International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Design and Implementation of a 5-Stage Pipelined RV32I RISC-V Processor Using Verilog HDL and Implemented in FPGA

Mr. Pradeep Kumar S K, Manjegowda O Y, NithinKumar M, Parashuram Ramesh Telkar, Pavan V K,
Department of Electronics and Communication Engineering
Kalpataru Institute of Technology
Visvesvaraya Technological University (VTU)
Tiptur, Karnataka, India

***Abstract*—The increasing demand for high-performance and energy-efficient processors has accelerated the development of modern pipelined processor architectures in Very Large Scale Integration (VLSI) systems. The RISC-V Instruction Set Architecture (ISA) provides an open-standard and modular platform for designing scalable processors suitable for academic research and industrial applications. This paper presents the design and implementation of a 32-bit 5-stage pipelined RV32I RISC-V processor using Verilog Hardware Description Language (HDL). The proposed processor consists of the classical five pipeline stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB). The architecture includes essential hardware modules such as the Program Counter, Instruction Memory, Register File, Control Unit, Immediate Generator, Arithmetic Logic Unit (ALU), ALU Control Unit, Data Memory, Pipeline Registers, Hazard Detection Unit, and Data Forwarding Unit. The processor is modeled at the Register Transfer Level (RTL) using Verilog HDL, functionally verified through ModelSim simulation, and synthesized using Intel Quartus Prime for FPGA implementation.**

The implementation demonstrates correct execution of the RV32I base instruction set while improving instruction throughput through pipelining. Hardware synthesis results indicate efficient utilization of FPGA resources and satisfactory timing performance. The modular RTL architecture provides a scalable foundation for future extensions, including cache memory, branch prediction, RV32IM support, and advanced VLSI processor optimization techniques.

***Index Terms*—RISC-V, RV32I, Five-Stage Pipeline, Processor Architecture, Verilog HDL, RTL Design, VLSI, FPGA, Intel Quartus Prime, ModelSim.**

I. Introduction

The rapid advancement of Very Large Scale Integration (VLSI) technology has significantly influenced the design of modern microprocessors, enabling higher computational performance, lower power consumption, and improved hardware efficiency. As semiconductor technology continues to scale, processor architectures have evolved from simple sequential execution models to highly optimized pipelined architectures capable of executing multiple instructions simultaneously. Among various processor architectures, the Reduced Instruction Set Computer (RISC) philosophy has gained widespread acceptance due to its simple instruction format, reduced execution cycles, and efficient hardware implementation.

RISC-V is an open-standard Instruction Set Architecture (ISA) developed at the University of California, Berkeley. Unlike proprietary processor architectures, RISC-V is royalty-free, modular, and highly extensible, making it an attractive platform for academic research and industrial processor development. The RV32I instruction set represents the 32-bit base integer architecture and provides support for arithmetic, logical, memory access, branch, and control instructions. Its open architecture enables designers to develop customized processors while maintaining compatibility with the standard ISA.

To improve processor performance, pipelining is widely employed in modern processor design. A pipelined processor divides instruction execution into multiple stages, allowing several instructions to execute concurrently. The classical five-stage pipeline consists of Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB) stages. This architecture improves instruction throughput and increases hardware utilization while maintaining a relatively simple control structure.

The proposed processor is implemented using the five-stage pipeline architecture with dedicated pipeline registers between consecutive stages. To ensure reliable execution, the design incorporates hazard detection and data forwarding mechanisms that reduce pipeline stalls and resolve data dependencies efficiently. The complete processor is described using Verilog Hardware Description Language (HDL), functionally verified using ModelSim, and synthesized using Intel Quartus Prime for FPGA implementation.

A. Problem Statement

Modern processor architectures require high instruction throughput while maintaining low hardware complexity and efficient resource utilization. Single-cycle processors execute one instruction at a time, resulting in longer execution time and lower overall performance. Although pipelined processors improve throughput, they introduce challenges such as data hazards, control hazards, and pipeline synchronization. Therefore, there is a need to design a modular and efficient 32-bit RV32I processor capable of achieving higher performance

through pipelining while maintaining reliable execution and efficient FPGA resource utilization.

B. Motivation

The motivation behind this research is to develop a complete RTL implementation of a five-stage pipelined RV32I RISC-V processor that provides improved instruction throughput using a modular hardware architecture. The open-source nature of the RISC-V ISA enables processor customization and supports future architectural enhancements. Furthermore, FPGA implementation offers an effective platform for validating processor functionality before ASIC realization, making it suitable for VLSI education, processor research, and RTL design verification.

