Implementation of MIPS 8-BIT Single Cycle Processor Using Verilog HD

M Sai Krishna[1], Bai Harikumar[2], G Varun Satya[3], G Rajashekar[4], G Hari Prasad[5], G Vamshi Goud[6]
[1]*Assistant Professor, TKR College Of Engineering and Technology*
[2,3,4,5,6]*Student, TKR College Of Engineering and Technology*

***Abstract*—This research presents the design and implementation of an 8-bit single-cycle MIPS processor using Verilog HDL. The primary objective is to develop a simplified processor architecture that enables clear understanding of instruction execution within a single clock cycle. The system integrates essential components such as Program Counter, Instruction Memory, Register File, Arithmetic Logic Unit, Data Memory, and Control Unit. Each instruction undergoes sequential stages including fetch, decode, execute, memory access, and write-back within one cycle, ensuring structural simplicity. Unlike conventional 32-bit processors, the proposed 8-bit design minimizes hardware complexity while retaining core architectural principles. The processor supports arithmetic, logical, memory, and control flow operations, making it suitable for educational and FPGA-based applications. Simulation results validate correct functionality of instruction execution and control signal behavior. This work provides a practical platform for understanding processor design fundamentals and serves as a foundation for future enhancements such as pipelining and performance optimization.**

***Index Terms*—MIPS Processor, Verilog HDL, Single Cycle Architecture, FPGA Design, 8-bit Processor, RISC Architecture.**

I. INTRODUCTION

The design and understanding of processor architecture form a fundamental part of digital electronics and computer engineering. Processors act as the core component of any computing system, executing instructions and managing data flow between hardware units. However, modern processors have become increasingly complex due to advanced features such as pipelining, cache memory, superscalar execution, and parallel processing. While these advancements improve performance, they also make it difficult for students and beginners to clearly understand the basic working principles of a processor. As a result, there is a strong need for simplified processor models that focus on fundamental concepts without introducing unnecessary complexity. To address this challenge, this work presents the design and implementation of an 8-bit MIPS single- cycle processor using Verilog Hardware Description Language (HDL). The MIPS architecture, based on Reduced Instruction Set Computing (RISC), is widely used in academic learning due to its simple and clean instruction format. In a single-cycle processor, each instruction is completed in one clock cycle, passing through all stages of execution including instruction fetch, decode, execute, memory access, and write-back. This approach eliminates the need for complex control mechanisms and makes the internal operation of the processor more transparent and easier to analyze.

The proposed processor includes all essential components required for instruction execution, such as the Program Counter, Instruction Memory, Register File, Arithmetic Logic Unit (ALU), Data Memory, Control Unit, multiplexers, and adders. Each module is designed independently and later integrated into a complete system. The processor supports various instruction types, including arithmetic operations, logical operations, memory access instructions, and control flow instructions such as branching and jumping. Despite using an 8-bit architecture, the design retains the key functional characteristics of a standard MIPS processor.

One of the main advantages of this approach is the reduction in hardware complexity, which makes the design suitable for FPGA implementation using tools like Xilinx ISE and Vivado. Simulation and verification are performed to ensure correct

functionality of the processor under different instruction scenarios. This project not only helps in understanding how instructions are executed at the hardware level but also bridges the gap between theoretical concepts and practical implementation.

Overall, the proposed 8-bit MIPS single-cycle processor serves as an effective educational model for learning processor design. It provides a strong foundation for future enhancements such as pipelined architecture, performance optimization, and integration of advanced features, making it highly relevant for academic and research purposes.

II. LITERATURE SURVEY

A. Review of 8-bit Custom MIPS Processor

Previous research on 8-bit MIPS processors focuses on simplifying traditional processor architectures for educational purposes. These designs adopt Reduced Instruction Set Computing (RISC) principles, where a limited number of instructions ensures faster execution and easier hardware implementation. The reviewed work highlights a compact processor with fewer registers and reduced instruction length, typically 16 bits, which simplifies decoding and memory usage. The processor supports basic instruction types such as register, immediate, and jump instructions. Each instruction is executed in a single cycle, making the design straightforward and suitable for learning environments. Key components such as ALU, control unit, and memory modules are implemented using hardware description languages. The main advantage of such systems is reduced hardware complexity while maintaining functional accuracy. However, limitations include lower processing capability and restricted instruction sets compared to full-scale processors.

