EVOLUTION IN ELECTRICAL AND ELECTRONIC ENGINEERING
e-ISSN: 2756-8458

EEEE

Vol. 6 No. 2 (2025) 465-476
https://publisher.uthm.edu.my/periodicals/index.php/eeee

Design of 32-Bit RISC-V Processor Using Verilog Hardware Description Language

Lim Huang Hong[1], Chua King Lee[1]*

[1] *Faculty of Electrical and Electronic Engineering,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: chua@uthm.edu.my
DOI: https://doi.org/10.30880/eeee.2025.06.02.056

Article Info

Received: 30 June 2025
Accepted: 03 September 2025
Available online: 30 October 2025

Keywords

RISC-V Architecture, Verilog HDL, RV32I Instruction Set, Functional Simulation, Timing Analysis.

Abstract

RISC-V architecture is an open standard instruction set architecture that is established based on of RISC architecture. The RISC-V architecture is an open source, free, user-friendly access, and easy to modify. RISC-V offers a reduced instruction set that is lightweight and modular. Thus, its architecture is less complex and easy to be customised. This paper presents the design and development of a 32-bit RISC-V processor architecture from fundamental concepts, encompassing instruction set definition, datapath design and control logic implementation. Verilog HDL is used to design the architecture that comprises combination of behavioural modelling and structural modelling. The processor is designed based on the RV32I instruction set where the key components include Program Counter, Instruction Memory, Register File, ALU, Immediate Generator, Data Memory, Branch Comparator, Multiplexer and Control Unit. The design is expected to perform operations for instruction types of R, I, B, S, U and J. Eleven instructions were chosen for the implementation and validation. The functionality of the design was tested using ModelSim-Altera, to ensure the architecture able to produce the desired results. After the functional test, a timing performance analysis is conducted to ensure that the processor meets the time requirements. The simulation results indicate that the designed architecture able to produce correct results for all the selected operations. In addition, the timing analysis shows that the processor can operate with a maximum operating frequency of 40.88 MHz. As conclusion, the processor met all the design objectives and functions as expected. The study also revealed that the Performance (High Effort) mode is the most suitable for speed optimization, achieving the highest Fmax of 62.64 MHz, indicating the design can operate at a higher clock speed. Meanwhile, the Power (Aggressive) mode is most efficient in terms of logic cell usage and power consumption, with values of 21,018 and 183.24 mW respectively.

1. Introduction

Reduced Instruction Set Computer – V (RISC-V) architecture is an open standard instruction set architecture that is based on the established of RISC principles. RISC describes a computing architecture that executes complex task with many simple instructions. Alternative to RISC is Complex Instruction Set Computer (CISC) principles which execute a small number of more complex instructions to complete a processing activity [1]-[2]. The

This is an open access article under the CC BY-NC-SA 4.0 license.

Instructions Set Architecture (ISA) defines a set of basic operations which are needed by instructions for the execution [3].

In the evolution of processor architecture, traditional proprietaries such as x86 and ARM create barriers to innovation due to high licensing costs and restricted customizability. These have limited the development of cost-effective, specialized processors, especially educational purposes [4]. As RISC-V was introduced to support research and education, it offers free and flexible architecture to overcome traditional constraints in the IC design industry. Thus, RISC-V is particularly well-suited for educational use, particularly in fields like computer architecture and embedded systems [5] - [6]. RISC-V is way more modular as it has a smaller standard base ISA meaning that developers can start with a minimal base instruction set and then customized processor that meets the unique requirements of different applications without unnecessary complexity. RISC-V is much more stable as the base and first standard extension are fixed, so there is no need for updates, less time is spent on maintenance and more time on innovation and performance improvement [5] - [6].

This paper describes the design of a 32-bit RISC-V processor using Verilog Hardware Description Language (HDL). The processor is implemented with a reduced subset of the RISC-V base ISA, ensuring simplicity while maintaining core functionality. In terms of Quartus Prime Lite tools that are acquired to validate design functionality and analyze the design performance. The research objectives are, design datapath unit and control units for 32-bit RISC-V architecture using Verilog language, validating functionality of the designed RISC architecture with ModelSim-Altera simulation, and analyze the design performance in terms of area, latency time, and power consumption.