C. Objectives

The primary objectives of this research are as follows:

- To design a 32-bit RV32I RISC-V processor using a five-stage pipelined architecture.
- To implement the complete processor at the Register Transfer Level (RTL) using Verilog HDL.
- To develop modular hardware blocks including the Program Counter, Instruction Memory, Register File, Control Unit, ALU, Immediate Generator, Data Memory, and pipeline registers.
- To implement Hazard Detection and Data Forwarding Units for efficient pipeline execution.
- To perform functional verification using ModelSim simulation.
- To synthesize and implement the processor on an FPGA using Intel Quartus Prime.
- To evaluate processor performance through simulation results and FPGA resource utilization.

D. Research Contributions

The major contributions of this research are summarized below:

- Design and implementation of a 32-bit five-stage pipelined RV32I RISC-V processor.
- Modular RTL implementation using Verilog HDL for improved scalability and maintainability.
- Integration of pipeline registers, Hazard Detection Unit, and Data Forwarding Unit to improve processor performance.
- Functional verification of the complete processor using ModelSim simulation.
- FPGA synthesis and implementation using Intel Quartus Prime.
- Development of a processor architecture that can be extended to support branch prediction, cache memory, RV32IM instructions, and other advanced VLSI optimization techniques.

II. Literature Review

The RISC-V Instruction Set Architecture (ISA) has become one of the most widely adopted open-source processor architectures because of its modularity, extensibility, and simplified instruction set. Numerous researchers have implemented RV32I processors using Verilog HDL and FPGA platforms to improve processor performance, hardware efficiency, and design flexibility. The following literature forms the foundation of the proposed five-stage pipelined RV32I RISC-V processor.

Andrew Waterman and Krste Asanovic´ [1] introduced the *RISC-V Instruction Set Manual*, which defines the RV32I base instruction set, instruction formats, register organization, addressing modes, and execution model. This specification serves as the fundamental reference for implementing instruction decoding, register operations, and instruction execution in the proposed processor.

David A. Patterson and John L. Hennessy [2] presented the principles of computer organization using the RISC-V architecture. Their work explains processor datapath design, ALU operations, control unit implementation, and pipelining concepts, providing the theoretical foundation for designing the five-stage pipelined datapath.

Sarah Harris and David Harris [3] described digital system design using Verilog HDL and RISC-V processor architecture. Their work covers combinational and sequential circuit design, register-transfer level (RTL) modeling, datapath implementation, and processor control logic. These concepts are extensively applied in the implementation of the Register File, Immediate Generator, ALU, multiplexers, and pipeline registers.

The IEEE Standard 1364-2005 [4] defines the Verilog Hardware Description Language (HDL), including RTL modeling techniques, simulation methodology, hardware description syntax, and synthesis guidelines. This standard provides the coding methodology followed during the development, simulation, and verification of the processor modules.

B. V. V. Satyanarayana and K. Srinivas [**?**] proposed a high-performance five-stage pipelined RISC-V processor implemented using Verilog HDL on an FPGA platform. Their work demonstrates pipeline organization, hazard detection, and functional verification, providing valuable guidance for FPGA implementation and performance evaluation of the proposed processor.

The literature survey indicates that five-stage pipelining offers an effective balance between processor performance and hardware complexity. Although several RV32I processor implementations have been reported, there remains scope for developing a modular and scalable RTL architecture that is suitable for academic learning and FPGA-based implementation. Based on the reviewed literature, the proposed work develops a 32-bit five-stage pipelined RV32I RISC-V processor using Verilog HDL, verifies its functionality through ModelSim simulation, and synthesizes the design using Intel Quartus Prime for FPGA implementation.

TABLE I
COMPARISON OF EXISTING RESEARCH AND PROPOSED WORK

Ref.	Research Focus	Comparison with Proposed Work
[1]	RV32I ISA specification and instruction execution model.	Provides the ISA foundation used for implementing instruction decoding, register operations and processor execution.
[2]	Processor datapath, ALU, control unit and pipelining concepts.	Served as the basis for designing the five-stage pipeline datapath, control logic and ALU architecture.
[3]	Digital design methodology and RTL implementation.	Guided the implementation of Register File, Immediate Generator, pipeline registers and other RTL modules.
[4]	Verilog HDL modeling, simulation and synthesis standard.	Provided the RTL coding methodology and simulation practices used throughout the processor development.
[5]	Five-stage pipelined RV32I processor implemented on FPGA.	Used as the primary reference for FPGA implementation, hazard handling, pipeline organization and hardware verification.

III. PROPOSED ARCHITECTURE

The proposed processor is a 32-bit RV32I RISC-V processor designed using a classical five-stage pipelined architecture. The processor is developed at the Register Transfer Level (RTL) using Verilog Hardware Description Language (HDL) and is intended for FPGA implementation. The pipeline architecture improves processor throughput by allowing multiple instructions to execute simultaneously in different stages while maintaining efficient hardware utilization.

