ISSN 2277-2685
IJESR/July-September. 2026/ Vol-16/Issue-3/152-160
M Bhargavi *et. al.*, / International Journal of Engineering & Science Research
Full Length Article

RISC V Processor Using Verilog

Ms Kazi Nikhat Parvin M[1], M Bhargavi[2], Farha Jabeen[3], D Narayani[4]
[1]Associate Professor: Department of Electronics and Communication Engineering, Bhoj Reddy Engineering College for Women Hyderabad, India
[2,3,4]B. Tech Students: Department of Electronics and Communication Engineering, Bhoj Reddy Engineering College for Women Hyderabad, India
Mail Id's: narayanidandala1@gmail.com[4],bhargavim511@gmail.com[2],farhajabeen253@gmail.com[3]
Accepted 27-06-2026

Abstract
*The **RISC-V Processor using Verilog HDL** project focuses on the design and implementation of a simplified processor based on the open-standard RISC-V Instruction Set Architecture (ISA). The primary objective of this work is to understand the internal organization and functionality of a processor by developing a modular hardware design using Verilog Hardware Description Language (HDL). The proposed processor demonstrates the fundamental concepts of instruction execution, data processing, and hardware control while providing practical knowledge of digital system design and processor architecture. The processor is developed using a modular design approach, where each functional unit is implemented independently and then integrated to form the complete processor. The architecture consists of essential components including the Program Counter (PC), Instruction Memory, Control Unit, Register File, Arithmetic Logic Unit (ALU), Data Memory, and multiplexers required for data routing. The processor follows the standard instruction execution cycle comprising instruction fetch, instruction decode, execution, memory access, and write-back stages, ensuring systematic processing of each instruction.*
The implemented processor supports a basic set of RISC-V arithmetic and logical instructions, including addition, subtraction, AND, OR, and XOR operations. Functional verification is carried out through simulation using Verilog HDL simulation tools, where waveform analysis is used to validate the correctness of instruction execution, control signal generation, ALU functionality, register operations, and overall data flow. The simulation results confirm that the processor executes the supported instructions accurately and produces the expected outputs.To simplify the hardware implementation and emphasize the fundamental concepts of processor design, an 8-bit architecture is adopted in this project. Although advanced processor features such as pipelining, cache memory, hazard handling, and branch prediction are not incorporated, the developed processor successfully demonstrates the core principles of the RISC-V architecture. The project provides a solid foundation for learning processor design, computer architecture, and hardware implementation using Verilog HDL, while offering opportunities for future enhancement through the integration of advanced architectural features and performance optimization.

Keywords: *RISC-V Processor, Verilog HDL, Processor Design, Computer Architecture, Instruction Set Architecture (ISA), Arithmetic Logic Unit (ALU), Register File, Digital System Design, Hardware Description Language, Instruction Execution Cycle.*

Introduction

The continuous evolution of digital technology has created an increasing demand for processors that deliver high computational performance while maintaining low power consumption and efficient hardware utilization. Embedded processors are now fundamental components in numerous application domains, including consumer electronics, industrial automation, communication systems, healthcare equipment, automotive control, and Internet of Things (IoT) devices. The processor serves as the central processing unit of these systems, directly influencing their speed, reliability, and overall operational efficiency. Consequently, designing processor architectures that provide improved performance with optimized hardware resources has become an important area of research in computer engineering.

Conventional processor architectures based on the Complex Instruction Set Computer (CISC) philosophy employ a large collection of sophisticated instructions capable of performing multiple operations within a single instruction. Although this approach provides programming flexibility, it also increases processor complexity, hardware requirements, execution latency, and power consumption. These limitations reduce the suitability of CISC processors for modern embedded applications where energy efficiency, compact hardware implementation, and rapid instruction execution are essential design objectives.