The selected RV32I Base Instruction Set for the processor design, categorized into six types: R-type, I-type, B-type, S-type, U-type, and J-type. Each type specifies operations such as arithmetic operation, (ADD and SUB), logical operation, (AND and OR), arithmetic operation with immediate, (ADDI), memory access operation, (LW and SW), control flow operation, (BEQ and JAL), and immediate value operation, (LUI and AUIPC) as shown in Table 1.

Table 1 provides the detailed of all instructions, including bit-width positions and the function of each field such as source registers (rs1, rs2), destination register (rd), immediate values (imm), function codes (funct3, funct7), and opcodes. R-type instructions use two source registers and a destination register to perform arithmetic and logical operations, these instructions rely on the funct7, funct3, and opcode fields to define the specific operation. I-type instructions like ADDI use a single source register and an immediate value, allowing simple arithmetic with constants while, LW is used to load data from memory. S-type instructions, such as SW, are used for storing data from a register into memory, where the address is calculated using a base register and an immediate offset. B-type instructions like BEQ are used for conditional branching, where the processor jumps to a target address if two register values are equal. J-type (JAL) and U-type (LUI, AUIPC) instructions handle jump operations and immediate value processing, often used for control transfer and address calculation.

Table 1 *Selected RV32I Base Instruction Set in The Processor Design*

Arithmetic Operation							
Bit-widths, position		[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]
Format	Mnemonic	funct7	rs2	rs1	funct3	rd	opcode
R-type	ADD	0000000	rs2	rs1	000	rd	0110011
R-type	SUB	0100000	rs2	rs1	000	rd	0110011

Logical Operation							
Bit-widths, position		[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]
Format	Mnemonic	funct7	rs2	rs1	funct3	rd	opcode
R-type	AND	0000000	rs2	rs1	111	rd	0110011
R-type	OR	0000000	rs2	rs1	110	rd	0110011

Arithmetic Operation with Immediate						
Bit-widths, position		[31:20]	[19:15]	[14:12]	[11:7]	[6:0]
Format	Mnemonic	imm[11:0]	rs1	funct3	rd	opcode
I-type	ADDI	imm[11:0]	rs1	000	rd	0010011

Memory Access Operation						
Bit-widths, position		[31:20]	[19:15]	[14:12]	[11:7]	[6:0]
Format	Mnemonic	imm[11:0]	rs1	funct3	rd	opcode
I-type	LW	imm[11:0]	rs1	010	rd	0000011

Evolution in Electrical and Electronic Engineering Vol. 6 No. 2 (2025) p. 465-476

Table 1 *Continued*

Bit-widths, position		[31:25]	[24:20]	[19:15]	[14:12]	[11:7]	[6:0]
Format	Mnemonic	imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
S-type	SW	imm[11:5]	rs2	rs1	010	rd	0100011

Control Flow Operation							
Bit-widths, position		[31|30:25]	[24:20]	[19:15]	[14:12]	[11:8|7]	[6:0]
Format	Mnemonic	imm[12|10:5]	rs2	rs1	funct3	imm[4:1|11]	opcode
B-type	BEQ	imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011
Bit-widths, position		[31|30:21|20|19:12]				[11:7]	[6:0]
Format	Mnemonic	imm[20|10:1|11|19:12]				rd	opcode
J-type	JAL	imm[20|10:1|11|19:12]				rd	1101111

Immediate Value Operation				
Bit-widths, position		[31:12]	[11:7]	[6:0]
Format	Mnemonic	imm[31:12]	rd	opcode
U-type	LUI	imm[31:12]	rd	0110111
U-type	AUIPC	imm[31:12]	rd	0010111

2. Literature Review

Table 2 provides a summary of previous studies on the implementation of 32-bit RISC-V processors, highlighting their varying levels of complexity and the diverse range of applications. First and foremost, [7] features a simple single-cycle RV32I design targeting IoT applications but lacks pipelining and hazard mitigation. Conversely, David Ngu Teck Joung's [8] approach employs a five-stage pipeline with advanced hazard detection, making it suitable for broader applications; however, it lacks detail on power analysis and hardware validation. Furthermore, [9] introduces an asynchronous five-stage pipelined design, emphasizing power efficiency and reduced delay, but does not offer hardware implementation or comparative analysis. Whereas, both the [10] and [11] implement five-stage pipelines with hazard mitigation, focusing on modularity and performance, but lack extended instruction sets and hardware validation.

