International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Conference Proceedings: Volume: 05 ACME – 2026 DOI: 10.55041/ISJEM.ACME121
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Design and Implementation of a Single-Cycle RV32I Processor using Verilog HDL

A. S. Lavanya [1], S. Bindusagar[2], P. Haritha[3], P. Ganesh[4], K. Gnaneswar[5]

[1]Assistant Professor, Department of ECE, Annamacharya Institute of Technology and Sciences, Tirupati, Andhra Pradesh, India, lavanya04.aits@gmail.com

[2]Student Department of ECE, Annamacharya Institute of Technology and Sciences, Tirupati, Andhra Pradesh, India, sbindusagar2003@gmail.com

[3]Student Department of ECE, Annamacharya Institute of Technology and Sciences, Tirupati, Andhra Pradesh, India, harithaponna15@gmail.com

[4]Student Department of ECE, Annamacharya Institute of Technology and Sciences, Tirupati, Andhra Pradesh, India, dreamboyganesh29@gmail.com

[5]Student Department of ECE, Annamacharya Institute of Technology and Sciences, Tirupati, Andhra Pradesh, India, kambakamgnaneswar@gmail.com

---

Abstract - This paper presents the design and implementation of a single-cycle processor based on the RV32I instruction set architecture (ISA) using Verilog HDL. The processor supports basic 32-bit operations such as arithmetic, logical, memory access, and control flow instructions, covering common instruction formats including R-type, I-type, S-type, B-type, and J-type. The overall design is divided into modular blocks such as the Program Counter, Control Unit, Register File, Arithmetic Logic Unit (ALU), and memory units, making the architecture easier to understand and implement. The functionality of the processor is verified through behavioral simulation and RTL analysis using the Xilinx Vivado tool. The simulation results show correct instruction execution, including proper register updates, memory read/write operations, and branch handling. In addition to simulation, synthesis and implementation steps are also carried out to analyze how the design maps onto hardware resources. Although the single-cycle approach simplifies the design and avoids pipeline-related complexities, it also introduces limitations in terms of performance due to the longer clock cycle required for instruction execution. Overall, this work provides a simple and clear implementation of an RV32I processor, which can be useful for learning processor design concepts and for basic embedded applications.

Keywords: RISC-V, RV32I, Single-Cycle Processor, Verilog HDL, FPGA, Vivado

1.INTRODUCTION

RISC-V has gained significant attention in recent years due to its open-source nature and flexibility in processor design. Unlike traditional proprietary architectures, RISC-V allows designers to implement and customize processors without licensing restrictions. This makes it particularly useful in academic environments and research applications. The RV32I instruction set forms the base of RISC-V and supports essential 32-bit integer operations Designing a processor based on RV32I provides a good starting point for understanding processor architecture and data path design. Among different processor architectures, the single-cycle design is the simplest, where each instruction is executed in one clock cycle. This approach eliminates the need for complex control mechanisms such as pipelining and hazard handling. However, the clock cycle must be long enough to accommodate the slowest instruction, which limits performance. In this work, a single-cycle RV32I processor is designed using Verilog HDL. The design is verified using simulation and further evaluated through synthesis and implementation in the Vivado environment. The main goal is to develop a clear and functional processor design that demonstrates the working of basic instruction execution.

2. RELATED WORK

Barriga et al. [1] presented a methodology for designing processors based on the RISC-V instruction set. Their work includes three different implementations of RV32I cores. The first model is based on a high-level state

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Conference Proceedings: Volume: 05 ACME – 2026 DOI: 10.55041/ISJEM.ACME121
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

