Description Language (HDL). It was undertaken as part of an internship program offered by NIELIT, aimed at providing hands-on experience with the full VLSI design process, from behavioral modeling to FPGA-level implementation.

The proposed processor architecture follows the classic five pipeline stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB). To facilitate correct execution in a pipelined environment, the design includes critical control features such as pipeline registers, a hazard detection unit, and a data forwarding unit. These components effectively handle data hazards and optimize instruction flow for improved overall efficiency.

The processor was rigorously tested and verified using the Vivado Simulator. Behavioral simulations confirmed proper execution of instructions and accurate register write-back functionality. Additionally, the design was successfully synthesized and implemented on an FPGA target device. This stage involved generating utilization reports, RTL schematics, timing analyses, and post-implementation simulation results. Timing analysis indicated that all specified user constraints were met with positive slack, ensuring stable operation.

In conclusion, this project highlights the successful development of a functional pipelined RISC-V processor core. It provides significant learning opportunities in processor design principles, Verilog-based implementation, techniques for hazard resolution, and the end-to-end FPGA design process.

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The relentless advancement of modern electronic systems has generated a significant demand for digital processing units that are faster, more reliable, and energy-efficient. With the exponential growth of embedded devices, Internet of Things (IoT) applications, industrial automation, real-time signal processing, and intelligent computing platforms, processors have become an indispensable component at the core of almost every electronic system. Acting as the "brain" of these systems, a processor handles tasks such as executing instructions, processing data, controlling peripheral operations, and ensuring the seamless functionality of the entire digital infrastructure.

In the early days of computing, processor designs operated on sequential execution models. Under this framework, each instruction was fetched, decoded, executed, and completed before the subsequent instruction began. While straightforward to implement, this method underutilized available processor hardware resources and significantly limited execution speed. To address these limitations, contemporary processor architectures employ advanced performance optimization techniques like pipelining, parallel execution, caching, and hazard management.

A popular architectural paradigm in processor design today is the Reduced Instruction Set Computer (RISC) philosophy. RISC processors rely on a streamlined instruction set with fixed instruction formats and finely-tuned execution stages, making them particularly well-suited for pipelined implementations. Compared to more complex instruction set processors, RISC-based designs deliver faster execution speeds, simpler control logic, and better scalability for today's VLSI systems.

In recent years, RISC-V has emerged as a groundbreaking open-source Instruction Set Architecture (ISA), capturing significant attention across academic and industry circles. Unlike proprietary ISAs, RISC-V offers an open, extensible, and royalty-free platform that allows researchers and engineers to create customized processors tailored to diverse applications ranging from embedded controllers to high-performance computing environments.

The RV32I instruction set constitutes the 32-bit foundational integer instruction set for RISC-V. It encompasses core operations such as arithmetic computations, immediate operations, load/store memory instructions, and conditional branching. Developing an RV32I-compatible processor provides a valuable opportunity to explore essential CPU architecture elements such as datapath design, instruction decoding, register file operations, arithmetic logic unit (ALU) management, memory interfacing, and pipelined execution.

This particular project was initiated during an internship program at NIELIT with the goal of designing and implementing a 5-stage pipelined RISC-V processor using Verilog Hardware Description Language (HDL) within the AMD Xilinx Vivado Design Suite. The processor adheres to the conventional five pipelines stages:

- **Instruction Fetch (IF)** - Retrieving instructions from instruction memory
- **Instruction Decode (ID)** - Decoding instructions and fetching registers
- **Execute (EX)** - Performing logical and arithmetic computations via the ALU
- **Memory Access (MEM)** - Loading or writing data for memory-related instructions
- **Write Back (WB)** - Storing computed results into registers

To enable functional pipelined processing while minimizing delays, pipeline registers such as IF/ID, ID/EX, EX/MEM, and MEM/WB were incorporated between stages. Furthermore, the design includes pipeline control mechanisms such as a hazard detection unit to handle load-use hazards and a forwarding unit for resolving data dependencies without excessive stalling.

The processor design underwent rigorous testing and verification through a comprehensive FPGA design workflow. Behavioral simulations confirmed its functional correctness, followed by synthesis to generate gate-level logic. It was further validated through post-synthesis and post-implementation simulations. Static timing analysis ensured compliance with timing constraints, while FPGA resource utilization reports offered insights into hardware efficiency.

This project offered in-depth practical experience in contemporary processor design methodologies, pipelining strategies, hazard mitigation techniques, and full FPGA-based VLSI application development. The knowledge and skills gained serve as a robust foundation for further exploration into advanced computer architecture, digital VLSI design principles, and embedded processor systems development.

CHAPTER 2
LITERATURE

The design and implementation of pipelined processors has been a fundamental area of research and development in computer architecture, particularly for achieving higher instruction throughput and improved performance. Modern embedded and high-performance computing systems widely rely on efficient processor architectures that support parallel instruction execution and optimized datapath structures.

The RISC (Reduced Instruction Set Computer) approach has proven to be highly effective for pipelined processor implementations due to its simplified instruction formats and uniform execution stages. Patterson and Hennessy emphasize that RISC-based architectures enable faster decoding, reduced control complexity, and efficient pipelining when compared to complex instruction set designs [2]. Their work provides a strong foundation for understanding the classical five-stage pipeline model consisting of Instruction Fetch, Decode, Execute, Memory Access, and Write Back stages.

In recent years, the emergence of the RISC-V Instruction Set Architecture (ISA) has introduced a new open-standard platform for processor development. The RISC-V ISA is extensible, royalty-free, and widely adopted in both academic and industrial applications. The official RISC-V specification defines the RV32I base integer instruction set, which forms the basis for designing 32-bit processors with arithmetic, load/store, and branch instructions [1], [5]. This openness has encouraged the implementation of custom RISC-V cores for FPGA-based and ASIC-based designs.

Pipelining improves processor throughput by overlapping multiple instructions across different stages of execution. However, pipelined execution introduces hazards such as data dependencies and control hazards. Patterson and Hennessy discuss that hazard resolution techniques like forwarding and stalling are essential for maintaining correctness in pipelined datapaths [3]. Forwarding units allow dependent instructions to receive results directly from intermediate pipeline stages, while hazard detection units introduce stalls during load-use situations.

Hennessy and Patterson further highlight that modern computer architecture design continues to evolve toward modular and scalable processor implementations, where open ISAs such as RISC-V play a significant role in future computing systems [4]. This supports the growing importance of implementing pipelined RISC-V processors in academic projects and industry training programs.

Hardware description languages such as Verilog are widely used for implementing processor datapaths and control logic. Palnitkar provides comprehensive insights into synthesizable Verilog constructs, modular design practices, and simulation methodologies, which are essential for building complex digital systems such as pipelined CPUs [9].