To address these challenges, the Reduced Instruction Set Computer (RISC) architecture was introduced as an alternative processing paradigm. RISC processors execute a smaller number of simple instructions that can be completed efficiently within fewer clock cycles. This simplified instruction set reduces hardware complexity, enhances execution speed, and improves processor performance. Among the various RISC architectures, RISC-V has gained widespread recognition because of its open-standard and extensible Instruction Set Architecture (ISA). Unlike proprietary processor architectures, RISC-V enables researchers, academic institutions, and industries to develop customized processor designs without licensing restrictions, thereby promoting innovation and reducing development costs.

The present work focuses on the design and implementation of a RISC-V processor using Verilog Hardware Description Language (HDL). The processor architecture adopts a five-stage pipelined datapath consisting of Instruction Fetch, Instruction Decode, Execute, Memory Access, and Write Back stages. By allowing multiple instructions to occupy different stages of the pipeline simultaneously, pipelining significantly increases instruction throughput and improves processor performance. Since pipelined execution introduces data dependency issues between consecutive instructions, a data forwarding mechanism is incorporated to minimize pipeline stalls and maintain efficient execution.

The processor is modeled, simulated, and verified using the Xilinx Vivado development environment, which provides comprehensive tools for hardware description, functional simulation, and waveform analysis. The implementation offers practical exposure to processor architecture, digital hardware design, pipelining concepts, and Verilog HDL programming while establishing a strong foundation for future development of advanced RISC-V processors incorporating features such as hazard detection, branch prediction, cache memory, and superscalar execution.

Literature Survey

Processor architecture has experienced remarkable advancement over the past several decades, progressing from complex instruction-based designs toward simplified architectures capable of delivering improved computational efficiency. This transition has largely been driven by the Reduced Instruction Set Computer (RISC) philosophy, which emphasizes simplified instruction execution, efficient hardware implementation, and higher processing speed. The evolution of RISC architectures has significantly influenced modern processor design, with RISC-V emerging as one of the most influential open-standard Instruction Set Architectures in both academic research and industrial applications.

The RISC-V architecture was originally developed at the University of California, Berkeley, with the objective of providing an open, modular, and extensible ISA suitable for education, research, and commercial processor development. Unlike proprietary architectures such as ARM and x86, RISC-V can be implemented without licensing fees, enabling processor designers to modify and extend the instruction set according to application-specific requirements. This flexibility has accelerated research activities and encouraged the development of customized processors across a wide range of embedded and high-performance computing applications.

Previous research demonstrates that RISC-V adopts a load-store architecture in which memory operations are restricted to dedicated load and store instructions, while arithmetic and logical computations are performed exclusively using processor registers. This architectural organization simplifies datapath implementation, reduces hardware complexity, and contributes to improved execution efficiency. The separation of memory access from computational operations enables efficient instruction scheduling and facilitates higher processor performance.

Several published studies have investigated pipelined RISC-V processor implementations to improve instruction throughput and hardware utilization. Pipeline architectures divide instruction execution into multiple sequential stages, allowing

different instructions to be processed concurrently. This overlapping execution significantly increases processor performance compared with single-cycle or multi-cycle implementations. However, pipelined processors introduce challenges including data hazards, control hazards, and structural hazards. To overcome these limitations, researchers have proposed techniques such as data forwarding, hazard detection units, branch prediction mechanisms, and pipeline control logic, resulting in improved execution efficiency and reduced pipeline stalls.

Comparative analyses reported in the literature indicate that RISC-V provides significant advantages over conventional processor architectures due to its modularity, scalability, low implementation cost, and architectural flexibility. The availability of open-source specifications enables extensive experimentation, making RISC-V particularly attractive for academic instruction, embedded system development, and hardware research. Furthermore, its extensible ISA supports the incorporation of specialized instruction extensions for domains such as artificial intelligence, cryptography, digital signal processing, and real-time embedded applications.