B. Review of Pipelined 8-bit MIPS Design

Another important area of research involves improving processor performance using pipelining techniques. In these designs, instruction execution is divided into multiple stages such as fetch, decode, execute, memory access, and write-back. This allows multiple instructions to be processed simultaneously, increasing throughput. Compared to single-cycle processors, pipelined architectures significantly reduce execution delay and improve clock frequency. Studies show that pipelined processors achieve better timing performance and efficient resource utilization. However, they introduce additional complexity such as hazard detection and control dependencies. These designs are often implemented using VHDL or Verilog and tested on FPGA platforms. Simulation tools like ModelSim and Xilinx are commonly used to validate functionality. While pipelined processors offer high performance, they are less suitable for beginners due to their complexity. Therefore, single-cycle processors remain important for foundational learning.

C. Review of 32-bit MIPS FPGA Implementation

Research on 32-bit MIPS processors focuses on achieving higher computational power and real-world applicability. These processors use larger data paths, extensive instruction sets, and multiple registers, enabling complex operations. The architecture typically includes advanced features such as sign extension, shifting units, and extensive control logic. Implementation on FPGA platforms demonstrates real-time performance and efficient hardware utilization. These designs support various instruction formats including register, immediate, and jump types. Simulation and synthesis results confirm high-speed operation and reliable execution. However, the major drawback of 32-bit processors is increased hardware complexity and resource consumption. This makes them less suitable for beginner-level projects and low-cost FPGA boards. As a result, simplified versions like 8-bit processors are preferred for academic purposes. They provide a balance between functionality and simplicity while retaining essential architectural concepts.

III. PROPOSED SYSTEM

The proposed system presents the design and implementation of an 8-bit single-cycle MIPS processor using Verilog HDL. The main goal is to develop a simplified yet functionally complete processor that demonstrates the core principles of instruction execution. Unlike complex modern processors, this system focuses on clarity, modularity, and ease of understanding.

The architecture consists of essential components including the Program Counter, Instruction Memory, Register File, Arithmetic Logic Unit (ALU), Data Memory, multiplexers, and Control Unit. Each

component is designed as an independent module and later integrated into a complete processor system. The Program Counter stores the address of the next instruction, while the Instruction Memory provides the corresponding instruction. The Register File stores temporary data required for computation, and the ALU performs arithmetic and logical operations.

The processor follows a single-cycle execution model, where every instruction completes all stages—fetch, decode, execute, memory access, and write-back—within a single clock cycle. This simplifies the control logic and eliminates the need for pipeline management. The Control Unit generates control signals based on the opcode, directing data flow across the system.

The proposed system supports multiple instruction types, including arithmetic operations (ADD, SUB), logical operations (AND, OR), memory operations (LW, SW), and control flow instructions (BEQ, JUMP). The design ensures correct data path flow and proper interaction between modules.

Simulation results verify the correctness of instruction execution and control signal generation. The system is implemented using FPGA-compatible tools, making it suitable for real-time hardware testing. Overall, the proposed design provides a strong foundation for understanding processor architecture and can be extended for advanced features like pipelining and cache integration.

IV. METHODOLOGY

A. System Design Approach

The system design follows a modular approach where each processor component is developed independently and then integrated to form a complete architecture. This method improves clarity, simplifies debugging, and enhances scalability. The design begins with understanding the MIPS architecture and identifying the required modules such as Program Counter, Instruction Memory, Register File, ALU, Data Memory, and Control Unit.

Each module is implemented using Verilog HDL, ensuring proper definition of inputs, outputs, and internal logic. The Program Counter is designed as a sequential circuit that updates its value based on clock signals. Instruction Memory is implemented as a read-only memory structure containing predefined instructions. The Register File is designed with multiple registers supporting simultaneous read and write operations.