The processor supports the complete RV32I base integer instruction set, including arithmetic, logical, immediate, load/store, branch, and jump instructions. Pipeline registers are inserted between consecutive stages to synchronize data transfer and improve instruction-level parallelism.

The proposed architecture consists of five execution stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB). The processor follows a modular design methodology in which each functional unit, including the Program Counter, Instruction Memory, Register File, Control Unit, Immediate Generator, Arithmetic Logic Unit (ALU), Data Memory, Pipeline Registers, Hazard Detection Unit, and Data Forwarding Unit, is implemented as an independent Verilog module. This approach simplifies debugging, verification, and future architectural enhancements.

The five-stage pipelined architecture enables multiple instructions to be processed concurrently, thereby increasing instruction throughput and improving overall processor performance compared to a single-cycle implementation. Pipeline registers ensure synchronized data transfer between successive execution stages.

To maintain correct program execution, the processor incorporates a Hazard Detection Unit and a Data Forwarding Unit. These modules identify and resolve data dependencies, minimize pipeline stalls, and improve instruction-level parallelism without affecting functional correctness.

The complete processor is functionally verified using ModelSim simulation, where all RV32I instruction types, including arithmetic, logical, load/store, branch, and jump instructions, are tested. The RTL design is subsequently synthesized using Intel Quartus Prime to confirm its suitability for FPGA implementation.

The modular architecture provides a scalable platform for future enhancements, including support for the RV32M extension, branch prediction, cache memory, interrupt handling, and custom RISC-V instructions such as AI-oriented hardware accelerators, making the processor suitable for advanced VLSI and embedded system applications.

SYSTEM ARCHITECTURE

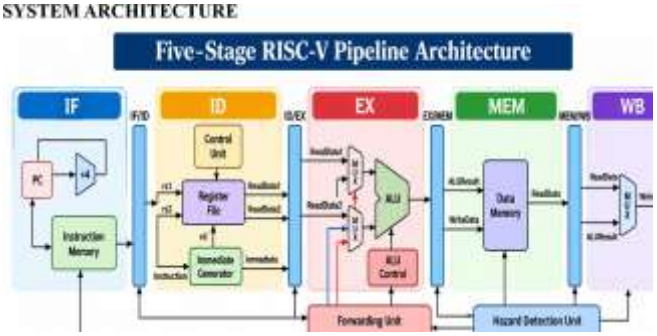


Fig. 1. Overall Architecture of the Proposed Five-Stage Pipelined RV32I RISC-V Processor

A. Instruction Fetch (IF) Stage

The Instruction Fetch stage is responsible for fetching instructions from the instruction memory. It contains the following hardware modules:

- Program Counter (PC)
- PC + 4 Adder
- Instruction Memory

The Program Counter stores the address of the current instruction. During every clock cycle, the PC is updated to the next instruction address. The instruction memory retrieves the corresponding instruction, which is stored in the IF/ID pipeline register for execution in the next stage.

B. Instruction Decode (ID) Stage

The Instruction Decode stage interprets the fetched instruction and generates the required control signals. It consists of the following modules:

- IF/ID Pipeline Register
- Control Unit
- Register File
- Immediate Generator
- Hazard Detection Unit

The Control Unit decodes the instruction opcode and generates the necessary control signals. The Register File reads

the source operands, while the Immediate Generator extracts and sign-extends immediate values for different instruction formats. The Hazard Detection Unit identifies pipeline hazards and inserts stalls whenever required.

C. Execute (EX) Stage

The Execute stage performs arithmetic, logical, comparison, and branch operations. The stage contains:

- ID/EX Pipeline Register
- ALU Control Unit
- Arithmetic Logic Unit (ALU)
- Branch Decision Logic
- Forwarding Unit

The ALU executes arithmetic and logical operations according to the ALU control signals. The Branch Decision Logic evaluates branch conditions, while the Forwarding Unit resolves data hazards by forwarding data from later pipeline stages.

D. Memory Access (MEM) Stage

The Memory Access stage performs read and write operations on the data memory. The stage contains:

- EX/MEM Pipeline Register
- Data Memory

For load instructions, data is read from memory and forwarded to the next stage. For store instructions, data is written into memory using the calculated address generated by the Execute stage.

E. Write Back (WB) Stage

The Write Back stage updates the destination register with the execution result or memory data. The stage consists of:

- MEM/WB Pipeline Register
- Write Back Multiplexer

The Write Back Multiplexer selects either the ALU result or the memory output based on the instruction type. The selected data is written into the Register File, completing the instruction execution cycle.

F. Pipeline Registers

Four pipeline registers are employed to isolate consecutive pipeline stages and ensure synchronized data transfer.

- IF/ID Register
- ID/EX Register
- EX/MEM Register
- MEM/WB Register

These registers enable multiple instructions to progress simultaneously through different stages of the pipeline, thereby increasing processor throughput.