Table 2 *Comparison of Previous Work*

Aspect / Title	Processor	ISA	Pipeline	Application	Future Work
Implement 32-bit RISC-V Architecture Processor using Verilog HDL [7]	Single-cycle RV32I processor	Supports 13 RV32I instructions	Single cycle, no pipelining	IoT and embedded systems	Extend instruction set, hardware validation
Design and Simulate RISC-V Processor Using Verilog [8]	5-stage pipelined RV32I processor	Supports 9 RV32I instructions	5-stage pipelining with hazard mitigation	Broader applications	Enhance pipeline stages and optimize energy efficiency
Design of 32-Bit Asynchronous RISC-V Processor Using Verilog [9]	5-stage asynchronous RV32I processor	Supports 8 RV32I instructions	5-stage pipelining, asynchronous architecture	Systems requiring low power and asynchronous designs	Scale to 64-bit design and optimize area and logic
Design of RISC-V Processor Using Verilog (IJRPR) [10]	5-stage pipelined RV32I processor	Supports 7 RV32I base instructions	5-stage pipelining with hazard unit	General-purpose processor design	Expand instruction set, hardware validation

Table 2 *Continued*

Aspect / Title	Processor	ISA	Pipeline	Application	Future Work
A Novel 32-Bit RISC-V Based Pipelined Processor Design Using Verilog [11]	5-stage pipelined RV32I processor	RV32I base instructions	5-stage pipelining with hazard unit	Low-power, high-performance embedded systems	Hardware implementation and performance analysis
Low-Power Design of a Datapath Architecture Based on RISC-V Processor Core [12]	Single-cycle RV32I processor	Supports 13 RV32I base instructions	Single cycle, no pipelining	Low-power IoT devices	Add pipelining and explore more on power-saving techniques
An Enhanced UTHM RISC-V Processor Core Architecture Implemented on FPGA [13]	Single-cycle RV32I processor	Supports 23 RV32I instructions	Single cycle, no pipelining	Targeted for educational purposes and embedded system prototyping. Demonstrated with LED blinking and binary counter program	Implement pipelining to improve performance and enhance instruction set further

Moreover, Chong Li Ting's [12] features a single-cycle thirteen instruction RV32I implementation described in Verilog HDL, targeting low-power IoT applications by using clock gating but lacks pipelined execution and advanced hazard handling. Lastly, [13] by Goh Jun Kang, features a single-cycle RV32I design extended to twenty-three instructions for improved flexibility, implemented in Verilog HDL and deployed on an FPGA, but lacks pipelining and advanced hazard management. In general, the seven papers highlight modularity, pipelining, and hazard handling, as well as extending instruction sets, providing detailed power and performance metrics, and validating designs on hardware. These enhancements would significantly strengthen their contributions to RISC-V development.

3. Methodology

Fig. 1 illustrates the flowchart for designing a 32-bit RISC-V processor. The development process includes stages: Datapath Unit (DU) design, Control Unit (CU) design, and the integration of both designs into a top entity architecture testing and performance analysis. A literature review is conducted to gather technical information on the processor architecture before proceeding to define the design specifications.

After identifying the design scope, the corresponding components for the DU and CU of the processor are created and tested individually. To visualize internal structures for each unit, the common practice is to manually create the block diagram, then compute its RTL code and test with ModelSim-Altera simulation. The top entity architecture of the processor would be constructed after all the DU and CU are designed and function correctly. At this stage, the DU and CU are combined and tested ModelSim-Altera.

If the top entity architecture passes the functionality test, the process continues with timing simulation, where the processor's timing behavior is tested. Between these analyses, timing simulation results are examined to ensure that no timing violations occur in the design. The final step is to analyze the performance of the processor in terms of speed, area, and power consumption.