FPGA-based implementation tools such as the AMD Xilinx Vivado Design Suite enable complete design verification through behavioral simulation, synthesis, and post-implementation timing analysis. The Vivado user guides describe the importance of functional simulation, gate-level verification, static timing analysis, in ensuring that the design meets hardware constraints [6]-[8].

Based on these studies and references, the implementation of a 5-stage pipelined RV32I processor with hazard detection and forwarding support serves as a strong academic demonstration of modern processor design techniques, combining architectural concepts with FPGA-based verification.

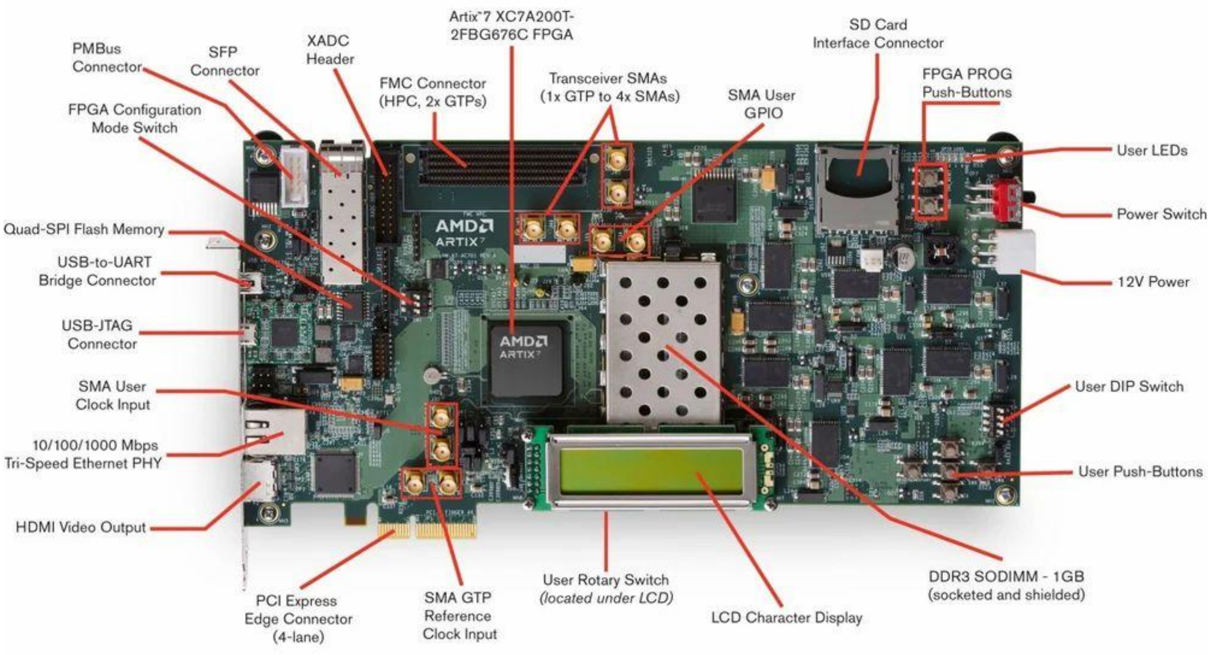
CHAPTER 3
HARDWARE TOOLS AND COMPONENTS

The project involved the implementation of a 5-stage pipelined RV32I RISC processor using an FPGA-based digital design methodology. Focused on the development of processor architecture through Verilog HDL, the key hardware requirement was an FPGA platform capable of synthesis, implementation. The hardware tools and components utilized during this internship project are summarized below.

1. FPGA Development Platform

To deploy the processor in a hardware-oriented environment, an FPGA target device was selected. With their flexibility, FPGA platforms allow the realization of complex digital designs, such as pipelined processors, without necessitating fabrication.

Target FPGA Device



Artix™7 XC7A200T-2FBG676C FPGA
SD Card Interface Connector
PMBus Connector
SFP Connector
XADC Header
FMC Connector (HPC, 2x GTPs)
Transceiver SMAs (1x GTP to 4x SMAs)
SMA User GPIO
FPGA PROG Push-Buttons
FPGA Configuration Mode Switch
User LEDs
Power Switch
Quad-SPI Flash Memory
USB-to-UART Bridge Connector
12V Power
USB-JTAG Connector
SMA User Clock Input
User DIP Switch
10/100/1000 Mbps Tri-Speed Ethernet PHY
User Push-Buttons
HDMI Video Output
PCI Express Edge Connector (4-lane)
SMA GTP Reference Clock Input
User Rotary Switch (located under LCD)
LCD Character Display
DDR3 SODIMM - 1GB (socketed and shielded)

Figure: 1 - AMD Artix™-7 AC701 Evaluation Kit

- **FPGA Family:** AMD Artix™-7
- **Development Board:** AC701 Evaluation Kit
- **Device Support:** Optimized for RTL-based processor implementation and validation

Architecture Highlights:

- Configurable logic blocks (LUTs and Flip-Flops)
- Integrated Block RAM resources to support memory modules
- High-speed clocking infrastructure
- Support for timing analysis and prototyping

The Artix-7 FPGA was specifically chosen for its ample logic resources and its prevalent use in academic research and industrial prototyping of processor cores.

2. System Hardware Requirements

In addition to FPGA-based verification, a robust computer system was required to handle Vivado's synthesis and simulation processes efficiently.

Host System Specifications

- **Processor:** Multi-core Intel or AMD processor
- **RAM:** Minimum 8 GB (16 GB recommended)
- **Storage:** At least 20 GB of free disk space for Vivado project files
- **Operating System:** Windows 10/11 (64-bit)

These specifications were necessary to ensure seamless execution of RTL simulation, synthesis, implementation, and report generation tasks.

3. Clock and Reset Components

For digital designs on FPGA, clock and reset signals are indispensable for ensuring synchronized operation within the processor pipeline.

Clock Source

- A global FPGA clock input was deployed to drive pipeline stages.
- Timing constraints were applied via an XDC file to ensure proper timing closure.

Reset Signal

A synchronous reset mechanism was implemented to initialize the program counter and pipeline registers during system startup.

4. Hardware Modules Designed in RTL

While external peripherals were not incorporated, several critical hardware components were internally developed as Verilog modules to construct a complete processor datapath. These include:

- Program Counter (PC)
- Instruction Memory

- Register File
- Immediate Generator
- ALU and ALU Control Unit
- Pipeline Registers (IF/ID, ID/EX, EX/MEM, MEM/WB)
- Data Memory Module
- Forwarding Unit
- Hazard Detection Unit
- Writeback Multiplexer

Together, these modules constitute the complete 5-stage pipelined processing core.