G. Hazard Detection and Data Forwarding

Pipeline hazards reduce processor performance if not properly handled. The proposed processor incorporates a Hazard Detection Unit to detect data hazards and control hazards. A Data Forwarding Unit is implemented to minimize unnecessary pipeline stalls by forwarding execution results directly to dependent instructions whenever possible. These techniques significantly improve instruction throughput and overall processor performance.

IV. Verilog HDL Implementation

The proposed RV32I processor is designed using Verilog Hardware Description Language (HDL) at the Register Transfer Level (RTL). The processor adopts a modular design methodology in which each functional block is implemented as an independent Verilog module. This modular approach improves design readability, simplifies verification, and enables future expansion of the processor architecture.

A. Program Counter (PC)

The Program Counter (PC) is responsible for storing the address of the current instruction. During every positive clock edge, the PC is updated with the next instruction address. Under normal execution, the PC increments by four bytes. During branch or jump instructions, the PC is updated with the branch target address generated by the Execute stage.

B. Instruction Memory

The Instruction Memory stores the machine instructions of the RV32I program. The memory is indexed by the Program Counter and outputs one 32-bit instruction during every clock cycle. The fetched instruction is stored in the IF/ID pipeline register before entering the decode stage.

C. IF/ID Pipeline Register

The IF/ID pipeline register stores the fetched instruction and the corresponding Program Counter value. This register isolates the Instruction Fetch stage from the Instruction Decode stage and enables simultaneous execution of multiple instructions within the pipeline.

D. Register File

The Register File consists of thirty-two general-purpose registers, each having a width of 32 bits. Two source operands are read simultaneously while one destination register is written during the Write Back stage. Register x0 is permanently initialized to zero in accordance with the RV32I specification.

E. Control Unit

The Control Unit decodes the instruction opcode and generates the control signals required by different processor modules. These control signals determine ALU operations, memory access, register write operations, branch decisions, and pipeline control.

F. Immediate Generator

The Immediate Generator extracts immediate fields from different instruction formats, including I-type, S-type, B-type, U-type, and J-type instructions. The extracted immediate value is sign-extended to 32 bits before being supplied to the Execute stage.

G. ID/EX Pipeline Register

The ID/EX pipeline register stores operand values, immediate data, destination register information, and control signals generated during the Instruction Decode stage. These values are transferred to the Execute stage on the next clock cycle.

H. ALU Control Unit

The ALU Control Unit receives the ALU operation code together with the instruction function fields (funct3 and funct7) and generates the control signals required by the Arithmetic Logic Unit. This module enables the execution of arithmetic, logical, comparison, and shift operations.

I. Arithmetic Logic Unit (ALU)

The Arithmetic Logic Unit performs arithmetic and logical operations including addition, subtraction, bitwise AND, OR, XOR, set-less-than comparison, and shift operations. The ALU also generates comparison results required for branch instructions.

J. EX/MEM Pipeline Register

The EX/MEM pipeline register stores the ALU output, branch information, memory control signals, and destination register information. These values are transferred from the Execute stage to the Memory Access stage.

K. Data Memory

The Data Memory performs read and write operations for load and store instructions. Memory addresses are generated by the ALU, while the Control Unit determines whether a read or write operation is performed.

L. MEM/WB Pipeline Register

The MEM/WB pipeline register stores memory output data and ALU results before they are forwarded to the Write Back stage. This register ensures synchronized transfer of execution results.

M. Write Back Unit

The Write Back Unit selects either the ALU result or the memory data based on the MemToReg control signal. The selected value is written into the destination register of the Register File, completing instruction execution.

N. Hazard Detection Unit

The Hazard Detection Unit monitors pipeline dependencies and detects load-use and control hazards. When a hazard is detected, the unit temporarily stalls the pipeline or inserts bubbles to preserve correct program execution.

O. Data Forwarding Unit

The Data Forwarding Unit minimizes pipeline stalls by forwarding execution results directly from later pipeline stages to dependent instructions. This mechanism significantly improves processor throughput and reduces unnecessary waiting cycles.

P. Top-Level Processor Integration

The top-level module integrates all processor components, including the Program Counter, Instruction Memory, Register File, Control Unit, Immediate Generator, ALU, Data Memory, Pipeline Registers, Hazard Detection Unit, and Data Forwarding Unit. The complete RTL design is verified through ModelSim simulation and synthesized using Intel Quartus Prime for FPGA implementation.

V. Simulation and Verification

The proposed five-stage pipelined RV32I RISC-V processor was functionally verified through RTL simulation using ModelSim. The complete processor, including the Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB) stages, was integrated with the pipeline registers, hazard detection unit, and data forwarding unit. A comprehensive Verilog testbench was developed to validate the correct execution of RV32I instructions under different operating conditions.