Evolution in Electrical and Electronic Engineering Vol. 6 No. 2 (2025) p. 465-476

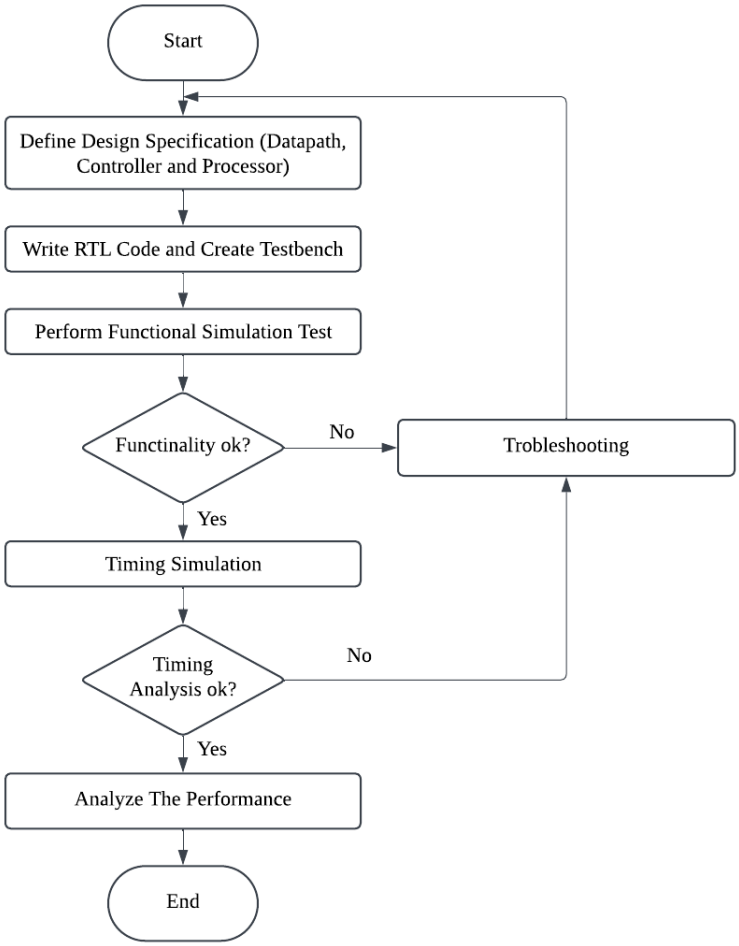

Fig. 1 *The Flowchart of the 32-bit RISC-V Processor Design Process*

4. Results and Discussion

This section presents simulation results and observations for the design, demonstrating the operations that can be performed by the design and its capabilities, including maximum operating frequency, required design area, and power consumption. The functionality of selected instructions is demonstrated through output waveforms generated using ModelSim-Altera, presented according to the type of operations for each instruction. Furthermore, the timing analysis is conducted to assess the performance and reliability of the design under the Slow 1200mV 85°C timing model. For performance analysis, six optimization modes were applied to observe the best condition that the design can achieve.

4.1 Complete RISC-V Architecture Processor Design

Fig. 2 and Fig. 3 illustrate the block diagrams for the proposed RISC-V architecture design and the actual RISC-V architecture design, respectively. Fig. 3 is a netlist diagram captured from Quartus Prime Lite. It is observed that all the key components, such as the Program Counter, Instruction Memory, Register File, ALU, Immediate Generator, Data Memory, Branch Comparator, and the Multiplexer, are all correctly implemented and connected. The control logic can input to these components using external inputs such as PCSel, ALUSelA, ALUSelB, and RFWEn.