The ALU is developed using combinational logic to perform arithmetic and logical operations based on control signals. The Control Unit is designed to decode instructions and generate appropriate control signals for data path operations. Multiplexers and adders are used to control data flow and address calculations.

After designing individual modules, they are integrated into a top-level module where connections between components are established. This ensures smooth data flow across the processor. The design is then verified through simulation using waveform analysis to confirm correct operation of each instruction.

B. Instruction Execution Method

The instruction execution process in the proposed processor follows a structured sequence of operations within a single clock cycle. The process begins with the Program Counter, which provides the address of the next instruction. This address is sent to the Instruction Memory, which retrieves the corresponding instruction.

The fetched instruction is then decoded into different fields such as opcode, register addresses, and immediate values. The Control Unit analyzes the opcode and generates control signals that determine the operation to be performed. These signals guide the data path components during execution.

The Register File reads data from the specified registers, and the ALU performs the required operation such as addition, subtraction, or logical comparison. For memory-related instructions, the ALU calculates the memory address, and the Data Memory performs read or write operations accordingly.

Finally, the result is written back to the destination register if required. This entire process occurs within a single clock cycle, ensuring simplicity in design. The absence of multiple cycles eliminates the need for complex control mechanisms, making the processor easier to understand and implement.

C. Simulation and Verification

Simulation and verification play a crucial role in validating the functionality of the processor design. After implementing the modules in Verilog HDL, a testbench is created to simulate the behavior of the

system under different conditions. The testbench generates clock and reset signals and provides input instructions to the processor.

Simulation tools such as Xilinx ISE or ModelSim are used to observe waveform outputs. These waveforms display the behavior of various signals such as Program Counter updates, ALU outputs, register values, memory access signals, and control signals. By analyzing these waveforms, it becomes possible to verify whether each instruction is executed correctly.

Different test cases are applied to validate arithmetic, logical, memory, and control flow operations. For example, ADD and SUB instructions are tested to confirm ALU functionality, while LW and SW instructions verify memory operations. Branch and jump instructions are tested to ensure proper control flow.

Errors identified during simulation are corrected by modifying the Verilog code and re-running the tests. This iterative process ensures accuracy and reliability of the design. Once verified, the design can be synthesized and implemented on FPGA hardware for real-time testing, confirming practical feasibility.

V. RESULTS AND DISCUSSIONS

Fig 1. Simulation Output Waveform Analysis of 8-bit MIPS Single-Cycle Processor

The displayed waveform represents the simulation results of the implemented 8-bit MIPS single-cycle processor using Verilog HDL. It illustrates the behavior of key signals such as the Program Counter (pc_out), instruction input, ALU result, write data, clock (clk), and reset over time. The waveform confirms that the processor operates correctly according to the single-cycle execution model.

At each clock pulse, the Program Counter increments sequentially, indicating proper instruction fetching from instruction memory. The instruction signal changes accordingly, showing that different instructions are being executed in consecutive cycles. The ALU result output reflects correct arithmetic or logical operations based on the decoded instruction. Similarly, the write data signal shows the values being written back to the register file or memory, confirming proper data flow within the processor.

The clock signal drives the entire system, ensuring synchronization of all modules, while the reset signal initializes the processor at the beginning of execution. The consistent transitions and expected signal patterns validate that instruction fetch, decode, execute, memory access, and write-back operations are successfully completed within a single clock cycle.

Overall, the waveform demonstrates accurate processor functionality, confirming that the design is logically correct and ready for further implementation or FPGA deployment.

A. Discussion

The implementation of the 8-bit MIPS single-cycle processor demonstrates a clear and functional realization of fundamental processor architecture concepts. The simulation results confirm that each instruction is executed successfully within a single clock cycle, validating the correctness of the data path and control logic. The waveform outputs show proper synchronization between the Program Counter, instruction memory, ALU operations, and register updates, indicating that all modules are correctly integrated.