5. FPGA Implementation Outputs

The FPGA design flow led to the generation of the following implementation outputs:

- Synthesized RTL Netlist
- FPGA Resource Utilization Report
- Static Timing Analysis Report
- Implemented Device Layout View
- Post-Implementation Timing Simulation Results

In this study, the hardware environment primarily comprised an FPGA implementation platform along with a compatible clock and reset infrastructure. The Artix-7 FPGA device was utilized as it offered an efficient and adaptable foundation for the implementation of the RV32I pipelined processor. This research effectively illustrates the process of designing, verifying, and preparing a processor core for deployment on an FPGA, employing conventional hardware design tools and methodologies.

CHAPTER 4
SOFTWARE TOOLS AND LANGUAGES

The successful design and development of a pipelined RISC processor hinge on integrating hardware description languages with FPGA development tools to perform simulation, synthesis, timing verification, and implementation effectively. This internship project focused on the design and validation of a 5-stage pipelined RV32I RISC-V processor core using standard industry-grade tools and methodologies. Below is a detailed overview of the tools and languages employed during the project.

1. Verilog Hardware Description Language (HDL)

The entire processor architecture was implemented using Verilog HDL, a widely adopted hardware description language for digital design.

Project Contributions Using Verilog

- Verilog played a crucial role in:
- Creating the processor datapath modules.
- Implementing pipeline registers between processing stages.
- Developing the control unit for RV32I instructions.
- Designing the Arithmetic Logic Unit (ALU) and its corresponding control logic.
- Building hazard detection and forwarding mechanisms.
- Developing memory interfaces and writeback logic

This language facilitated modular, synthesizable designs, making it ideal for FPGA-based processor development.

2. AMD Xilinx Vivado Design Suite (Vivado 2025.2)

The AMD Xilinx Vivado Design Suite served as the core development and verification toolset for this project. It provided an integrated platform to address different aspects of FPGA design.

Key Functions Performed with Vivado

- The software supported various stages of the design workflow, including:
- RTL design entry and project creation.
- Behavioral simulation with waveform analysis.
- Synthesis of Verilog code into gate-level netlists.
- Post-synthesis functional and timing simulations.
- Static Timing Analysis (STA).

- Physical implementation (place and route).
- Timing verification after implementation.

Additionally, Vivado generated critical reports, such as:

- Resource utilization summaries.
- Timing analysis reports.
- Device layout visualizations.
- Elaborated RTL schematic diagrams.

Overall, Vivado acted as the central tool for end-to-end processor design and implementation.

3. Vivado Simulator (XSim)

Vivado's built-in simulator, XSim, was extensively utilized for functional verification across different stages of the development process.

Simulation Stages Covered

- Simulations were conducted in three major stages:
- Behavioral Simulation: To confirm RTL code correctness.
- Post-Synthesis Simulation: To verify behavior of the synthesized netlist.
- Post-Implementation Timing Simulation: To validate performance considering routing delays.
- Waveform analyses ensured:
- Accurate instruction propagation through pipeline stages.
- Correct register value writeback procedures.
- Effective handling of hazards through forwarding and stalling.

4. XDC Constraint File Format

An XDC (Xilinx Design Constraints) file was created to ensure proper timing constraints and compatibility with the FPGA platform.

Purpose of the XDC File

The constraint file defined key parameters such as:

- Clock input pin assignments.
- Clock frequency constraints.
- Configuration voltage settings.

These constraints were critical for correct system implementation and achieving timing closure.

5. RISC-V Instruction Set Reference

The RV32I RISC-V instruction set architecture formed the basis of the processor design. Relevant documentation was used as a guideline throughout the project.

Reference Utility

- The documentation provided details on:
- Instruction encoding formats.
- Decoding opcode and function fields.
- Generation of immediate values.
- Load/store and branching operations.

The official RISC-V ISA manual served as the authoritative reference for verifying implementation accuracy.

6. Supporting Tools for Documentation and Visualization

To prepare project deliverables and documentation, additional tools were utilized:

- Microsoft Word / Google Docs for detailed report writing.
- Screenshot utility tools like Snipping Tool for capturing simulation outputs and waveform data from Vivado.

These tools played a crucial role in compiling a professional finalized report for the internship project.

Within the scope of this internship project, the integration of Verilog HDL with the Vivado Design Suite established a comprehensive platform for the design, simulation, synthesis, and implementation of a 5-stage pipelined RV32I RISC-V processor. The employed software tools facilitated industry-standard verification processes and ensured the FPGA compatibility of the processor core, effectively meeting all procedural and technical objectives outlined in the internship's project framework.

CHAPTER 5
IMPLEMENTATION

The realization of a 5-stage pipelined RV32I RISC-V processor adhered to a systematic hardware design methodology, designed to align with established principles of digital circuit development. This undertaking extended beyond the architectural construction using Verilog HDL by incorporating functional simulations, synthesis tailored for FPGA integration, and timing validation through comprehensive implementation analysis.

The methodology adopted for this project was anchored in the conventional FPGA-based VLSI design workflow, which encompasses processes such as Register Transfer Level (RTL) modeling, simulation, synthesis, timing verification, and ultimate hardware implementation.