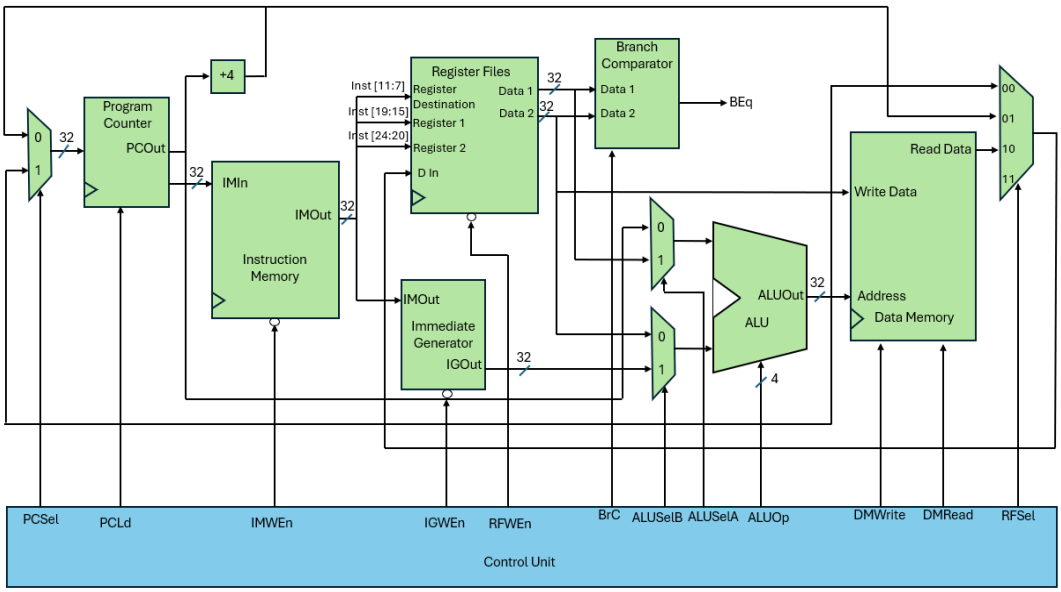

Fig. 2 *Proposed RISC-V Architecture*

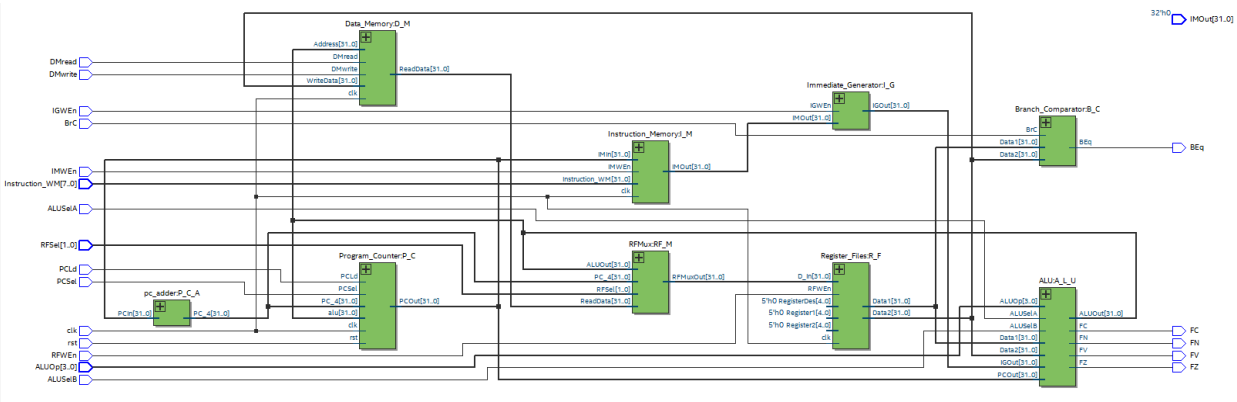

Fig. 3 *Netlist of RISC-V Architecture, Capturing from Quartus Prime Lite*

4.2 Functionality of Datapath Architecture

The following simulation waveform is generated using ModelSim-Altera. It is used to analyze functional behaviors of the proposed RV32I processor for instructions. There are six waveforms, where each is selected to exhibit the result of the type of operations that the processor can execute: Arithmetic Operation (R-type), Logical Operation (R-type), Arithmetic Operation with Immediate (I-type), Memory Access Operation (I-type), Control Flow Operation (B-type), Immediate Value Operation (U-type).