Design and Implementation of the RISC-V Processor

Processor architecture forms the foundation of modern digital computing systems and plays a significant role in determining computational efficiency, scalability, and hardware flexibility. Among contemporary processor architectures, the RISC-V Instruction Set Architecture (ISA) has emerged as a widely accepted open-standard platform because of its modular design, extensibility, and royalty-free availability. Unlike proprietary architectures such as ARM and MIPS, RISC-V enables researchers, students, and developers to customize processor functionality without licensing restrictions, making it highly suitable for academic research, embedded system development, and FPGA-based experimentation.

This work presents the design and implementation of an 8-bit RISC-V processor using Verilog Hardware Description Language (HDL). The proposed processor adopts a modular hardware architecture in which each functional unit is independently designed, verified, and integrated into the complete processor. Such a methodology improves design flexibility, simplifies debugging, and allows individual modules to be reused or extended for future processor implementations. The processor architecture incorporates essential functional blocks, including the Program Counter, Instruction Memory, Register File, Control Unit, Arithmetic Logic Unit (ALU), Data Memory, and pipeline support components required for instruction execution.

The design methodology is supported by a detailed study of existing processor architectures, highlighting their strengths and limitations while demonstrating the advantages of adopting an open-source RISC-V platform. The implementation is verified through simulation using industry-standard Electronic Design Automation (EDA) tools, ensuring correct functionality before hardware realization. Table 2.1 summarizes the technologies and development tools considered during the implementation process, including RISC architecture, CISC architecture, RISC-V ISA, pipelining concepts, Verilog HDL, and Xilinx Vivado.

Existing Processor Architectures and Motivation

Several processor architectures have been extensively employed in both industrial and academic environments, including MIPS, ARM, and traditional microprocessors such as the Intel 8085 and 8086. Although these processors have significantly contributed to computer architecture development, they exhibit limitations when used for educational processor implementation and hardware experimentation.

The MIPS architecture provides a simple Reduced Instruction Set Computing (RISC) model; however, its proprietary nature and fixed instruction set limit customization opportunities for students and researchers. Similarly, ARM processors deliver excellent computational performance and power efficiency but restrict direct architectural modification because of commercial licensing requirements. Legacy processors such as the 8085 and 8086 remain valuable for understanding fundamental microprocessor concepts but lack the scalability, flexibility, and architectural features expected in modern embedded computing platforms. These limitations create the need for an open and customizable processor architecture that supports hardware experimentation while minimizing implementation complexity. The RISC-V Instruction Set Architecture satisfies these requirements by providing an extensible, modular, and royalty-free design that enables complete architectural customization. Consequently, RISC-V

has become an ideal choice for processor development, digital design education, FPGA implementation, and computer architecture research.

Proposed System

The proposed system consists of an 8-bit RISC-V processor implemented entirely in Verilog HDL using a modular hardware design methodology. Individual hardware components, including the Program Counter, Register File, Instruction Memory, Data Memory, Control Unit, and Arithmetic Logic Unit, are designed independently before being integrated into the complete processor architecture. This modular implementation simplifies testing, debugging, verification, and future system expansion.

The processor supports a subset of RISC-V instructions required for demonstrating fundamental processor functionality. Arithmetic instructions such as ADD and SUB perform numerical computations, while logical operations including AND, OR, and XOR enable bitwise processing. Memory operations are supported through load (LW) and store (SW) instructions, whereas branch instructions such as BEQ and BNE facilitate conditional program execution. Functional verification is performed using ModelSim and Xilinx Vivado simulation environments, ensuring that each instruction executes correctly before hardware implementation. The processor follows the conventional fetch-decode-execute cycle. During execution, the Program Counter supplies the instruction address to the Instruction Memory, where the corresponding instruction is retrieved. The Control Unit decodes the instruction and generates the required control signals to coordinate data movement between the Register File, ALU, and Data Memory. Following execution, the computed result is either stored back in the destination register or written to memory, depending on the instruction type. Owing to its modular design, the processor can be extended to 16-bit or 32-bit architectures with minimal structural modifications.