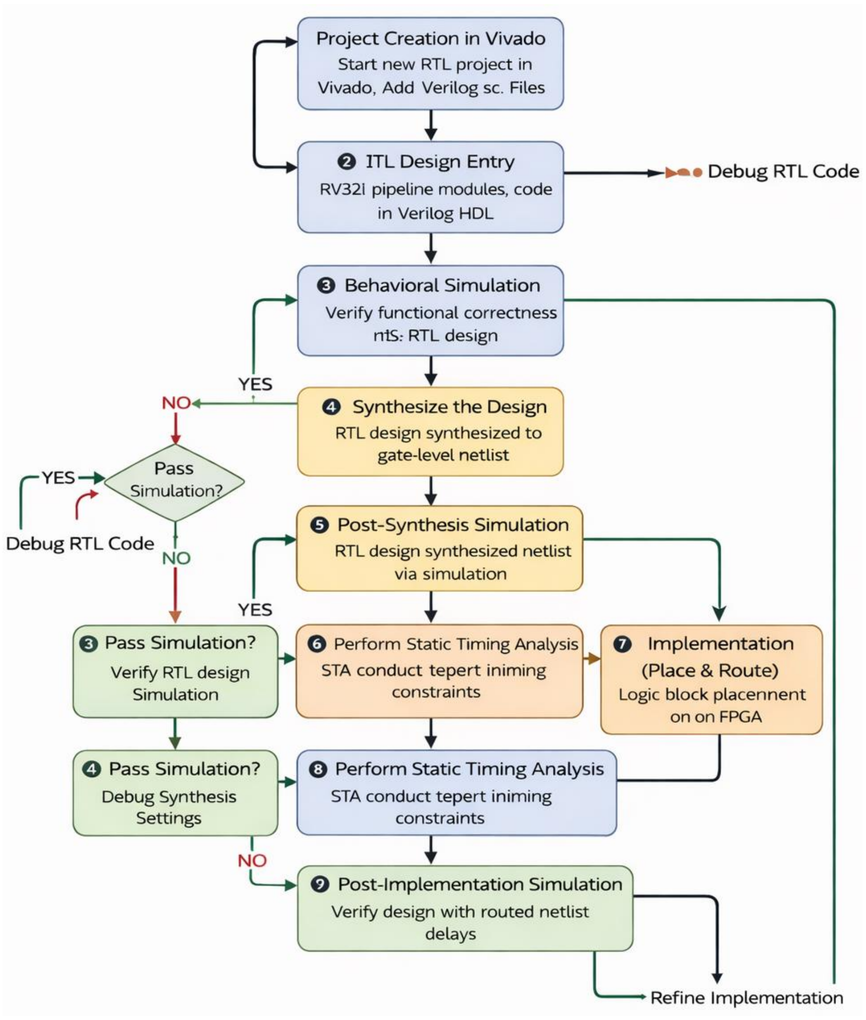

Project Creation in Vivado
Start new RTL project in Vivado, Add Verilog sc. Files
❷ ITL Design Entry
RV32I pipeline modules, code in Verilog HDL
Debug RTL Code
❸ Behavioral Simulation
Verify functional correctness ntS: RTL design
YES
NO
❹ Synthesize the Design
RTL design synthesized to gate-level netlist
YES
Pass Simulation?
Debug RTL Code
NO
❺ Post-Synthesis Simulation
RTL design synthesized netlist via simulation
YES
❸ Pass Simulation?
Verify RTL design Simulation
❻ Perform Static Timing Analysis
STA conduct tepert inimng constraints
❼ Implementation (Place & Route)
Logic block placennent on on FPGA
❹ Pass Simulation?
Debug Synthesis Settings
❽ Perform Static Timing Analysis
STA conduct tepert inimng constraints
NO
❾ Post-Implementation Simulation
Verify design with routed netlist delays
Refine Implementation

Figure: 2 – Design Methodology Architecture

Design Methodology:

The processor design employed a modular and incremental approach, wherein each pipeline stage was independently realized as a distinct Verilog module. The architecture adhered to the classical RV32I 5-stage pipelining structure, incorporating the stages of Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB). Pipeline registers were introduced between successive stages to enable concurrent execution of multiple instructions.

Pipeline Hazard Management:

Given that pipelined processors execute overlapping instructions concurrently, certain hazards may arise due to inter-instruction dependencies. Specific mechanisms were implemented to mitigate these issues effectively:

a. **Data Forwarding:**

The inclusion of a forwarding unit facilitated seamless communication of intermediate results from later pipeline stages back to earlier stages, specifically at the execution stage input. This optimization reduced unnecessary instruction stalls while improving system throughput.

b. **Hazard Detection:**

A hazard detection unit was employed to identify load-use hazards. When such conflicts emerged, the pipeline introduced a single cycle stall, ensuring the correctness of data dependencies.

These mechanisms collectively sustained efficient instruction execution within scenarios characterized by data-dependent requirements.

Implementation Workflows in Vivado:

The full processor development and verification cycle were executed using AMD Xilinx Vivado Design Suite. The implementation steps are outlined below:

Step 1 - Project Initialization:

A new RTL project targeting an FPGA platform was established in Vivado. All Verilog-based source files, constituting various processor modules, were grouped under design sources.

Step 2 - RTL Design Specification:

The processor logic architecture followed a modular framework. Utilizing Verilog HDL, a top-level module was constructed, encapsulating pipeline stages and their interconnecting components.

Step 3 - Behavioral Testing:

Behavioral simulations, leveraging Vivado's Simulator (XSim), verified logical coherence across modules. Key waveform outputs such as program counter progression, fetched instruction data, writeback register identification, and writeback values were examined.

Step 4 - RTL Synthesis:

After successful behavioral verification, synthesis was conducted via Vivado's synthesis engine. This stage translated RTL representations into gate-level netlists encompassing core FPGA logic components such as look-up tables (LUTs) and flip-flops. Post-synthesis utilization summaries provided insight into resource allocation.

Step 5 - Functional Post-Synthesis Verification:

Functional simulations continued post-synthesis to affirm that synthesized netlists accurately replicated RTL behaviors, safeguarding against unintended functionality deviations during synthesis.

Step 6 - Static Timing Validation:

Static timing analysis (STA) evaluated compliance with defined timing constraints. Generated timing reports revealed positive Worst Negative Slack (WNS) values and confirmed zero failing endpoints, substantiating timing reliability across the processor design.

Step 7 - Final Implementation Stages:

Vivado execution extended to placement and routing processes involving logical block installation within FPGA fabric and establishing interconnects along designated paths. Clock network routing was equally verified through device visualization post-placement operations.

Step 8 - Post-Implementation Simulation:

A post-implementation timing simulation was conducted, incorporating realistic routing delays. This served as the final validation step to ensure the processor's behavior aligned with expectations under actual hardware conditions.

