IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 1, January 2025

DOI: 10.17148/IJARCCE.2025.14143

# Implementation of RISC-V Single Cycle Core

**Anjana K M[1], Anusha S B[2], Dikshitha U[3], Mrs. Shilpa V[4]**

Students, Electronics and Communication Engineering, SJB Institute of Technology, Bengaluru,India[1,2,3]

Assistant Professor, Electronics and Communication Engineering, SJB Institute of Technology, Bengaluru, India[4]

**Abstract:** This research paper presents the design and implementation of a 32-bit single-cycle RISC-V (RV32I) processor using Verilog HDL, targeting FPGA-based deployment for educational and embedded system applications. RISC-V, an open-standard instruction set architecture (ISA), provides an alternative to proprietary architectures by offering flexibility, scalability, and ease of implementation. The processor is structured around the fundamental stages of instruction execution, including instruction fetch, decode, execute, memory access, and write-back. To enhance performance, a five-stage pipelining approach is incorporated, reducing execution time per instruction while maintaining design simplicity. Key modules implemented in the processor architecture include the program counter, instruction memory, register file, arithmetic logic unit (ALU), data memory, pipeline registers, multiplexers, and a hazard detection unit to address data and control hazards. The entire design is synthesized and validated using FPGA platforms such as Xilinx Spartan-6 and Spartan-3E, with functional verification conducted through Xilinx ISE and Vivado simulation tools. The processor achieves a maximum operating frequency of 32 MHz, with an estimated power consumption of 7.9 mW, as analyzed using the Xilinx Power Analyzer. The implementation demonstrates the feasibility of a low-cost, fully synthesizable RISC-V processor, offering an efficient and scalable solution for embedded system applications. Furthermore, the development framework includes assembling tools and automated test suites to validate the processor's functionality. This work contributes to the advancement of open-source processor design, providing insights into the hardware realization of RISC-V and establishing a foundation for future research into more complex architectures.

**Keywords:** RISC-V, Instruction Set Architecture , Verilog, Simulation.

## I. INTRODUCTION

The rapid evolution of computer hardware has significantly shaped the landscape of modern computing, with x86 and ARM architectures dominating the semiconductor industry. These processors serve as the backbone of various computing applications, ranging from general-purpose computing in personal computers and mobile devices to embedded systems in consumer electronics, industrial automation, and IoT devices. At the core of a processor's functionality lies the ability to execute a series of operations based on instructions. This concept stems from the Turing Machine, where a processor systematically fetches, decodes, and executes instructions to perform computational tasks. Processors are generally categorized into two broad types: general-purpose processors and dedicated processors. General-purpose processors, such as those found in personal computers and mobile phones, provide versatility through software programmability, enabling them to run a diverse set of applications. In contrast, dedicated processors are optimized for specific tasks, prioritizing speed, power efficiency, and area constraints. Examples of such specialized processors include Graphics Processing Units (GPUs) for rendering graphics and Artificial Intelligence (AI) accelerators for machine learning computations [1].

The microprocessor's operation revolves around fetching instructions from external memory, decoding them, and executing the necessary arithmetic or logical operations. The control unit plays a crucial role in directing the flow of data within the processor, ensuring that operands are correctly routed to the Arithmetic Logic Unit (ALU) and selecting the appropriate function to execute a specific instruction. A conventional microprocessor consists of three key components: the control unit, which manages instruction execution and data flow; the ALU, which performs mathematical and logical operations; and the data path, which facilitates communication between various components. Microprocessors are designed based on one of two fundamental architectures: Harvard or Von Neumann. The Harvard architecture employs separate memory units for data and instructions, allowing parallel data access and improved performance. In contrast, the Von Neumann architecture stores both instructions and data in a single memory unit, utilizing a shared bus for fetching and executing operations, which can sometimes create bottlenecks [1].

This research focuses on the implementation of a 32-bit RISC-V processor using Verilog. RISC-V, an open-source Reduced Instruction Set Computing (RISC) architecture, has gained significant traction due to its modularity, scalability, and simplicity, making it an ideal candidate for both academic research and industrial applications. Unlike traditional Complex Instruction Set Computing (CISC) architectures, which feature a large and intricate set of instructions, RISC

DOI: 10.17148/IJARCCE.2025.14143

architectures emphasize simplicity and efficiency, using a fixed instruction format and optimizing performance through pipelining and parallel execution. The single-cycle RISC-V processor implemented in this research executes each instruction in a single clock cycle, simplifying control logic but requiring a higher clock speed for increased performance. The design incorporates essential components, including the instruction fetch unit, decode unit, ALU, register file, memory access unit, and write-back stage, all of which are simulated and verified using Verilog HDL (Hardware Description Language) [1].