Functional Architecture

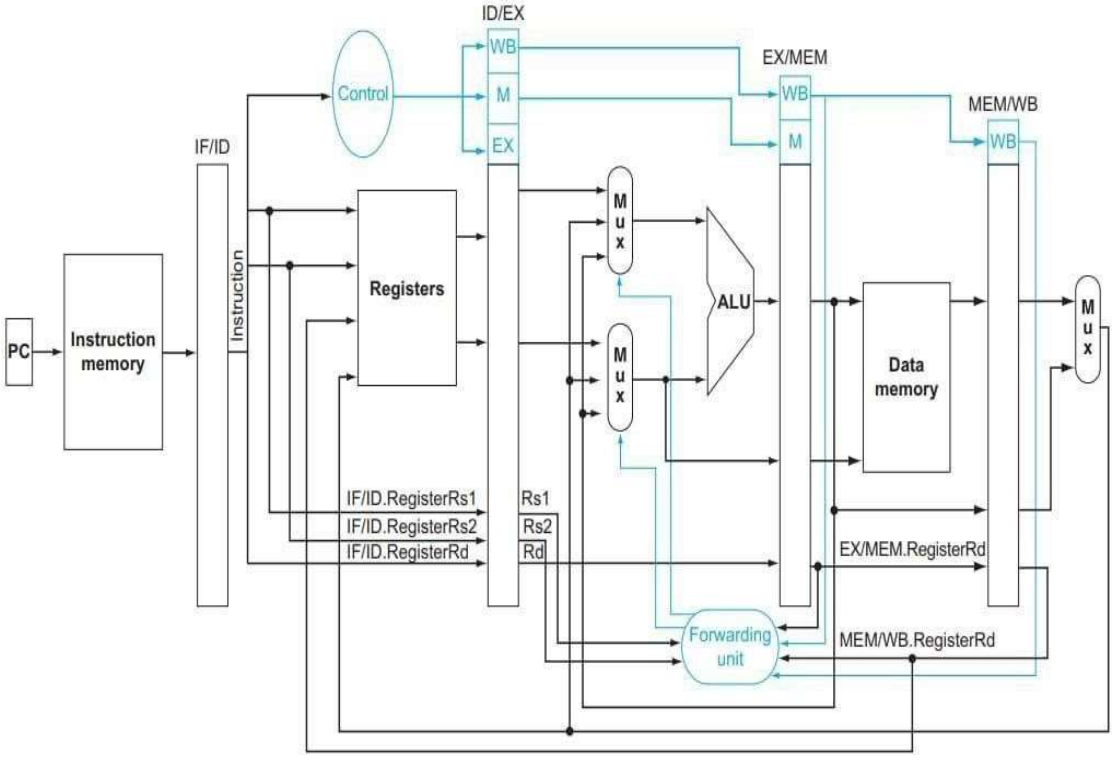


Figure 1 Functional Diagram of RISC V Processor

The proposed pipelined RISC-V processor consists of multiple interconnected functional units that collectively execute program instructions in an organized sequence. Each module performs a specialized task while communicating with neighboring components through dedicated control and data paths.

The Program Counter serves as the starting point of processor execution by maintaining the address of the next instruction. During normal operation, the Program Counter increments after every instruction

fetch, whereas branch and jump instructions update its value according to the target address, thereby controlling program flow.

Instruction Memory stores the executable program in binary format and delivers the appropriate instruction corresponding to the Program Counter address. The fetched instruction is forwarded to the decoding stage for interpretation by the Control Unit.

The Register File functions as high-speed internal storage by maintaining processor operands and intermediate computation results. It supports simultaneous read and write operations, enabling efficient data exchange during instruction execution.

The Control Unit interprets the opcode contained within each instruction and generates the control signals required to coordinate processor operation. These signals determine ALU functionality, register write operations, memory access, and branch decisions.