CHAPTER 5.1
FLOW CHART

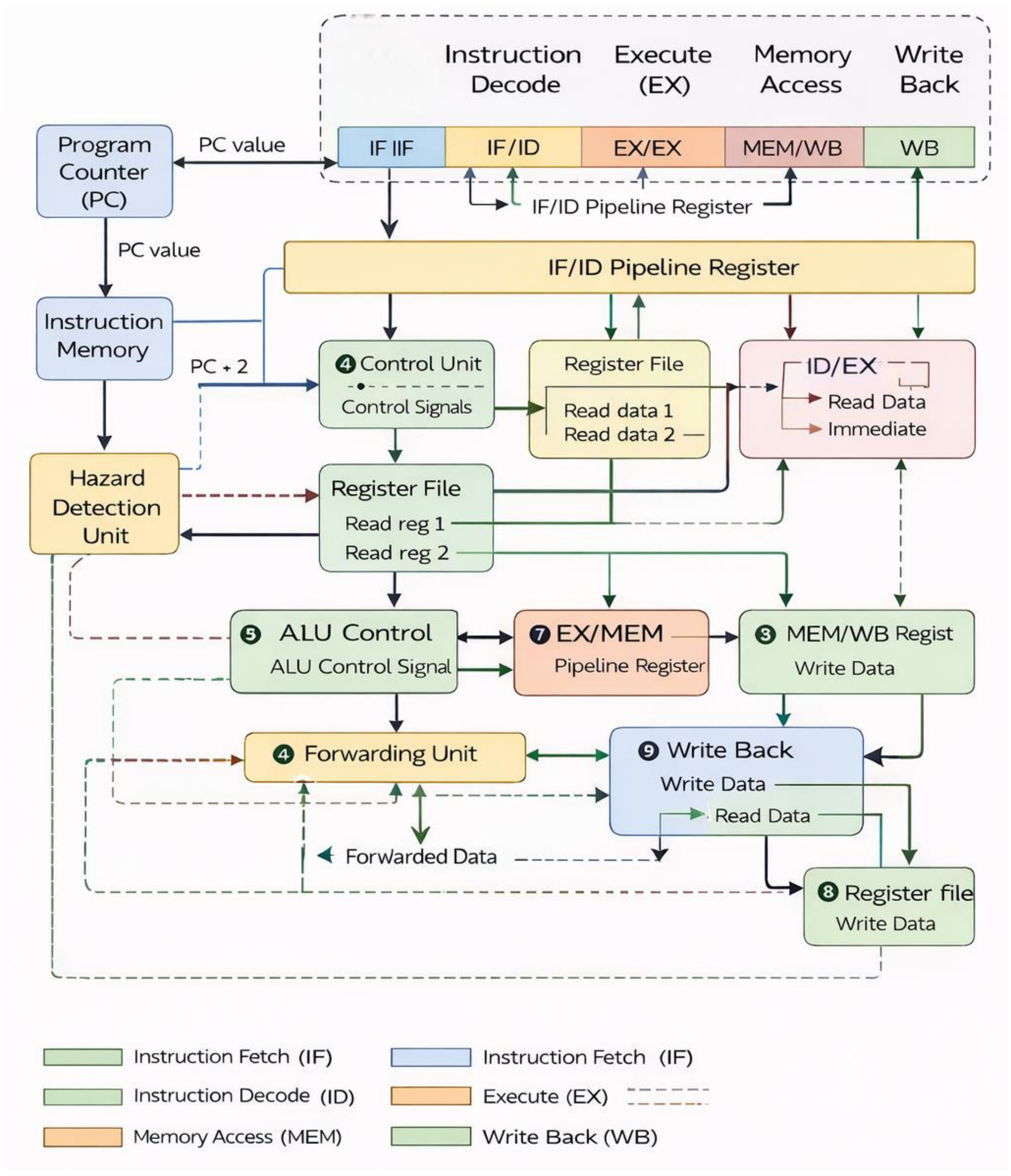


Figure: 3 - Dataflow Architectural Flowchart of the Implemented 5-Stage Pipelined RV32I RISC-V Processor

The above flowchart illustrates the complete internal data movement and execution process of the implemented 5-stage pipelined RV32I RISC processor. It represents how instructions are processed sequentially through different pipeline stages while allowing multiple instructions to execute in parallel for improved throughput.

The processor design follows the classical RISC pipeline architecture, consisting of the following five stages:

1. Instruction Fetch (IF) Stage

The execution begins with the Program Counter (PC) supplying the current instruction address. The instruction stored in the Instruction Memory is fetched based on the PC value. Simultaneously, the PC is incremented to point to the next instruction address, ensuring continuous instruction flow.

2. Instruction Decode (ID) Stage

In this stage, the fetched instruction is decoded by the Control Unit, which generates the necessary control signals for the remaining stages. The source register operands are read from the Register File, and the Immediate Generator produces the required immediate value for I-type, S-type, or branch instructions. The decoded instruction fields are then passed into the ID/EX pipeline register.

3. Execute (EX) Stage

The Execute stage performs the core arithmetic and logical operations using the Arithmetic Logic Unit (ALU). The ALU Control Unit determines the exact operation based on the instruction's funct3/funct7 fields. Additionally, the design includes a Forwarding Unit, which resolves data hazards by selecting the most recent operand values from later pipeline stages instead of waiting for write-back.

4. Memory Access (MEM) Stage

In the MEM stage, load and store instructions interact with the Data Memory. The computed ALU result is used as the effective address for memory operations. Outputs of this stage are stored in the EX/MEM pipeline register for the next stage.

5. Write Back (WB) Stage

Finally, the Write Back stage updates the Register File with either the ALU result or the memory output depending on the instruction type. This ensures correct completion of arithmetic, logical, and load instructions.

6. Hazard Detection and Pipeline Control

The flowchart also highlights the presence of a Hazard Detection Unit, which detects load-use hazards and introduces pipeline stalls when required. Together with forwarding, this ensures correct execution even under data dependency conditions.

This flowchart provides a clear functional representation of how the implemented pipelined processor achieves instruction-level parallelism. By dividing instruction execution into five stages and incorporating hazard resolution mechanisms, the design demonstrates improved performance compared to a non-pipelined processor while maintaining correctness.

CHAPTER 6
RESULTS AND DISCUSSION

This section presents the simulation, synthesis, implementation, and timing verification results obtained for the designed **5-Stage Pipelined RV32I RISC Processor**. The processor was modeled in Verilog HDL and validated through multiple stages in AMD Vivado, including behavioral simulation, post-synthesis functional simulation, post-implementation timing simulation, and FPGA resource analysis. The outputs confirm correct instruction flow through pipeline stages along with successful forwarding and hazard management.

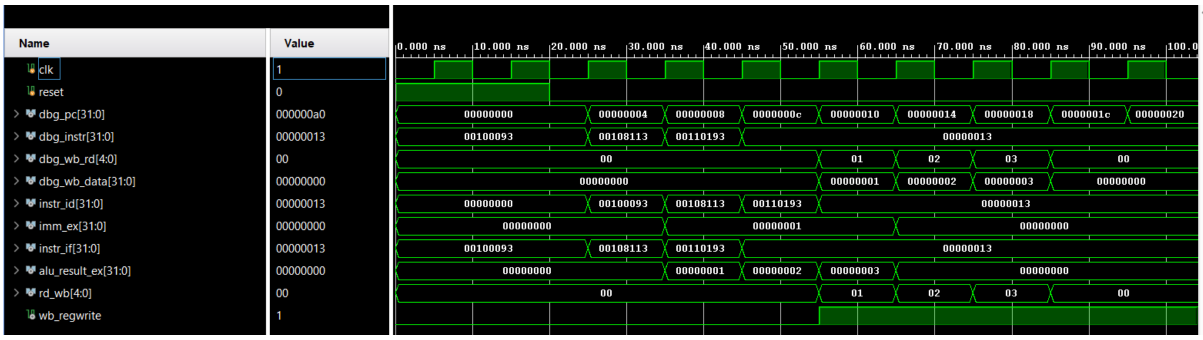
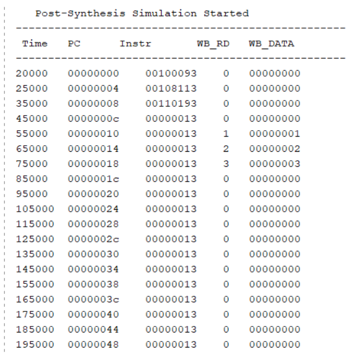