4.2.1 Arithmetic Operation (R-type)

Fig. 4 depicts the simulation result of the arithmetic instruction, ADD x1, x2, x3. The instruction performs the addition operation of registers x2 and x3, the result is stored at register x1. Table 3 shows the operation of the instruction in mathematical format.

Evolution in Electrical and Electronic Engineering Vol. 6 No. 2 (2025) p. 465-476

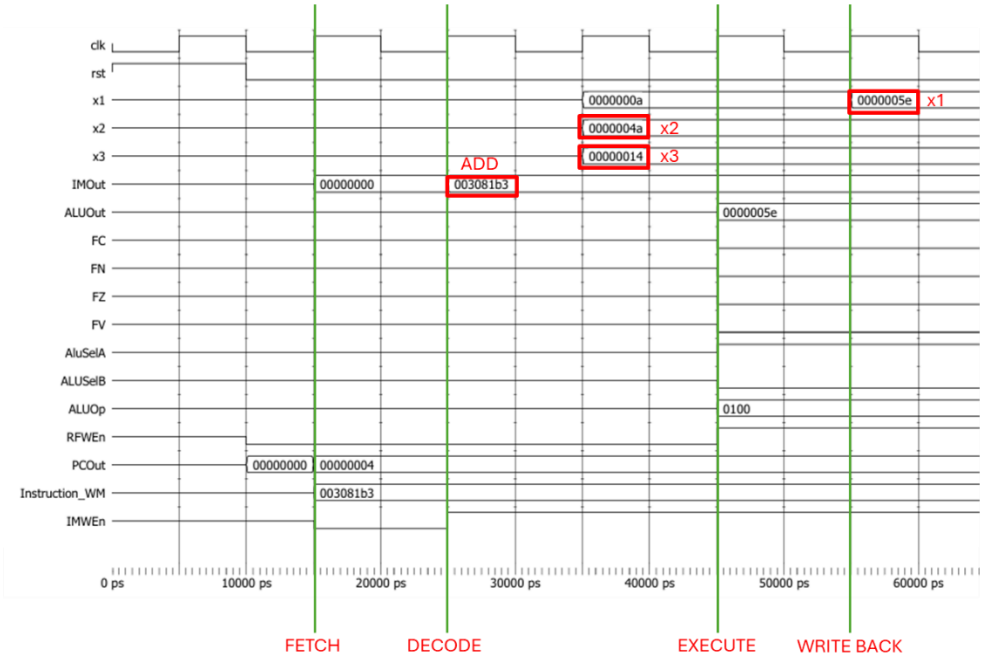

Fig. 4 *The ADD Instruction Simulation Waveform*

Table 3 *Operation of ADD Instruction*

Instructions:		ADD x1, x2, x3
Math Operation:		x2	0x0000004A
	+	x3	0x00000014
		x1	0x0000005E

4.2.2 Logical Operation (R-type)

Fig. 5 illustrates the execution of the instruction AND x7, x8, x9, which performs a bitwise AND logic operation between the contents of registers x8 and x9, with the resulting value stored in register x7. The detailed logical operation corresponding to this instruction is presented in Table 4.