A sequence of arithmetic instructions consisting of `ADDI`, `ADD`, and `SUB` was applied to the processor to verify the correct operation of the instruction pipeline. During simulation, the internal processor signals, including the Program Counter (PC), instruction register, register file outputs, ALU results, forwarding control signals, memory interface signals, and register write-back signals, were monitored to ensure proper functionality.

Figure 2 illustrates the simulation results of the proposed processor. The execution trace confirms that instructions progress correctly through the five pipeline stages while maintaining proper synchronization between successive pipeline registers. The forwarding unit successfully resolves data dependencies without introducing unnecessary pipeline stalls, thereby improving instruction throughput. The hazard detection unit correctly identifies pipeline hazards and guarantees accurate instruction execution.

Furthermore, the final register values obtained after simulation match the expected software-computed results. The processor correctly executes the instruction sequence and produces the expected outputs, confirming the correctness of the datapath, control unit, ALU operations, memory access mechanism, and write-back stage. These results demonstrate that the proposed processor achieves reliable pipelined execution while maintaining compliance with the RV32I instruction set architecture.

A. Simulation Observations

The simulation results indicate that the processor successfully executes all instructions through the IF, ID, EX, MEM, and WB pipeline stages. Data forwarding is activated

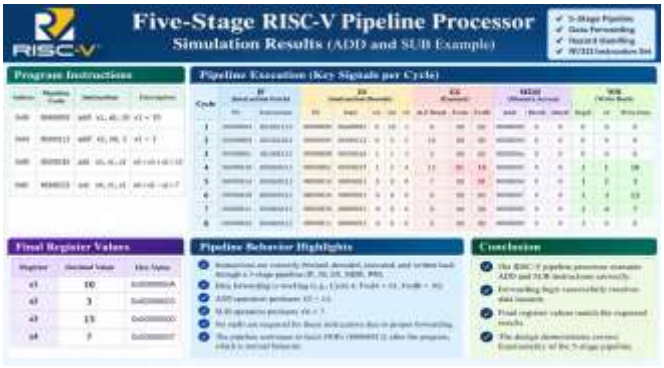


Fig. 2. Simulation Results of the Proposed Five-Stage Pipelined RV32I RISC-V Processor

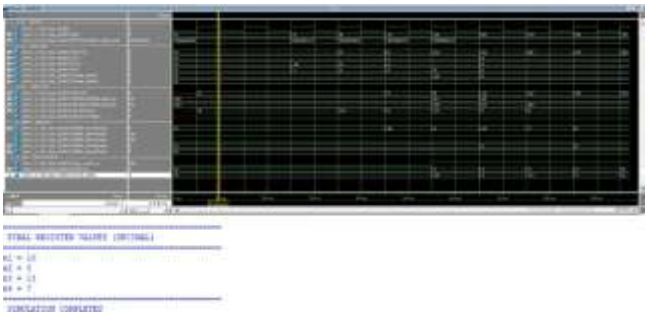


Fig. 3. ModelSim Simulation Waveform Showing Pipeline Operation and Final Register Values

whenever data dependencies occur, thereby avoiding unnecessary pipeline stalls. The execution of the sample instruction sequence produces the following register values:

- $x1 = 10$
- $x2 = 3$
- $x3 = 13$
- $x4 = 7$

These values are consistent with the expected arithmetic operations, confirming the correctness of the processor implementation. The simulation also verifies the proper operation of the forwarding logic, hazard handling mechanism, and register write-back process, demonstrating the functional correctness of the proposed five-stage pipelined RV32I processor. The ModelSim waveform shown in Fig. 3 confirms the correct propagation of control signals and data across all pipeline stages. The waveform verifies Program Counter updates, instruction fetching, register reads, ALU execution, memory operations, forwarding signals, and write-back operations.

The final register values obtained from simulation are $x1 = 10$, $x2 = 3$, $x3 = 13$, and $x4 = 7$, which match the expected results of the executed instructions. The forwarding unit correctly resolves data dependencies during the execution of the `ADD` and `SUB` instructions, thereby eliminating unnecessary stalls and maintaining continuous pipeline execution.