Figure: 4 - Behavioral Simulation Waveform Output

The behavioral simulation waveform verifies functional correctness of the processor at RTL level. The program counter increments sequentially, showing correct instruction fetch operation. Instructions flow through the pipeline stages and register writeback occurs with expected values. The waveform confirms correct execution of RV32I ADDI instructions in a pipelined manner. Thus, the RTL design passes pre-synthesis functional verification.



```
   Post-Synthesis Simulation Started
 ------------------------------------------------------
  Time    PC        Instr        WB_RD   WB_DATA
 ------------------------------------------------------
 20000    00000000    00100093     0    00000000
 25000    00000004    00108113     0    00000000
 35000    00000008    00110193     0    00000000
 45000    0000000c    00000013     0    00000000
 55000    00000010    00000013     1    00000001
 65000    00000014    00000013     2    00000002
 75000    00000018    00000013     3    00000003
 85000    0000001c    00000013     0    00000000
 95000    00000020    00000013     0    00000000
 105000   00000024    00000013     0    00000000
 115000   00000028    00000013     0    00000000
 125000   0000002c    00000013     0    00000000
 135000   00000030    00000013     0    00000000
 145000   00000034    00000013     0    00000000
 155000   00000038    00000013     0    00000000
 165000   0000003c    00000013     0    00000000
 175000   00000040    00000013     0    00000000
 185000   00000044    00000013     0    00000000
 195000   00000048    00000013     0    00000000
```

Figure: 5 - Simulation Console Output Verification

The console output shows executed instructions with corresponding writeback register values. Registers x1, x2, and x3 are correctly updated with values 1, 2, and 3 respectively. This confirms correct pipeline execution and writeback stage operation. The repeated NOP instructions demonstrate stable pipeline flow after program completion. Therefore, the processor successfully executes RV32I instruction sequences.

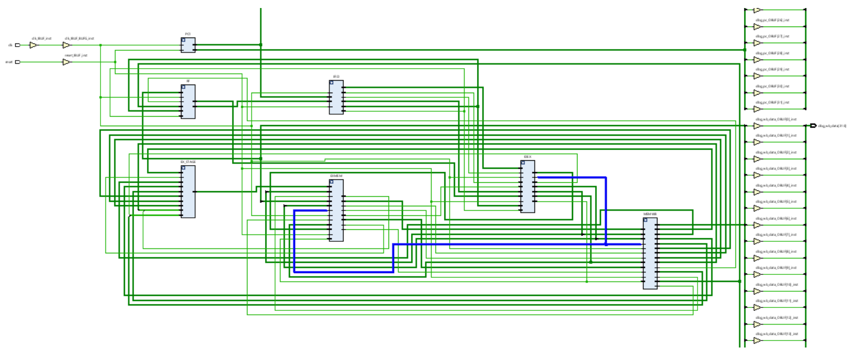


Figure: 6(a) - RTL Schematic of the 5-Stage Pipelined Processor

The RTL schematic represents the elaborated structural view of the designed RV32I processor. It clearly shows the interconnection between major pipeline blocks such as IF, ID, EX, MEM, and WB stages along with pipeline registers (IF/ID, ID/EX, EX/MEM, MEM/WB). This confirms that the modular Verilog design has been successfully integrated into a complete datapath. The schematic also highlights proper signal flow through instruction fetch, execution, and writeback paths. Thus, the design architecture matches the intended 5-stage pipelined processor model.

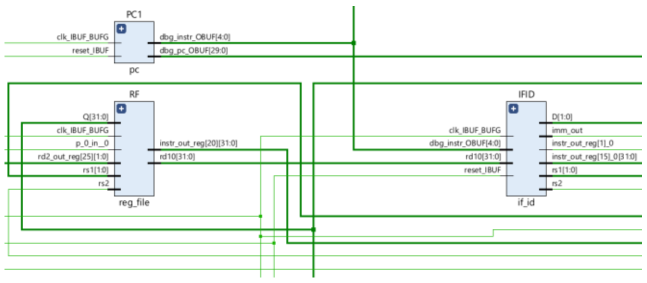


Figure: 6(b) - IF/ID Stage and Register File Interface

This figure illustrates the instruction decode path, including IF/ID pipeline register outputs and Register File read operations. It confirms that source registers (rs1, rs2) and immediate values are correctly passed into the decode stage. The instruction field extraction is visible, validating proper instruction breakdown. This ensures that operands required for ALU execution are correctly prepared. Hence, the decode pipeline stage operates as expected.

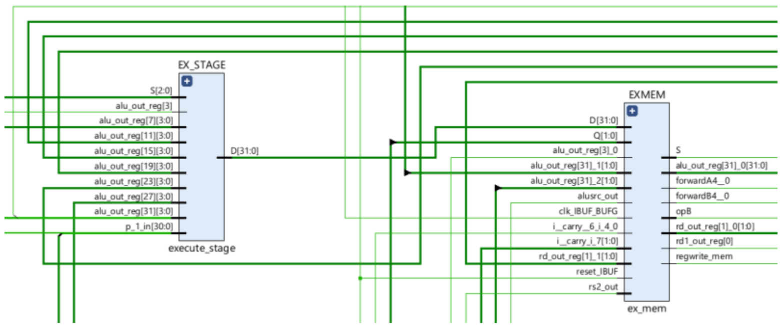


Figure: 6(c) - Execute Stage and EX/MEM Register Connectivity

This zoomed schematic focuses on the Execute stage and its interface with the EX/MEM pipeline register. It confirms that ALU results, control signals, and forwarded operands are properly captured and transferred into the memory stage. The presence of forwarding control lines verifies integration of hazard resolution mechanisms. Correct EX/MEM pipelining ensures that ALU outputs are not lost between stages. Thus, execution-stage functionality is validated structurally.

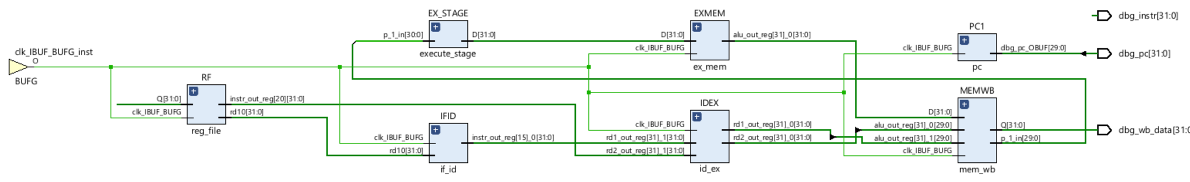


Figure: 7 - Dataflow-Level Pipeline Block Diagram

The dataflow schematic provides a simplified representation of how data moves across pipeline stages. It highlights the instruction progression from Register File through Execute Stage and finally to Writeback. Pipeline registers ensure stage separation and support overlapped instruction execution. This confirms that pipelining has been correctly incorporated into the datapath structure. Such visualization is useful for verifying forwarding and hazard control connections.