## II. LITERATURE SURVEY

This research work designed and implemented a 32-bit single-cycle RISC-V processor using Verilog, executing operations like addition, subtraction, logical OR, AND, XOR, and SLT. Rigorous testing confirmed its functionality, validating components such as the Program Counter, Instruction Memory, Register File, and ALU. The research work demonstrated a solid grasp of RISC-V architecture and Verilog, paving the way for future improvements. Planned enhancements include a pipelined architecture for better performance, expanded instruction support for versatility, optimization techniques to reduce power consumption, and interrupt handling for real-time applications. Future work of this project is to implement multi-cycle RISC-V processor with pipelined structure. This can be implemented by using NOP operation or by using the concept of data forwarding. These upgrades will make the processor more powerful and versatile [1].

The complexity of modern microprocessor designs presents significant challenges in their implementation and usage. This paper details the development of a novel single-cycle tiny embedded processor, designed to be modular and highly extensible. The processor utilizes 16KB of on-chip block RAM for data memory and 64KB for instruction memory, achieving a peak clock speed of 32MHz while consuming 7.9mW of power on a Spartan 3EXC3S500E FPGA. Leveraging the expanding RISC-V ecosystem, existing toolchains, and software support, this design serves as a foundation for future implementations in IoT and embedded applications. Planned enhancements include I/O bus integration, multi-level caching, interrupt handling, and floating-point co-processor integration, with potential applications in education, research, and industry, driving faster and more efficient processor designs [2].

The primary objective is to create a 32-bit pipelined processor based on the open- source RV32I Version 2.0 ISA, RISC-V, with several modules. A processor known as a RISC (Reduced Instruction Set Computer) requires less hardware than a CISC (Complex Instruction Set Computer) in order to reduce the complexity of the instruction set and accelerate the execution time of each instruction. With the help of the required block diagrams, we also built this processor with five levels of pipelining, each of which has a detailed description of its operation. This project uses Verilog to develop and simulate a RISC-V. Because of its free and open instruction set architecture (ISA), the RISC-V processor design offers computer designers an alternative for both software and hardware creation. In addition, the five-stage pipeline procedures that the proposed RISC-V processor will employ will enhance the machine's overall performance. The pipeline register (IF/ID, ID/IEx, IEx/IMem, and IMem/IW), alu, aludec, maindec, imem, dmem, regfile, pc_mux, result_mux, and pipeline register are the first main modules to be implemented in the project. In addition, a hazard unit is incorporated into the design to lessen hazardous situations. Xilinx Vivado was used to simulate and validate these modules functioning [3].

This paper presents the design and implementation of a single cycle RISC-V RV32I processor on FPGA using Xilinx ISE Design Suite. RISC-V is an open standard instruction set architecture that provides flexibility, scalability and privilege from proprietary constraints, making it an excellent choice for educational purposes. The processor is designed using Verilog HDL, includes essential components such as the instruction fetch, decode, execute, memory access and write-back stages. The entire design is synthesized and implemented on a Spartan-6 FPGA board. This project demonstrates the feasibility and effectiveness of implementing a single cycle RISC-V processor on FPGA, providing valuable insights into processor design and hardware implementation [4].

## III. METHODOLOGY

1. **Study the RISC-V ISA:** The first step involves understanding the RISC-V RV32I instruction set architecture,

  

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 1, January 2025
DOI: 10.17148/IJARCCE.2025.14143

2. including the different instruction types such as R-type, I-type, S-type, B-type, U-type, and J-type. A detailed analysis of each instruction's operation, encoding, and role in the data path is conducted. Additionally, key ISA features like the number of general-purpose registers and immediate value formats are identified.

**2. Define Objectives:** Clear design goals are established, emphasizing single-cycle execution, simplicity for educational use, and compliance with the RISC-V ISA. Evaluation metrics such as resource usage, clock cycle time, and correctness are also defined to assess performance.

**3. Design Processor Architecture:** A high-level block diagram is created, incorporating essential components such as the control unit for instruction decoding and control signal generation, the ALU for arithmetic and logical operations, a register file with 32 general-purpose registers, instruction memory for fetching instructions, data memory for load/store operations, a program counter for tracking the next instruction, multiplexers for routing data, and adders for address calculations. The design focuses on a single-cycle implementation to simplify timing and control while ensuring optimal coordination between components to minimize pipeline stalls and delays.

**4. Design the Data Path:** A single-cycle data path is developed, integrating key execution stages:

- Instruction Fetch (IF): Retrieves instructions from instruction memory.
- Instruction Decode (ID): Decodes the instruction and generates control signals.
- Execution (EX): Performs arithmetic/logic operations or calculates memory addresses using the ALU.
- Memory Access (MEM): Reads or writes data from/to memory as required.
- Write-Back (WB): Writes computation results back to the register file.