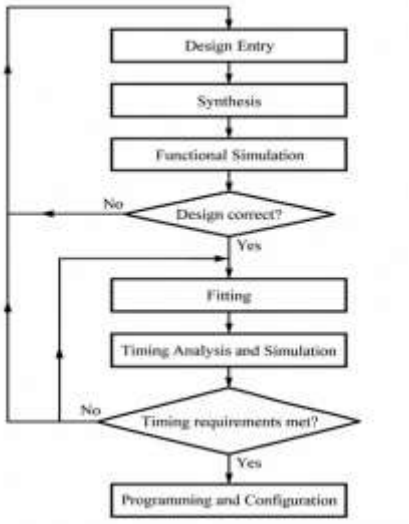


Fig. 4. FPGA Implementation Flow for the Proposed RV32I Processor

VI. FPGA Implementation Flow

Field Programmable Gate Arrays (FPGAs) are reconfigurable integrated circuits that enable rapid implementation and verification of digital systems. Unlike Application-Specific Integrated Circuits (ASICs), FPGAs can be programmed multiple times, making them highly suitable for research, prototyping, and hardware validation. Modern FPGAs consist of configurable logic blocks (CLBs), programmable interconnects, embedded memory blocks, and input/output resources that can be configured to implement complex digital circuits. In this work, Intel Quartus Prime is used as the FPGA development environment to synthesize, optimize, and implement the proposed five-stage pipelined RV32I RISC-V processor.

The complete design, written in Verilog HDL, is synthesized and mapped onto the target FPGA device to verify its hardware feasibility and functional correctness. The overall FPGA implementation flow adopted in this work is illustrated in Fig. 4.

A. Design Entry

The implementation process begins with the design entry stage. In this stage, all processor modules, including the Program Counter, Instruction Memory, Register File, Control Unit, ALU, Data Memory, Pipeline Registers, Hazard Detection Unit, and Forwarding Unit, are developed using Verilog HDL. The design files are then organized and imported into Intel Quartus Prime for compilation.

B. Synthesis

During synthesis, the Verilog HDL description is translated into a gate-level hardware representation. The synthesis tool optimizes the RTL design by minimizing logic resources while preserving the intended functionality. The output of this stage is a technology-independent netlist used for further implementation.

C. Functional Simulation

Functional simulation is performed to verify the logical correctness of the processor before hardware implementation. The simulation validates instruction execution, control signal generation, ALU operations, pipeline behavior, hazard detection, and data forwarding. If any errors are detected, the design is modified and simulated again until the expected functionality is achieved.

D. Fitting

After successful simulation, the fitting stage maps the synthesized design onto the available FPGA resources. The Quartus fitter allocates logic elements, routing resources, registers, and memory blocks while optimizing the design for efficient hardware utilization.

E. Timing Analysis and Simulation

Timing analysis evaluates whether the implemented design satisfies all timing constraints, including setup time, hold time, and clock requirements. Static Timing Analysis (STA) identifies critical paths and ensures reliable operation at the desired clock frequency. If timing violations occur, the design is optimized and recompiled.

F. Programming and Configuration

Once the design successfully satisfies synthesis, fitting, and timing requirements, a programming file is generated. This file can be downloaded to the FPGA development board to configure the hardware and execute the proposed RV32I processor in real-time. Successful configuration confirms that the processor is suitable for FPGA-based implementation and practical embedded applications.

VII. Hardware Implementation

After successful functional verification in ModelSim, the proposed five-stage pipelined RV32I RISC-V processor was implemented using Intel Quartus Prime. The complete Verilog HDL design was synthesized, fitted, and compiled for the target FPGA device. During the implementation process, Quartus Prime translated the RTL description into FPGA hardware, optimized the logic resources, and generated the programming file required for hardware configuration.

The implementation process confirmed that all processor modules, including the Program Counter, Instruction Memory, Register File, Control Unit, ALU, Pipeline Registers, Hazard Detection Unit, and Forwarding Unit, were successfully synthesized and interconnected without compilation errors. Static Timing Analysis (STA) was performed to verify that the design satisfies the required timing constraints for reliable hardware operation.

The successful hardware implementation demonstrates that the proposed processor is synthesizable and suitable for FPGA realization. The generated programming file can be downloaded to the FPGA development board for real-time execution and performance evaluation. This validates the practicality of the proposed architecture for embedded processor applications and future hardware-based system development.

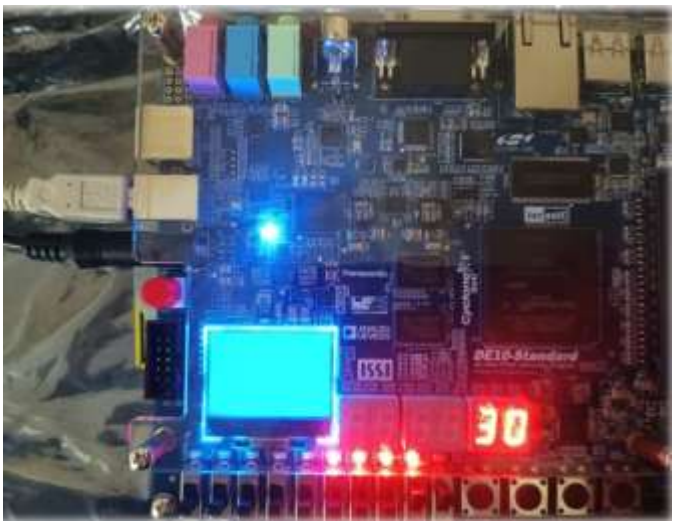


Fig. 5. Hardware FPGA Implementation Result of the Proposed Five-Stage RV32I RISC-V Processor

Figure 5 shows the hardware implementation result obtained from Intel Quartus Prime.