The Arithmetic Logic Unit performs arithmetic and logical computations required during instruction execution. Depending on the generated control signals, the ALU executes operations including addition, subtraction, logical AND, logical OR, exclusive OR, and comparison functions before forwarding the result to the subsequent stage.

Data Memory provides storage for load and store instructions. Memory read and write operations are controlled by signals generated by the Control Unit, while addresses are calculated by the ALU during execution.

Pipeline registers, including IF/ID, ID/EX, EX/MEM, and MEM/WB, temporarily store data and control information between consecutive pipeline stages. These registers enable overlapping instruction execution and improve processor throughput by allowing multiple instructions to be processed simultaneously.

To minimize performance degradation caused by data dependencies, the processor incorporates a Forwarding Unit. This hardware block detects data hazards and forwards results directly from later execution stages to earlier pipeline stages whenever possible, thereby reducing pipeline stalls and improving execution efficiency.

The complete interaction among these modules is illustrated in the functional block diagram of the proposed RISC-V processor.

Methodology

The implementation methodology adopted in this work follows a systematic hardware development approach beginning with instruction set analysis and concluding with complete functional verification. Initially, an appropriate subset of RISC-V instructions was selected for implementation within an 8-bit processor architecture. The instruction format was carefully defined by specifying opcode fields, source and destination registers, and immediate operands while ensuring adequate support for arithmetic, logical, memory, and branch operations.

Following instruction analysis, each processor module was independently developed using Verilog HDL. The Arithmetic Logic Unit was designed to perform arithmetic and logical computations, while the Register File was implemented to provide reliable operand storage and retrieval. The Control Unit was developed to generate appropriate control signals for every supported instruction, and the Program Counter was implemented to manage sequential and branch-based instruction execution.

After successful module-level verification, all functional blocks were integrated through common buses and synchronized control signals within a top-level processor module. Particular attention was given to maintaining proper timing relationships so that instruction fetch, decode, execute, memory access, and write-back operations occurred in the correct sequence without functional conflicts.

The completed processor was then verified through simulation using comprehensive Verilog test benches. Various arithmetic, logical, memory access, and branch instructions were executed to validate processor functionality. Simulation waveforms were carefully analyzed to verify signal timing, instruction execution, data movement, and control signal generation. Any functional inconsistencies identified during simulation were corrected through iterative debugging until the processor achieved reliable and error-free operation.

Software Requirements

The successful implementation of the proposed RISC-V processor requires an integrated software development environment capable of supporting hardware description, functional simulation, synthesis, and verification. Industry-standard Electronic Design Automation (EDA) tools were employed throughout the development process to ensure accurate hardware implementation and performance evaluation.

Verilog Hardware Description Language serves as the primary design language for describing the

processor architecture. Verilog enables both behavioral and structural modeling of digital hardware, allowing designers to implement, simulate, and verify processor functionality before hardware realization. In this work, Verilog was used to develop every processor component, including the Program Counter, Register File, Arithmetic Logic Unit, Control Unit, Data Memory, and pipeline registers.

Xilinx Vivado was selected as the primary design environment because of its comprehensive support for hardware development, simulation, synthesis, and implementation. The software provides facilities for compiling Verilog modules, generating simulation waveforms, debugging hardware logic, analyzing timing behavior, and preparing the processor for FPGA deployment. Functional verification performed within Vivado ensures that the processor operates according to the intended architecture before physical implementation.