**5. Implement Control Logic:** The control unit is designed to generate appropriate control signals for the ALU, register file, memory, and other components based on instruction type and opcode. A truth table or logic equations are developed to define control signal generation accurately.

**6. Write Verilog Modules:** Verilog modules are developed for each processor component:

- ALU Module: Implements arithmetic and logical operations.
- Control Unit Module: Decodes instructions and generates control signals.
- Register File Module: Provides read and write access to registers.
- Memory Modules: Simulates instruction and data memory. The Verilog code is written to be reusable and parameterized for easier testing and integration.

**7. Integrate Modules:** The individual Verilog modules are integrated into a complete processor design, ensuring proper connections between components to establish a functional data path. The ALU, register file, memory, control unit, and data path are systematically tested to ensure all control signals and data paths function correctly.

**8. Create Testbenches:** Testbenches are developed in Verilog to verify the functionality of each module separately. A comprehensive testbench is also created to test the entire processor, ensuring it correctly executes all RISC-V RV32I instructions.

**9. Simulate the Design:** Simulation tools are used to test the processor with various instruction sequences. The outputs are compared with expected results to verify correct functionality. Any errors encountered are debugged, and refinements are made to improve the design.

**10. Analyze Performance:** The final step involves evaluating the processor's performance based on execution speed, resource utilization, and area requirements. This ensures the design is optimized for efficiency, functionality, and compliance with the RISC-V ISA.

  

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 1, January 2025

DOI: 10.17148/IJARCCE.2025.14143

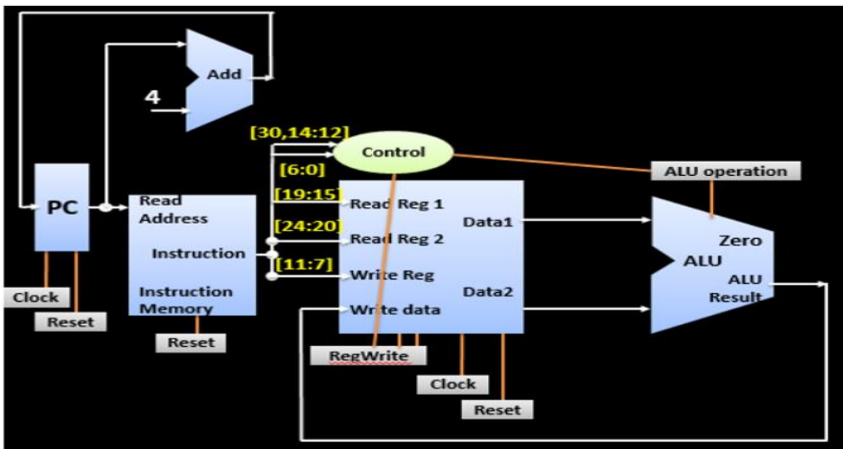


Fig.1 Block diagram of RISC-V Single Cycle Core

The key components of a RISC-V single-cycle core include the Program Counter (PC), which holds the address of the current instruction and updates every clock cycle. Instruction Memory stores program instructions and uses the PC to fetch the current instruction. The Control Unit decodes instructions, generates control signals, and determines data flow based on opcode and function codes. The Register File consists of 32 registers for temporary data storage, with specific inputs for reading and writing data and outputs for retrieved values. The Arithmetic Logic Unit (ALU) performs arithmetic and logical operations based on control signals, providing an operation result and a zero flag for branching. An Adder calculates the next instruction address by adding 4 to the PC. Multiplexers select between multiple inputs based on control signals, such as choosing between immediate values and register data for ALU operations. Instruction Fields contain specific bit segments that define function codes, opcodes, and register addresses. Finally, Clock and Reset Signals synchronize processor operations and initialize components to a known state.

## IV. RESULTS

The GTKWave displays a waveform simulation of a RISC-V single-cycle processor, showing key signals over time. The clock signal is running continuously, synchronizing all operations, while the reset signal appears to be inactive. The instruction_code[31:0] signal represents the binary instructions being fetched and executed sequentially, with corresponding changes in the PC[31:0] value, indicating the program counter incrementing correctly (typically by 4 for each instruction). The alu_control[3:0] signal defines the ALU operations, changing according to the instruction type.