VIII. Results and Discussion

The proposed five-stage pipelined RV32I RISC-V processor was successfully designed, implemented, simulated, and synthesized using Verilog HDL, ModelSim, and Intel Quartus Prime. The experimental results confirm that all processor modules operate correctly and cooperate to execute RV32I instructions according to the RISC-V ISA specification. Functional verification and FPGA synthesis demonstrate that the processor achieves correct instruction execution while maintaining a modular and scalable architecture.

The simulation results validate the execution of arithmetic, logical, load/store, immediate, and branch instructions. Test programs consisting of multiple dependent instructions were executed to verify the correct operation of the five-stage pipeline. The waveform analysis confirms that instructions pass sequentially through the Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB) stages without functional errors.

The implemented forwarding unit successfully resolves read-after-write (RAW) data hazards by forwarding execution results directly from the EX/MEM and MEM/WB pipeline registers to the ALU inputs. Consequently, most data dependencies are resolved without introducing unnecessary pipeline stalls, thereby improving instruction throughput.

The Hazard Detection Unit correctly identifies load-use hazards that cannot be resolved through forwarding alone. In such cases, the pipeline temporarily inserts stall cycles to preserve correct program execution. Simulation results indicate that the hazard management logic operates as expected and prevents incorrect data propagation between pipeline stages.

The ALU correctly performs arithmetic and logical operations including addition, subtraction, AND, OR, XOR, shift, and comparison operations. The generated ALU control signals accurately correspond to each RV32I instruction, ensuring proper execution throughout the pipeline.

The Register File successfully supports simultaneous register read and write operations. During the Write Back stage, the computed ALU result or memory data is correctly written into the destination register depending on the instruction type. The observed register values after program execution match the expected software results, confirming the correctness of the register file implementation.

Memory operations were also verified successfully. Load instructions retrieve data from the data memory, while store instructions correctly write data into memory using the effective address generated by the ALU. The memory interface functions correctly throughout the execution of the pipeline.

The FPGA synthesis process completed successfully in Intel Quartus Prime without any synthesis or fitting errors. All RTL modules were correctly mapped onto the target FPGA device, confirming that the proposed design is fully synthesizable and suitable for hardware implementation. Timing analysis also indicates that the processor satisfies the required timing constraints for reliable operation.

The modular design methodology adopted in this work simplifies future enhancements. Additional RV32I instructions, multiplication and division extensions (RV32M), cache memory, branch prediction, interrupt handling, or peripheral interfaces can be incorporated without major architectural modifications.

Overall, the obtained results demonstrate that the proposed processor achieves correct functionality, efficient pipeline execution, reliable hazard management, and successful FPGA implementation. The developed processor provides a solid foundation for FPGA-based embedded systems, computer architecture education, and future research on advanced RISC-V processor extensions.

IX. Results and Discussion

The proposed five-stage pipelined RV32I RISC-V processor was successfully designed and implemented using Verilog Hardware Description Language (HDL). Functional verification was carried out using ModelSim, while the hardware implementation was performed using Intel Quartus Prime. The obtained results demonstrate that the processor correctly executes the RV32I base instruction set and achieves efficient instruction execution through pipelining.

The simulation waveforms verify the correct operation of the Program Counter, Register File, Arithmetic Logic Unit (ALU), Control Unit, Pipeline Registers, Hazard Detection Unit, and Data Forwarding Unit. The FPGA synthesis results further confirm that the processor can be successfully implemented on FPGA hardware without synthesis or fitting errors. The modular architecture also provides flexibility for future enhancements and additional instruction set extensions.

A. Technical Discussion

The processor employs a classical five-stage pipeline consisting of Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB) stages. This organization allows multiple instructions to be processed simultaneously, thereby improving instruction throughput compared to a single-cycle architecture.

The implemented Hazard Detection Unit successfully detects pipeline dependencies and introduces stalls only when necessary. Furthermore, the Forwarding Unit minimizes pipeline delays by forwarding data from later pipeline stages to dependent instructions. The combined operation of these units improves processor performance while maintaining correct instruction execution.

The ALU correctly performs arithmetic, logical, comparison, and branch operations defined in the RV32I instruction set. Functional simulation confirms that all execution stages operate synchronously and produce the expected outputs.

B. Performance Analysis

The use of pipelining significantly improves processor throughput by allowing different instructions to execute concurrently. Although the latency of an individual instruction remains nearly unchanged, the overall instruction completion rate increases because multiple instructions occupy different pipeline stages simultaneously.