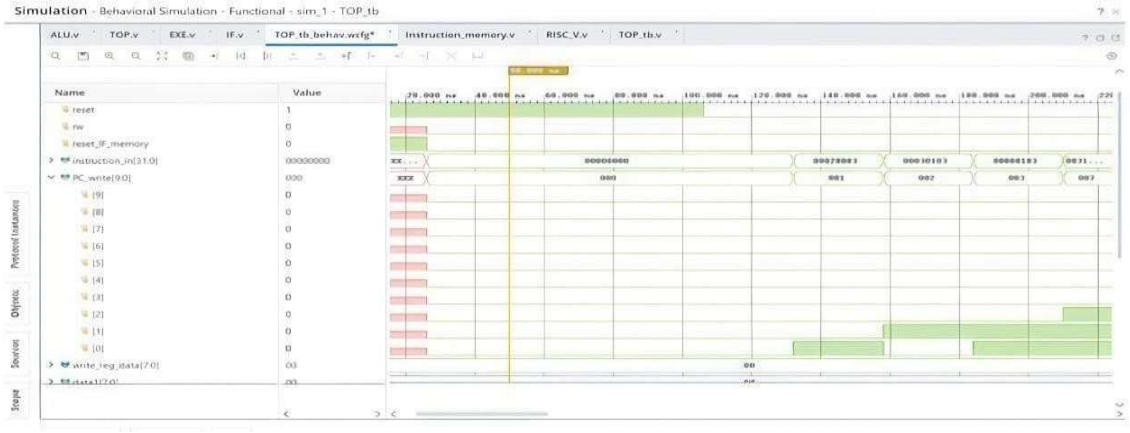
The development tools operate efficiently under both Windows and Linux operating systems, providing a stable environment for project management, simulation, and hardware design activities. The operating system facilitates software execution, file management, and interaction with the development tools throughout the processor implementation process.

Simulation plays a critical role in validating processor functionality prior to hardware synthesis. The integrated simulation environment available within Vivado enables comprehensive verification of instruction execution, timing behavior, control signal generation, and data transfer among processor modules. Waveform analysis provides a detailed visualization of processor operation across multiple clock cycles, allowing design errors, timing violations, and logical inconsistencies to be identified and corrected before implementation on FPGA hardware. Consequently, simulation significantly improves design reliability while reducing hardware debugging effort.

Results

The proposed 8-bit RISC-V processor was successfully designed and functionally verified using Verilog HDL in the Xilinx Vivado/ModelSim simulation environment. Functional validation was performed by executing a sequence of arithmetic, logical, memory access, and control instructions while monitoring the generated waveforms. The simulation results confirmed the correct interaction among the major processor modules, including the Program Counter (PC), Instruction Memory, Control Unit, Register File, Arithmetic Logic Unit (ALU), and Data Memory. The observed timing diagrams demonstrated proper synchronization of signals, accurate data movement, and reliable execution of the instruction cycle, indicating that the processor architecture operates according to the intended design.

Program Counter Operation



Simulation of the Program Counter verified that it correctly controls the instruction execution sequence. During system reset, the PC was initialized to zero, providing a defined starting address for program execution. After the reset signal was deactivated, the Program Counter incremented sequentially on every clock cycle to fetch consecutive instructions. When branch instructions were executed, the PC successfully updated to the specified target address instead of continuing sequential execution. These observations confirm that the Program Counter accurately supports both sequential instruction execution and conditional program flow.

Instruction Fetch and Decode

The instruction fetch and decode stages were validated by observing the instruction memory output and the generated control signals. Based on the address supplied by the Program Counter, the instruction memory consistently delivered the correct instruction to the decoder. The Control Unit successfully interpreted the opcode of each instruction and generated the corresponding control signals required for arithmetic operations, memory access, and branching. The simulation demonstrates that the fetch and decode stages operate reliably and provide accurate control information for subsequent execution stages.