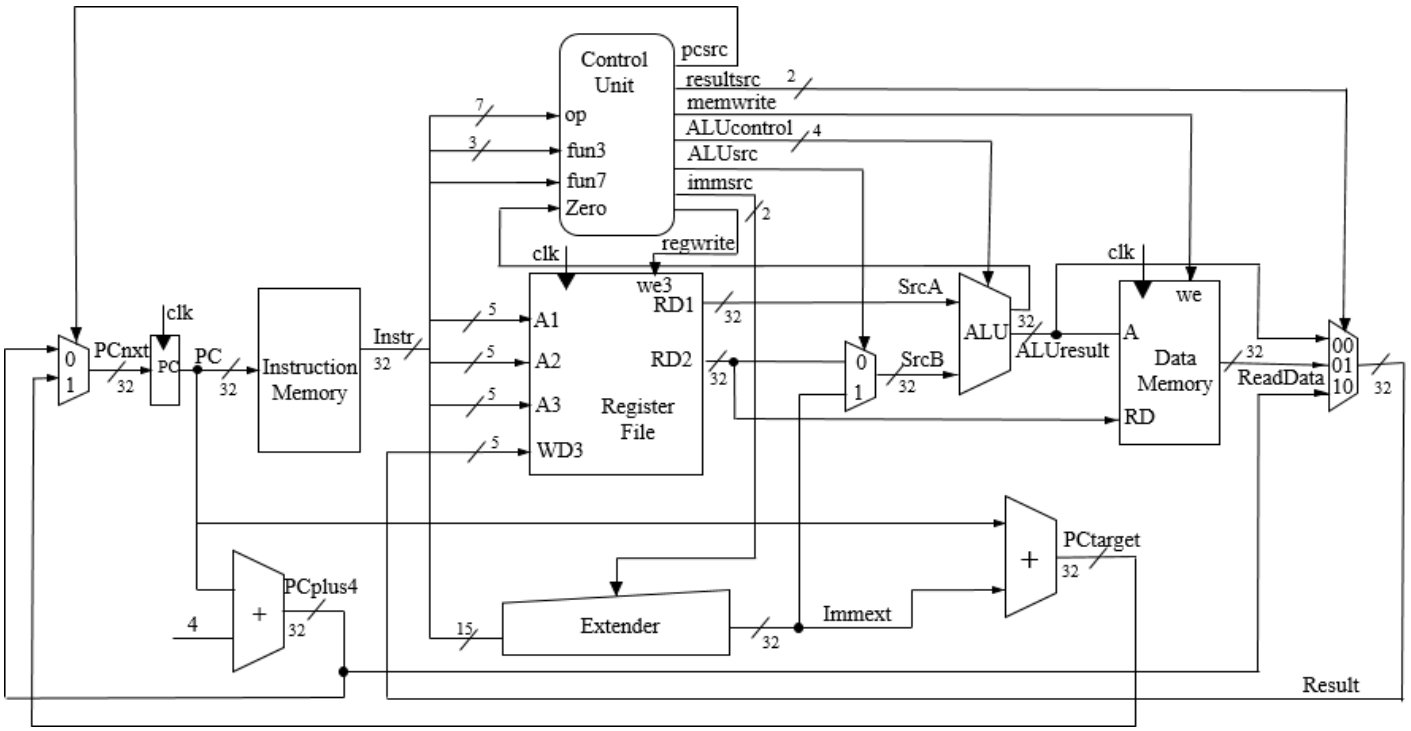

Fig. 1. Block diagram of the proposed single-cycle RV32I processor architecture

machine, mainly focusing on the behavior of instruction execution. However, this approach is not efficient for hardware implementation, as it results in higher resource usage and lower performance after synthesis. The second design follows a Von Neumann architecture, where instructions are executed one after another. The third is implementation uses a Harvard architecture and supports pipelining, allowing better performance through parallel execution. These design were implemented on a Xilinx FPGA, showing their practical applicability. Jain et al. [2] developed a RISC-V processor with additional bit manipulation capabilities and implemented it on an FPGA platform. The processor was designed using Verilog HDL and is based on the RV32I instruction set. Their architecture is divided into three main parts: datapath, control unit, and memory. The datapath handles the actual data processing, while the control unit decodes instructions and generates control signals. Memory is separated into instruction and data memory, which helps in efficient execution. Poli et al. [3] proposed a low-power implementation of a RISC-V processor using modern hardware design techniques. The design was written in SystemVerilog and included detailed module-level implementations along with testbenches. Their focus was mainly on reducing power consumption while maintaining correct functionality. A simplified version of the RV64I instruction set was used, and the design was successfully implemented on an FPGA. Nguyen-Hoang et al. [5] designed a RISC-V based processor system that includes cryptographic accelerationfeatures. Their implementation combines SpinalHDL and Verilog HDL to develop a 32-bit processor capable of running Linux. The system is based on the VexRiscV core and integrates cryptographic modules, making it suitable for secure applications such as IoT devices. Jiahui et al. [7] introduced a tool called MEIMAT, which is used for representing instructions of different processors using a common format. They extended this framework to support the RISC-V instruction set and demonstrated how RV32I instructions can be expressed using semantic and functional descriptions. This work helps in improving the flexibility of processor design tools. Puli and Pudi [8] discussed the power efficiency of ALU design in RISC-V processors. Their study highlights that RISC-based architectures achieve better energy efficiency due to simpler instruction sets and reduced hardware complexity. They focused on the RV32I architecture and analyzed how ALU operations can be optimized to reduce power consumption.