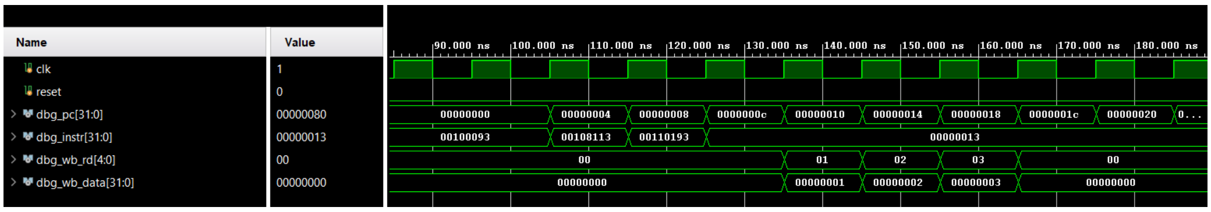
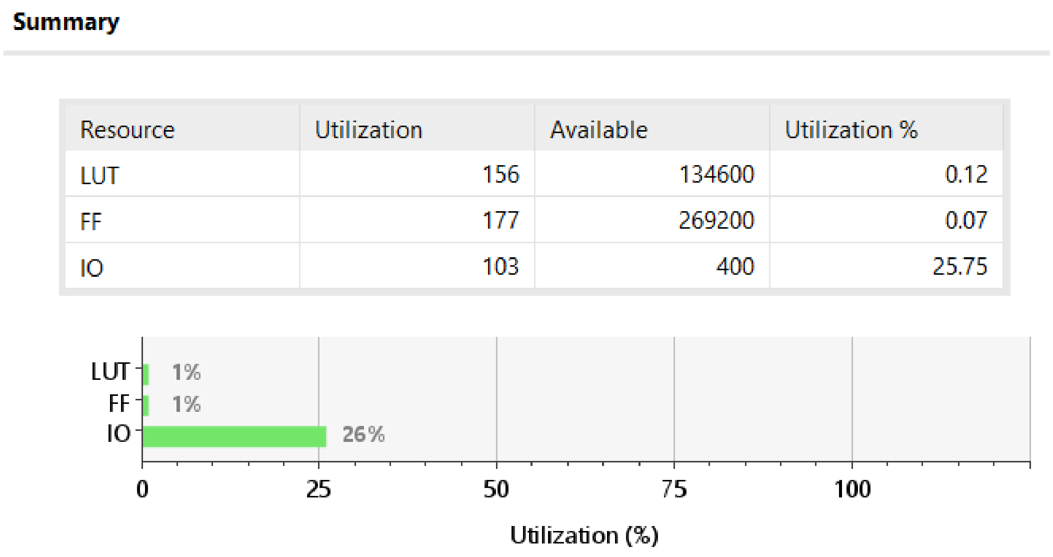


Figure: 8 - Post-Synthesis Functional Simulation Waveform

This waveform represents simulation after synthesis using the generated gate-level netlist. The results match behavioral simulation, confirming that synthesis preserved the processor's logical functionality. Pipeline writeback signals (WB_RD and WB_DATA) appear correctly after expected latency. This validates that the design is synthesizable and functionally stable. Hence, the processor works correctly even after hardware mapping.

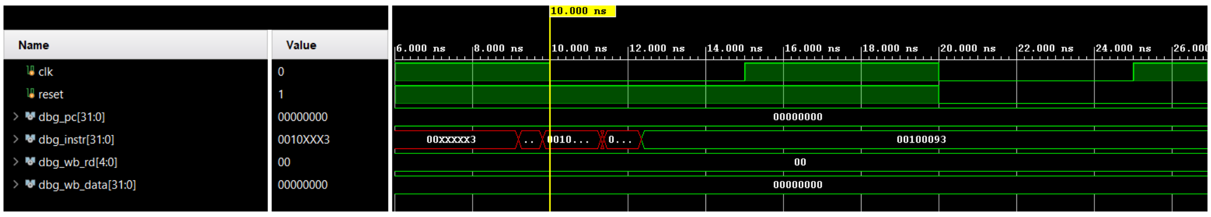


Summary

Resource	Utilization	Available	Utilization %
LUT	156	134600	0.12
FF	177	269200	0.07
IO	103	400	25.75

Figure: 9 - Post-Synthesis Utilization Summary

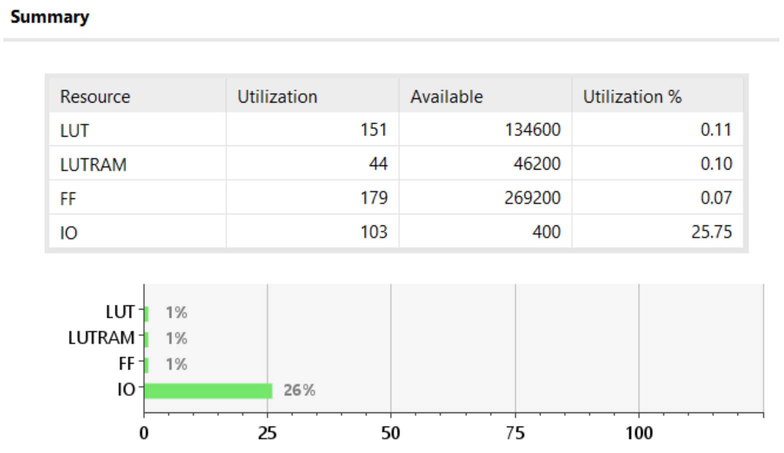
This report summarizes FPGA resources consumed after synthesis. The design uses a very small number of LUTs and flip-flops compared to available device capacity.Low utilization indicates an area-efficient RV32I core implementation. The I/O count appears higher due to debug output ports included for observation. Overall, synthesis results confirm compact and scalable design.



Name	Value
clk	0
reset	1
dbg_pc[31:0]	00000000
dbg_instr[31:0]	0010XXX3
dbg_wb_rd[4:0]	00
dbg_wb_data[31:0]	00000000

Figure: 10 - Post-Implementation Timing Simulation Waveform

The timing summary indicates that all user-defined constraints are satisfied. The positive Worst Negative Slack (WNS) confirms that setup timing violations do not exist. Hold slack is also positive, ensuring stable sequential operation. This demonstrates that the pipelined processor can operate safely at the defined clock frequency. Thus, timing closure is successfully achieved.