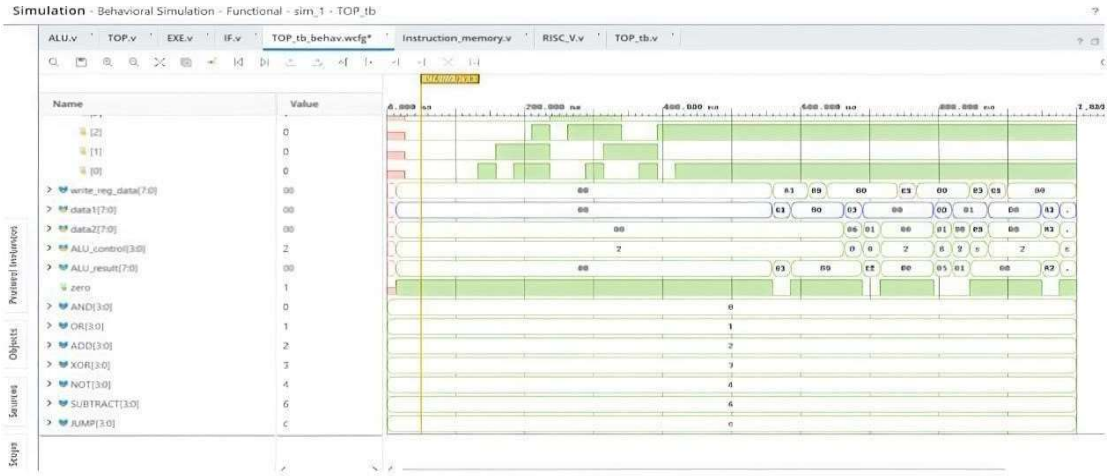
Arithmetic Logic Unit Performance

The functionality of the Arithmetic Logic Unit was verified by executing various arithmetic and logical instructions. Operations including addition, subtraction, AND, OR, XOR, and NOT produced the expected output values for different combinations of input operands. The ALU control signals correctly selected the required operation, while the zero flag was asserted whenever the computation resulted in a zero value. The simulation results confirm that the ALU performs all supported operations correctly and provides reliable outputs for instruction execution.

Register File Verification

The Register File was evaluated through multiple read and write operations during processor execution. The simulation confirmed that data values were accurately written into the destination registers and subsequently retrieved without corruption. The write-back stage successfully transferred ALU and memory results into the appropriate registers, maintaining data consistency throughout execution. These observations demonstrate that the Register File provides dependable storage for operands and intermediate computation results.

Datapath Verification



The processor datapath was analyzed to verify communication between all functional modules. Simulation waveforms showed that operand values were transferred correctly from the Register File to the ALU, while ALU outputs were appropriately routed either to the Data Memory or back to the Register File depending on the instruction type. The generated control signals consistently selected the correct data paths throughout execution, confirming the proper integration of the processor components.

Waveform Analysis

Waveform analysis served as the primary method for functional verification of the processor. The simulation results showed that all control and data signals remained synchronized with the system clock, ensuring correct timing relationships among the pipeline stages. Signal transitions occurred without unexpected glitches or timing violations, indicating stable processor operation. The waveforms also verified accurate instruction execution, memory access, register updates, and control signal generation throughout the simulation process.

Discussion

The experimental evaluation demonstrates that the proposed RISC-V processor satisfies the functional objectives established during the design phase. The modular architecture adopted in this work simplified implementation, testing, and verification by allowing each hardware block to be independently validated before system integration. This design methodology also improved maintainability and facilitated debugging during simulation.

The functional verification process confirmed that the ALU, Register File, Control Unit, Program Counter, Instruction Memory, and Data Memory operate correctly both as individual modules and as an integrated processor. The successful execution of arithmetic, logical, load/store, and branch instructions indicates that the processor architecture accurately implements the intended instruction execution cycle.

Performance analysis revealed that the processor executes instructions correctly using a straightforward execution mechanism. The simplified instruction set contributes to reduced hardware complexity and faster control logic, making the processor suitable for educational applications and introductory processor design studies. Although the implemented architecture provides reliable functionality, the absence of advanced pipelining and hazard management limits the achievable instruction throughput compared with modern commercial processors.

Reliability was further demonstrated through repeated simulation experiments, where all generated outputs matched the expected results without timing inconsistencies or functional failures. Stable signal transitions observed in the waveform analysis confirm that the processor maintains consistent operation under different instruction sequences.

From a design perspective, the modular implementation in Verilog HDL improves scalability and simplifies future hardware modifications. Individual modules can be extended or replaced with enhanced versions without requiring significant architectural changes, making the processor an appropriate platform for research and academic experimentation.