From the above studies, it can be observed that most existing works focus on improving performance, power efficiency, or adding advanced features. In contrast, this work focuses on designing a simple and modular single-

cycle processor to clearly demonstrate the working of the RV32I architecture.

3. PROPOSED ARCHITECTURE

The processor is designed using a single-cycle architecture in which all stages of instruction execution are completed in one clock cycle. The main components of the processor include:

- Program Counter (PC)
- Instruction Memory
- Control Unit
- Register File
- Arithmetic Logic Unit (ALU)
- Immediate Generator
- Data Memory
- Write-Back Logic

Each module is implemented independently and connected through a datapath.

A. Program Counter (PC)

The Program Counter (PC) is responsible for holding the address of the current instruction being executed. In this design, the PC updates sequentially by adding 4 to the current address, since each instruction is 32 bits wide. For branch and jump instructions, the PC is updated with a target address calculated based on immediate values and control signals.

The PC plays a critical role in controlling the flow of execution. It ensures that instructions are fetched in the correct order and allows conditional changes in control flow during branch operations.

B. Instruction Memory

Instruction Memory stores the program instructions in binary format. Each instruction is 32 bits wide and is accessed using the address provided by the Program Counter. When a new address is generated, the corresponding instruction is fetched and passed to the Control Unit and other components for decoding and execution.

In this design, instruction memory is modeled as a read-only memory (ROM) block, where instructions are preloaded for simulation and testing purposes.

C. Control Unit

The Control Unit is responsible for decoding the fetched instruction and generating appropriate control signals for the datapath. It takes input fields such as opcode, funct3, and funct7, and determines the type of instruction being executed.

Based on the instruction type, the Control Unit generates signals such as:

- RegWrite (register write enable)
- MemRead and MemWrite (memory operations)
- ALUSrc (selecting ALU input)

The Control Unit ensures proper coordination between all modules and controls the overall operation of the processor.

D. Register File

The Register File consists of 32 general-purpose registers, each 32 bits wide. It provides two read ports and one write port, allowing simultaneous reading of two operands and writing back the result.

During instruction execution, the source registers (rs1 and rs2) provide input operands to the ALU, and the destination register (rd) stores the result. The Register File plays an essential role in temporary data storage and data transfer between different stages of execution.

4. SYSTEM DESIGN AND IMPLEMENTATION

The proposed single-cycle RV32I processor is designed and implemented using Verilog HDL with a modular approach. Each functional block of the processor is developed as an independent module and later integrated to form the complete datapath. This approach simplifies debugging and makes the design easier to understand and verify.

A. Design Methodology

The design follows a bottom-up approach, where individual components such as the Program Counter, Register File, ALU, Control Unit, and memory modules are first implemented separately. Each module is tested individually to ensure correct functionality before integrating them into the complete processor design.

The datapath and control path are clearly separated. The datapath handles the movement and processing of data, while the control unit generates signals that determine how data flows through the system. This separation helps in maintaining clarity in design and reduces complexity during debugging.

B. Design Flow Using Vivado

The entire design is developed and verified using the Xilinx Vivado Design Suite. The design flow consists of the following steps:

1)BehavioralSimulation**:**
Initially, the Verilog modules are tested using a testbench to verify functional correctness. Different instruction types such as arithmetic, load, and branch instructions are applied, and their outputs are observed through simulation waveforms.

2)RTLAnalysis:
After simulation, RTL analysis is performed to visualize the structure of the design. The RTL schematic shows how different modules are interconnected and helps in verifying whether the datapath is correctly implemented.

3)Synthesis:
In this step, the Verilog code is converted into a gate-level representation. The synthesis report provides information about resource usage such as LUTs and flip-flops.

4)Implementation:
The synthesized design is mapped onto FPGA resources through placement and routing. This step gives an idea of how the design would behave in actual hardware, even though it is not physically deployed on an FPGA board.

C. Module Design and Integration

Each module in the processor is designed based on its specific functionality.

The Control Unit is implemented using combinational logic that decodes the instruction fields and generates the required control signals. The ALU performs arithmetic and logical operations based on control inputs. The Register File is designed to support simultaneous reading of two registers and writing to one register.

The Immediate Generator extracts immediate values from instructions and performs sign extension. The Data Memory is designed to support read and write operations based on control signals.

Once all modules are verified individually, they are integrated to form the complete processor. Proper connections between modules are ensured to maintain correct data flow.

D. Implementation Details

After integrating all modules, the complete processor design is tested using a testbench. The simulation results confirm that the processor correctly executes instructions such as load, arithmetic operations, and branch conditions.