Summary

Resource	Utilization	Available	Utilization %
LUT	151	134600	0.11
LUTRAM	44	46200	0.10
FF	179	269200	0.07
IO	103	400	25.75

Figure: 11 - Post-Implementation Utilization Summary

Post-implementation utilization provides final resource usage after routing. Resource consumption remains almost identical to synthesis, confirming stable mapping. The processor occupies only a fraction of the FPGA, leaving significant space for future extensions. This proves that the design is lightweight and FPGA-friendly. Hence, the implementation is efficient and optimized.

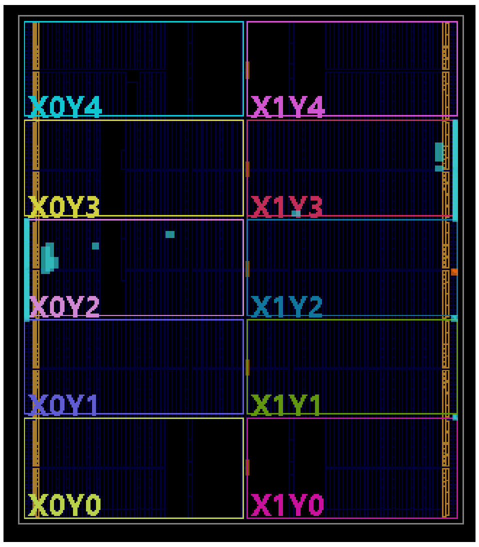
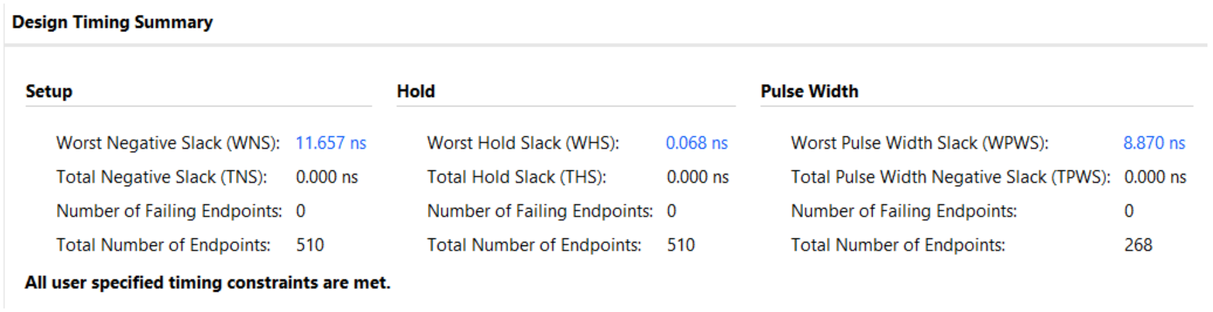


Figure: 12 - Device Placement View after FPGA Implementation

This figure shows the placement of synthesized logic on the selected FPGA device. The distribution of logic blocks across different FPGA regions indicates successful placement and routing during implementation. No congestion or placement failures are observed, confirming efficient mapping of the processor datapath. The sparse utilization reflects the lightweight nature of the RV32I core. Overall, the processor design is implementation-ready on the FPGA platform.



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	11.657 ns	Worst Hold Slack (WHS):	0.068 ns	Worst Pulse Width Slack (WPWS):	8.870 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	510	Total Number of Endpoints:	510	Total Number of Endpoints:	268

All user specified timing constraints are met.

Figure: 13 - Timing Summary Report

The timing summary indicates that all user-defined constraints are satisfied. The positive Worst Negative Slack (WNS) confirms that setup timing violations do not exist. Hold slack is also positive, ensuring stable sequential operation. This demonstrates that the pipelined processor can operate safely at the defined clock frequency. Thus, timing closure is successfully achieved.

CHAPTER 7
CONCLUSION

In this project, a 5-stage pipelined RV32I RISC processor was successfully designed and implemented using Verilog HDL as part of an internship program conducted by NIELIT. The processor was built based on the open-standard RISC-V RV32I instruction set architecture, supporting key operations such as arithmetic execution, immediate-based instructions, memory access, and fundamental control flow.

The pipelined processor was organized into the traditional five stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Writeback (WB). To maintain efficient instruction processing and stage isolation, pipeline registers (IF/ID, ID/EX, EX/MEM, and MEM/WB) were incorporated. This approach improved throughput by enabling concurrent execution of multiple instructions across the stages.

To ensure reliable pipelined execution, mechanisms for hazard detection and data forwarding were implemented. The forwarding unit addressed data hazards by directly rerouting results from later pipeline stages, while the hazard detection unit introduced pipeline stalls to manage load-use dependencies effectively. These features guaranteed accurate and consistent operation without disruptions in the pipeline.

The design underwent robust verification processes, including behavioral simulation, post-synthesis functional simulation, and post-implementation timing simulation using AMD Vivado. The simulation outcomes confirmed correct instruction execution and precise register writeback values. Additionally, static timing analysis indicated positive slack values, affirming that the processor met all timing constraints under the defined clock frequency.

FPGA synthesis and implementation demonstrated that the processor utilized a minimal amount of hardware resources, highlighting its area efficiency and scalability for potential future upgrades. This project achieved its primary goal of creating a functional pipelined RISC-V processor core while offering valuable hands-on experience in RTL design, pipelining strategies, hazard management, FPGA synthesis, and implementation methodologies.

In conclusion, the successful completion of this project not only validates the robustness of the proposed architecture but also establishes a solid foundation for further advancements such as branch prediction units, cache integration, and extension to a complete instruction set.

CHAPTER 8
REFERENCES

[1] A. Waterman and K. Asanović, *The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA*, RISC-V International, Version 2.2, 2019.

[2] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann, 2014.

[3] D. A. Patterson and J. L. Hennessy, *Computer Architecture: A Quantitative Approach*, 6th ed. Amsterdam, Netherlands: Elsevier, 2019.

[4] J. L. Hennessy and D. A. Patterson, "A new era for computer architecture," *Communications of the ACM*, vol. 62, no. 2, pp. 48–60, Feb. 2019.

[5] RISC-V International, "RISC-V Specifications," 2023. [Online]. Available: https://riscv.org/technical/specifications/

[6] AMD Xilinx, *Vivado Design Suite User Guide: Logic Simulation (UG900)*, 2022.

[7] AMD Xilinx, *Vivado Design Suite User Guide: Synthesis (UG901)*, 2022.

[8] AMD Xilinx, *Vivado Design Suite User Guide: Implementation (UG904)*, 2022.

[9] S. Palnitkar, *Verilog HDL: A Guide to Digital Design and Synthesis*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2003.

My Project Github Repository (For Reference): Click Here