Despite these positive results, certain limitations remain. The processor is currently restricted to an 8-bit datapath and implements only a limited subset of the RISC-V instruction set architecture. Furthermore, advanced architectural features such as cache memory, interrupt handling, hazard detection, forwarding logic, branch prediction, and superscalar execution are not included in the present implementation. Consequently, the processor is primarily intended for educational demonstrations rather than high-performance computing applications.

Future enhancements should focus on extending the datapath to 32 bits in accordance with the standard RISC-V specification and implementing the complete instruction set architecture. Additional improvements may include introducing pipeline hazard detection and forwarding mechanisms, branch prediction techniques, cache memory, interrupt support, and FPGA-based hardware implementation for real-time validation. Power optimization and area-efficient design strategies may also be investigated to improve suitability for embedded and low-power applications. These enhancements would significantly increase processor performance while preserving the modular design philosophy adopted in this work.

Overall, the presented processor successfully demonstrates the practical implementation of a RISC-V architecture using Verilog HDL. The simulation results verify the correctness of the design, while the modular organization provides a solid foundation for future research in computer architecture, embedded systems, and FPGA-based processor development.

Conclusion

This research presented the successful design, implementation, and functional verification of a simplified pipelined RISC-V processor using Verilog Hardware Description Language. The processor architecture incorporates essential components, including the Program Counter, Instruction Memory, Register File, Control Unit, Arithmetic Logic Unit, Data Memory, and pipeline registers, which collectively execute instructions through the standard fetch, decode, execute, memory access, and write-back stages. Functional verification performed using ModelSim/Xilinx Vivado confirmed correct instruction execution, accurate control signal generation, reliable data movement, and proper synchronization between hardware modules. The obtained simulation waveforms demonstrate stable processor operation without functional errors, validating the effectiveness of the proposed design. The developed processor provides an efficient educational platform for understanding processor architecture, digital system design, and hardware implementation using Verilog HDL while establishing a foundation for more advanced RISC-V processor development.

Future Scope

The proposed processor architecture provides several opportunities for future enhancement. The current 8-bit implementation can be extended to a standard 32-bit RISC-V architecture to support more

complex applications and improve computational capability. Future work may also include implementation of the complete RISC-V instruction set, advanced pipeline hazard detection and forwarding mechanisms, branch prediction techniques, interrupt handling, and cache memory integration to improve execution efficiency. Hardware validation using FPGA development boards will enable real-time performance analysis under practical operating conditions. Additional optimization of hardware resources, power consumption, and operating frequency can further improve processor efficiency for embedded applications. Furthermore, the modular architecture developed in this work can serve as a foundation for designing multicore processors, domain-specific accelerators, superscalar architectures, and real-time embedded processing systems, thereby extending its applicability to modern computer architecture research and industrial applications.

References

[1] H. Li, C. Jing, and J. Liu, "Performance-Optimized Design of a RISC-V Five-Stage Pipelined Processor," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 2, pp. 276–285, 2024.

[2] F. Hussain and S. Sarkar, "Design and FPGA Implementation of a Five-Stage Pipelined RISC-V Processor," in *Proceedings of the IEEE 9th International Conference on Convergence in Technology (I2CT)*, 2024.

[3] A. N. Ben Yousuf, D. K. Alamen, and M. M. Eljhani, "Design and Implementation of a Five-Stage Pipelined RISC Processor on FPGA," in *Proceedings of the IEEE International Conference on Automatic Control and Computer Engineering*, 2023.

[4] P. Saiprathyusha and C. Chandrasekhar, "Implementation of a RISC-V Processor," *ITM Web of Conferences*, vol. 74, 2025.

[5] Y. Li, Z. Yu, and W. Ji, "Design and Performance Optimization of a RISC-V Processor with a Five-Stage Pipeline," in *Proceedings of the IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, 2025.