The design is synthesized and implemented in the Vivado environment to analyze hardware utilization. The implementation results provide information about resource usage and confirm that the design is suitable for FPGA realization.

Although the processor is not deployed on a physical FPGA board, the synthesis and implementation results demonstrate that the design is hardware-ready and feasible.

5. RESULTS AND DISCUSSION

The designed single-cycle RV32I processor is verified through simulation and analysis using the Vivado environment. The results confirm correct execution of instructions and proper interaction between different modules of the processor.

A. Functional Verification

Functional verification is carried out using a Verilog testbench by applying different types of instructions. The simulation waveform clearly shows the behavior of the processor during instruction execution.

Fig. 2. Simulation waveform showing execution of instructions

The waveform includes signals such as the Program Counter (PC), instruction, ALU result, and control signals. It can be observed that:

- The Program Counter increments sequentially during normal execution

- Instructions are fetched correctly from memory
- The ALU produces correct outputs for arithmetic operations
- Register values are updated when the RegWrite signal is enabled
- Memory read and write operations occur correctly during load and store instructions

The waveform also demonstrates correct branching behavior, where the Program Counter updates based on branch conditions.

B. RTL Analysis

The RTL schematic generated in Vivado provides a structural view of the processor design. It shows how different modules such as the Control Unit, ALU, Register File, and memory blocks are connected.

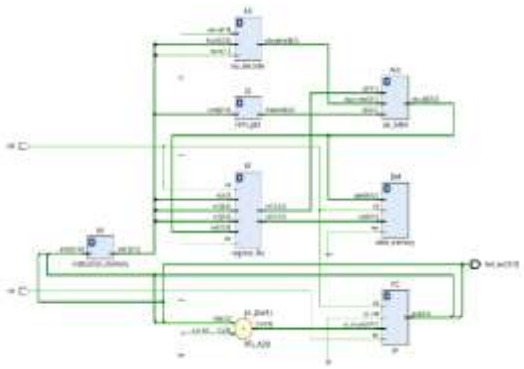

Fig. 3. RTL schematic of the processor design

The RTL view confirms that the datapath is correctly implemented and that all modules are properly interconnected.

C. Resource utilization

After synthesis and implementation, the resource usage of the design is analyzed. The results indicate efficient utilization of FPGA resources.

TABLE -1
RESOURCE UTILIZATION SUMMARY

Resource	Utilization
LUTs	1800
Flip-Flops	150
I/O Pins	30

The results show that the processor requires relatively low hardware resources, making it suitable for FPGA-based implementation.

D. Instruction Execution Analysis

To further verify the processor functionality, sample instructions are executed and their outputs are observed.

TABLE II
SAMPLE INSTRUCTION EXECUTION

Instruction	Operation	Result
lw t1, 0(x0)	Load data	t1 updated
add t2,t1,t1	Arithmetic addition	t2 = 2*t1
beq t1,t2,4	Branch condition	PC updated

The results confirm that each instruction is executed correctly, and the processor behaves as expected.

E. Control Signal Analysis

Control signals play an important role in coordinating processor operations. Their behavior is verified through simulation.

TABLE III
CONTROL SIGNAL DESCRIPTION

Signal	Function
RegWrite	Enables register update
MemRead	Writes to memory
ALUSrc	Selects ALU input

The control signals are observed to change appropriately depending on the instruction type.

F. Performance Discussion

The processor executes each instruction in a single clock cycle, resulting in a CPI (Cycles Per Instruction) of 1. However, the clock period is determined by the longest combinational path in the design.

This leads to a trade-off:

- Simpler design and control logic
- Reduced performance compared to pipelined processors

Despite this limitation, the design is effective for understanding processor architecture and basic applications.

G. Power analysis

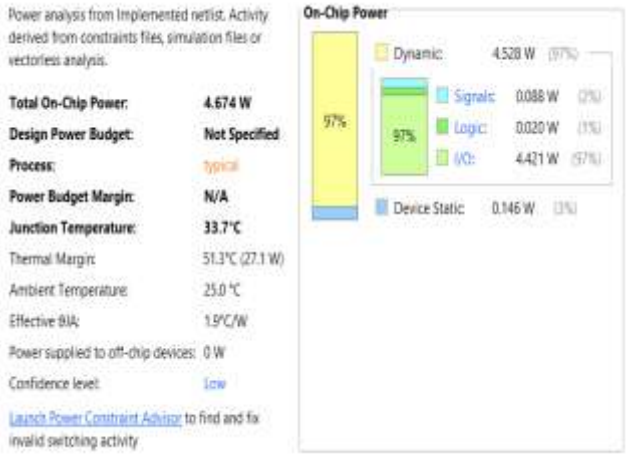

Fig. 4. Power analysis report generated from Vivado implementation

The power analysis of the designed processor is carried out using the Vivado tool after implementation. The total on-chip power consumption is observed to be approximately 4.67 W. The dynamic power contributes the majority of the total power, accounting for nearly 97% of the consumption, while static power accounts for a smaller portion.