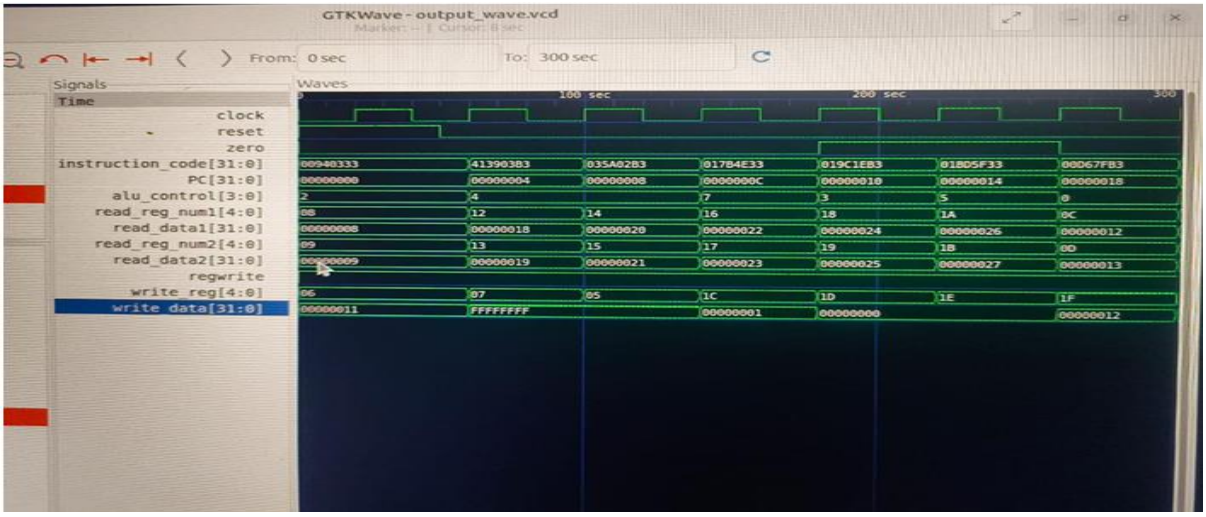


Fig.2 Output waveform using GKT Wave

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 1, January 2025
DOI: 10.17148/IJARCCE.2025.14143

The read_reg_num1[4:0] and read_reg_num2[4:0] signals specify the source registers being accessed, while read_data1[31:0] and read_data2[31:0] display the values fetched from these registers. These values match expected data for corresponding instructions, ensuring correct data flow. The write_reg[4:0] and write_data[31:0] signals show the destination register and the computed value being written back, demonstrating proper execution of write-back operations.

The zero signal, used in branch instructions, toggles as needed, indicating successful comparisons. The transitions between different values and correct data propagation confirm that the processor is executing instructions as intended, following the expected RISC-V pipeline stages. This simulation verifies functional correctness and helps in debugging any discrepancies in execution.

## V. CONCLUSION

This research work designed and implemented a 32-bit single-cycle RISC-V processor using Verilog, executing operations like addition, subtraction, logical OR, AND, XOR, and SLT. Rigorous testing confirmed its functionality, validating components such as the Program Counter, Instruction Memory, Register File, and ALU. The research work demonstrated a solid grasp of RISC-V architecture and Verilog, paving the way for future improvements. Planned enhancements include a pipelined architecture for better performance, expanded instruction support for versatility, optimization techniques to reduce power consumption, and interrupt handling for real-time applications. Future work of this project is to implement multi-cycle RISC-V processor with pipelined structure. This can be implemented by using NOP operation or by using the concept of data forwarding. These upgrades will make the processor more powerful and versatile.

## REFERENCES

[1] Manjusha Rao P, Prabha Niranjan, Dileep Kumar M J (2024) : Design and Implementation of 32-bit RISC-V Processor using Verilog.
[2] Baggu Maneesha, Gudla Sriram, Ganteti Sai,Murapa kaSiddhu,S mt.E.Jaya (2024) : Design of RISC-V processor using Verilog.
[3] Mr. Praveen A, Shwetha V, Thushar Cherian, Prayag Singh, Varshith (2024) : RISC-V Microarchitecture Design on FPGA .
[4] A. Waterman, Y. Lee, D. A. Patterson, K. Asanovic, V. I. U. level Isa, A. Waterman, Y. Lee, and D. Patterson, “The risc-v instruction set manual,” 2014.
[5] Poli, L., Saha, S., Zhai, X., & Mcdonald-Maier, K. (2021). Design and Implementation of a RISC V Processor on FPGA. 2021 17th International Conference on Mobility, Sensing and Networking (MSN), 161-166.
[6] Thakor, K., Pal, A., & Shirodkar, M. (2017). Design of a 16-bit RISC Processor Using VHDL. International Journal of Engineering Research.
[7] Barriga, A. (2020). RISC-V processors design: a methodology for cores development. 2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS), 1-6.
[8] Harmina, T., Hofman, D., & Benjak, J. (2023). DCT Implementation on a Custom FPGA RISC-V Processor. 2023 International Symposium ELMAR, 163-167.
[9] Akshay Birari et al., “A RISC-V ISA Compatible Processor IP,” 24th International Symposium on VLSI Design and Test, pp. 1-6, 2020.