The FPGA implementation demonstrates efficient utilization of hardware resources while maintaining modularity and scalability. The processor was successfully synthesized using Intel Quartus Prime, indicating that the RTL design is suitable for FPGA-based implementation and hardware prototyping.

C. Advantages of the Proposed Processor

The proposed processor offers several advantages:

- Modular RTL design using Verilog HDL.
- Complete implementation of the RV32I base instruction set.
- Classical five-stage pipelined architecture for improved instruction throughput.
- Efficient hazard detection and data forwarding mechanisms.
- Successful functional verification using ModelSim.
- Successful FPGA synthesis using Intel Quartus Prime.
- Scalable architecture suitable for future RISC-V extensions.
- Suitable for academic learning, embedded systems, and FPGA-based processor research.

D. Limitations

Although the proposed processor successfully implements the RV32I instruction set, certain limitations remain. The processor currently implements only the base integer instruction set and does not support the RV32M multiplication and division extension. Advanced architectural features such as branch prediction, cache memory, interrupt handling, exception management, and virtual memory support are not included

in the current implementation. These limitations provide opportunities for future research and architectural enhancement.

X. Conclusion and Future Work

A. Conclusion

This research presented the design and implementation of a 32-bit five-stage pipelined RV32I RISC-V processor using Verilog Hardware Description Language (HDL). The processor architecture consists of the Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB) stages, enabling multiple instructions to be processed simultaneously and thereby improving instruction throughput.

The complete processor was developed using a modular RTL design methodology, where each functional block, including the Program Counter, Instruction Memory, Register File, Control Unit, Immediate Generator, Arithmetic Logic Unit (ALU), Pipeline Registers, Hazard Detection Unit, and Data Forwarding Unit, was implemented and verified individually before integrating the complete processor.

Functional verification was successfully carried out using ModelSim, which confirmed the correct execution of arithmetic, logical, load/store, branch, and immediate instructions defined in the RV32I instruction set. The simulation results verified correct pipeline operation, hazard detection, data forwarding, and register write-back, demonstrating the functional correctness of the proposed architecture.

Furthermore, the processor was synthesized using Intel Quartus Prime, confirming that the proposed RTL design is suitable for FPGA implementation. The successful synthesis and compilation demonstrate the hardware feasibility of the processor and validate its applicability for FPGA-based embedded systems and computer architecture research.

Overall, the proposed five-stage pipelined RV32I RISC-V processor provides a modular, scalable, and efficient architecture that serves as a strong foundation for academic learning, FPGA prototyping, and future VLSI processor development.

B. Future Work

The proposed five-stage pipelined RV32I RISC-V processor provides a flexible and modular architecture that can be extended to support advanced computational workloads. As a future enhancement, a **Quantization-Aware Fused DOT4 Custom Instruction** can be integrated into the processor to accelerate artificial intelligence (AI) and machine learning (ML) applications. This custom instruction is designed to perform four 8-bit integer multiplication operations and accumulate the results into a single output within one instruction execution cycle.

Unlike conventional RV32I implementations, where multiple instructions are required to execute the multiplication, accumulation, and quantization processes separately, the proposed custom instruction combines these operations into a single hardware operation. This integration significantly reduces the total instruction count, minimizes pipeline overhead, and improves overall computational efficiency for quantized neural network inference.

The quantization-aware design ensures that arithmetic operations comply with the scaling and rounding requirements commonly used in INT8 deep learning models. By performing multiplication, accumulation, bias addition, scaling, and quantization within dedicated hardware, the processor can execute AI workloads with lower latency and improved energy efficiency.

The custom DOT4 instruction can be implemented by extending the Execute (EX) stage of the processor through a dedicated hardware accelerator and appropriate modifications to the instruction decoder, ALU control unit, and datapath. The compiler and assembler can also be updated to support the new instruction, enabling software developers to utilize the custom operation through standard programming tools.

Future research will focus on evaluating the proposed instruction using FPGA implementation and comparing its performance with the standard RV32I instruction set. Performance metrics such as execution cycles, throughput, FPGA resource utilization, power consumption, and timing performance will be analyzed. The proposed enhancement is expected to provide a scalable and efficient solution for edge AI, embedded machine learning, digital signal processing, and other compute-intensive applications while maintaining compatibility with the RISC-V ecosystem.

References

[1] A. Waterman and K. Asanovic´, “The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA,” Document Version 20191213, RISC-V International, Dec. 2019.
[2] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*, 2nd ed., Morgan Kaufmann, 2020.
[3] S. Harris and D. Harris, *Digital Design and Computer Architecture: RISC-V Edition*, Morgan Kaufmann, 2021.
[4] IEEE Standard for Verilog Hardware Description Language, *IEEE Std 1364-2005*, IEEE, 2006.