Among the dynamic power components, I/O power dominates due to switching activity, followed by signal and logic power. The junction temperature is observed to be within safe limits, indicating that the design does not suffer from thermal issues under the given conditions.

These results provide an estimate of power consumption and demonstrate that the design is feasible for hardware implementation. However, further optimization can be performed to reduce power usage.

6. CONCLUSION

The design and implementation of a single-cycle RV32I processor using Verilog HDL has been successfully completed. The processor supports essential instruction types including arithmetic, logical, memory access, and control flow operations. The modular design approach made it easier to develop and verify each component individually before integrating the complete system.

The functionality of the processor is verified through simulation using the Vivado environment. The results confirm correct instruction execution, proper register updates, and accurate memory operations. Synthesis and implementation further demonstrate that the design is hardware-feasible with efficient resource utilization.

Although the single-cycle architecture simplifies control logic and avoids the complexity of pipelining, it limits performance due to the longer clock cycle required for instruction execution. Despite this limitation, the design provides a clear understanding of processor architecture and serves as a strong foundation for further improvements.

7. FUTURE WORK

The current design can be extended in several ways to improve performance and functionality. One possible improvement is the implementation of a pipelined architecture, which allows multiple instructions to be executed simultaneously and increases overall throughput. Introducing hazard detection and forwarding mechanisms would further enhance the efficiency of the processor. Additional improvements may include optimizing the ALU design, reducing power consumption, and extending the instruction set to support advanced operations. The processor can also be implemented on physical FPGA hardware to evaluate real-time performance.

8. REFERENCES

[1] A. Barriga, "RISC-V processors design: A methodology for cores development,"
in Proc. XXXV Conf. Design of Circuits and Integrated Systems (DCIS),
Segovia, Spain, 2020.

[2] V. Jain, A. Sharma, and E. A. Bezerra,
"Implementation and extension of bit manipulation instruction on RISC-V architecture using FPGA,"
in Proc. IEEE 9th Int. Conf. Communication Systems and Network Technologies (CSNT),
Gwalior, India, 2020.

[3] L. Poli, S. Saha, X. Zhai, and K. D. McDonald-Maier,
"Design and implementation of a RISC-V processor on FPGA,"
in Proc. 17th Int. Conf. Mobility, Sensing and Networking (MSN),
Exeter, U.K., 2021.

[4] J.-Y. Lai, C.-A. Chen, S.-L. Chen, and C.-Y. Su,

"Implementation of a 32-bit RISC-V architecture processor using Verilog HDL,"
in Proc. Int. Symp. Intelligent Signal Processing and Communication Systems (ISPACS),
Hualien City, Taiwan, 2021.

[5] D. T. Nguyen-Hoang et al.,
"Implementation of a 32-bit RISC-V processor with cryptography accelerators on FPGA and ASIC,"
in Proc. IEEE 9th Int. Conf. Communications and Electronics (ICCE),
Nha Trang, Vietnam, 2022.

[6] J. Saussereau, C. Jego, C. Leroux, and J.-B. Begueret,
"Design and implementation of a RISC-V core with a flexible pipeline for design space exploration,"
in Proc. IEEE Int. Conf. Electronics, Circuits and Systems (ICECS),
Istanbul, Türkiye, 2023.

[7] L. Jiahui et al.,
"Microprocessor instruction design tool for RISC-V architecture,"
in Proc. Int. Symp. Communications and Information Technologies (ISCIT),
Sydney, Australia, 2023.

[8] K. Puli and V. Pudi,
"Design of low power ALU for RISC-V ISA,"
in Proc. IEEE Int. Symp. Smart Electronic Systems (iSES),
Ahmedabad, India, 2023.

[9] E. Cui, T. Li, and Q. Wei,
"RISC-V instruction set architecture extensions: A survey,"
IEEE Access, vol. 11, pp. 24696–24711, 2023.

[10] M. Wygrzvwalski, P. Skrzypiec, and R. Szczygiel,
"Hardware acceleration method using RISC-V core with no ISA extensions."