One of the key observations from this project is the simplicity achieved through the single-cycle design. Since all stages of instruction execution occur within one clock cycle, the control mechanism becomes straightforward, making it easier to design and understand. This is particularly beneficial for educational purposes, where clarity of operation is more important than performance optimization. However, this simplicity comes with a limitation: the clock cycle duration must be long enough to accommodate the slowest instruction, which reduces overall speed.

Another important aspect is the effectiveness of modular design. Each component, such as the ALU, register file, and control unit, was implemented

independently and later combined into a complete system. This approach simplifies debugging and enhances scalability. The use of Verilog HDL further enables accurate hardware modeling and easy simulation.

Despite being an 8-bit processor, the system successfully supports essential operations including arithmetic, logical, memory access, and control flow instructions. This proves that reduced bit-width designs can still effectively demonstrate processor functionality while minimizing hardware requirements.

Overall, the project highlights a trade-off between simplicity and performance. While the design is not optimized for speed, it provides a strong conceptual understanding of processor operation, making it highly suitable for academic learning and foundational research in digital system design.

VI. CONCLUSION

This project successfully presents the design and implementation of an 8-bit MIPS single-cycle processor using Verilog HDL. The primary objective of developing a simplified processor architecture for educational and practical understanding has been achieved. The processor integrates key components such as the Program Counter, Instruction Memory, Register File, ALU, Data Memory, and Control Unit, ensuring complete instruction execution within a single clock cycle.

The results obtained from simulation confirm that the processor performs correctly for various instruction types, including arithmetic, logical, memory, and control flow operations. The waveform analysis validates proper data flow, accurate control signal generation, and correct synchronization of all modules. The use of a modular design approach improves system clarity, ease of debugging, and flexibility for future expansion.

Although the processor operates with limited performance due to its single-cycle nature, it effectively demonstrates the core principles of processor architecture. The reduction to an 8-bit structure simplifies implementation while retaining essential MIPS functionalities, making it ideal for FPGA-based academic projects.

In conclusion, this work provides a strong foundation for understanding processor design at the hardware level. It bridges the gap between theoretical concepts and practical implementation, offering valuable insights into instruction execution and system integration. The project serves as a stepping stone for developing more advanced processor architectures in future research.

VII. FUTURE WORK

The current design of the 8-bit MIPS single-cycle processor provides a solid foundation for understanding processor architecture; however, several enhancements can be implemented to improve its performance and functionality. One of the most significant improvements would be the introduction of pipelining. By dividing instruction execution into multiple stages, pipelining can significantly increase throughput and reduce overall execution time. This would transform the processor from a simple educational model into a more efficient system.

Another important area for future work is increasing the bit-width of the processor from 8-bit to 16-bit or 32-bit. This would allow the processor to handle more complex computations and larger data sets, making it closer to real-world processor architectures. Additionally, expanding the instruction set to include more advanced operations such as multiplication, division, and floating-point calculations would enhance the processor's capabilities.

Integration of cache memory is another possible improvement. Adding cache can reduce memory access time and improve overall system performance. Furthermore, implementing hazard detection and forwarding techniques would be necessary when introducing pipelining to ensure correct execution.

The processor can also be extended by adding input/output interfaces and peripheral devices, enabling interaction with external systems. Implementing the design on advanced FPGA platforms such as Artix-7 or Zynq boards would provide better performance and real-time validation.

Overall, these enhancements would not only improve performance but also provide deeper insights into advanced processor design concepts, making the system suitable for both academic and research applications.

REFERENCES

[1] M. M. Mano and M. D. Ciletti, *Digital Design with an Introduction to the Verilog HDL*, 5th ed. Noida, India: Pearson Education, 2013.
[2] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 5th ed. Burlington, MA, USA: Morgan Kaufmann, 2014.
[3] Xilinx Inc., *ISE Design Suite User Guide*. San Jose, CA, USA: Xilinx Inc., 2014.
[4] Xilinx Inc., *Vivado Design Suite User Guide*. San Jose, CA, USA: Xilinx Inc., 2014.
[5] Various authors, "Design and implementation of MIPS processor using Verilog HDL," *IEEE and academic publications*.