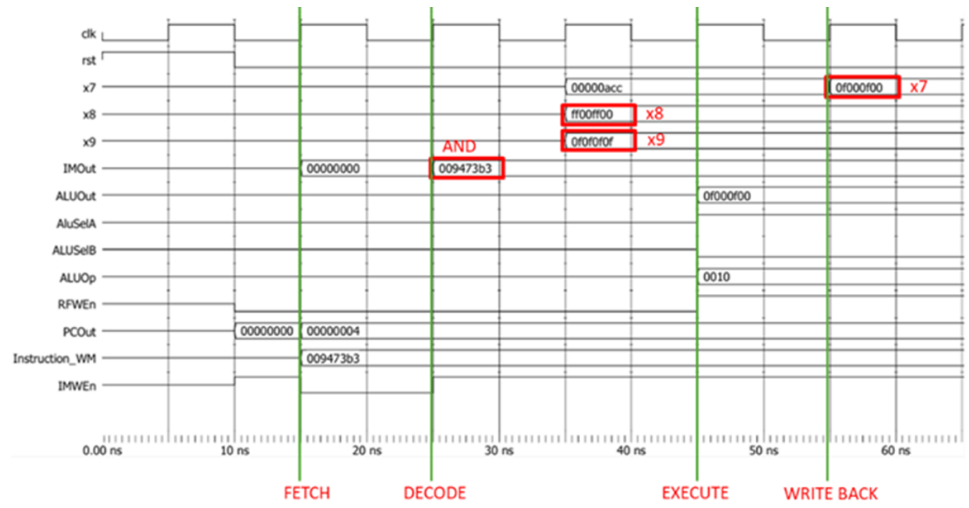

Fig. 5 *The AND Instruction Simulation Waveform*

Table 4 *Operation of AND Instruction*

Instructions:		AND x7, x8, x9		
Logical Operation:		x7		0xFF00FF00
	&	x8	&	0x0F0F0F0F
		x9		0x0F000F00

4.2.3 Arithmetic Operation with Immediate (I-type)

Fig. 6 illustrates the execution of the ADDI x14, x15, #25 instruction, in which the immediate value #25, its equivalent hexadecimal value 32'h0000019 is added to the content of register x15, and the resulting value is stored in register x14. The detailed execution process of this instruction in the designed RISC-V processor is presented in Table 5.

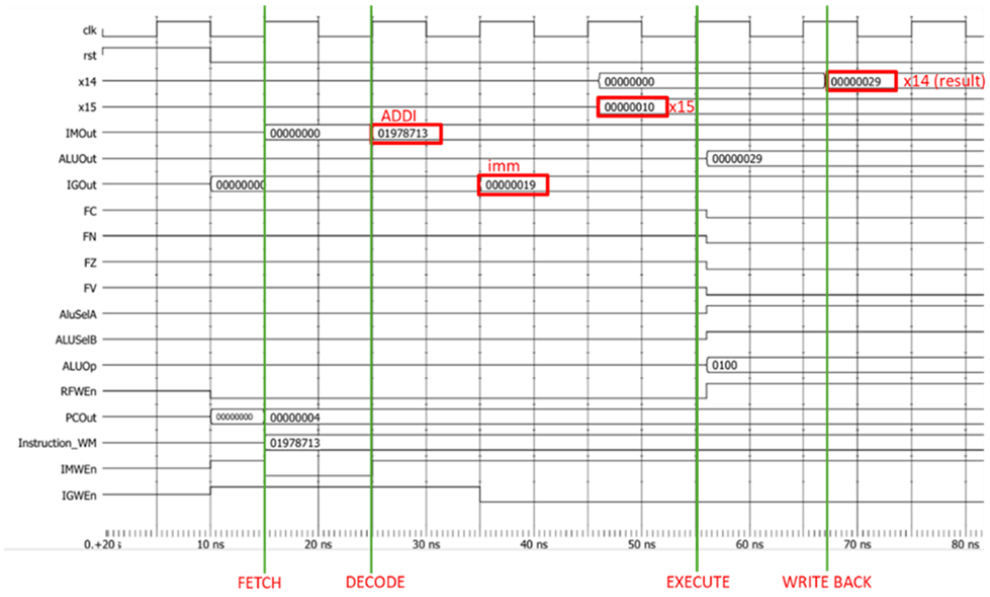

Fig. 6 *The ADDI Instruction Simulation Waveform*

Table 5 *Operation of ADDI Instruction*

Instructions:		ADDI x14, x15, #25	
Math Operations:		x15	0x00000010
	+	#25	0x00000019
		x14	0x00000029

4.2.4 Memory Access Operation (I-type)

Fig. 7 illustrates the execution of the LW x22, 8(x23) instruction, in which the content of register x23 is added with the immediate value #8, its equivalent hexadecimal 32'h0000008, to compute the effective memory address. The value stored at this computed address is then loaded into operand x22. Table 6 presents the detailed operation of this instruction.

Evolution in Electrical and Electronic Engineering Vol. 6 No. 2 (2025) p. 465-476

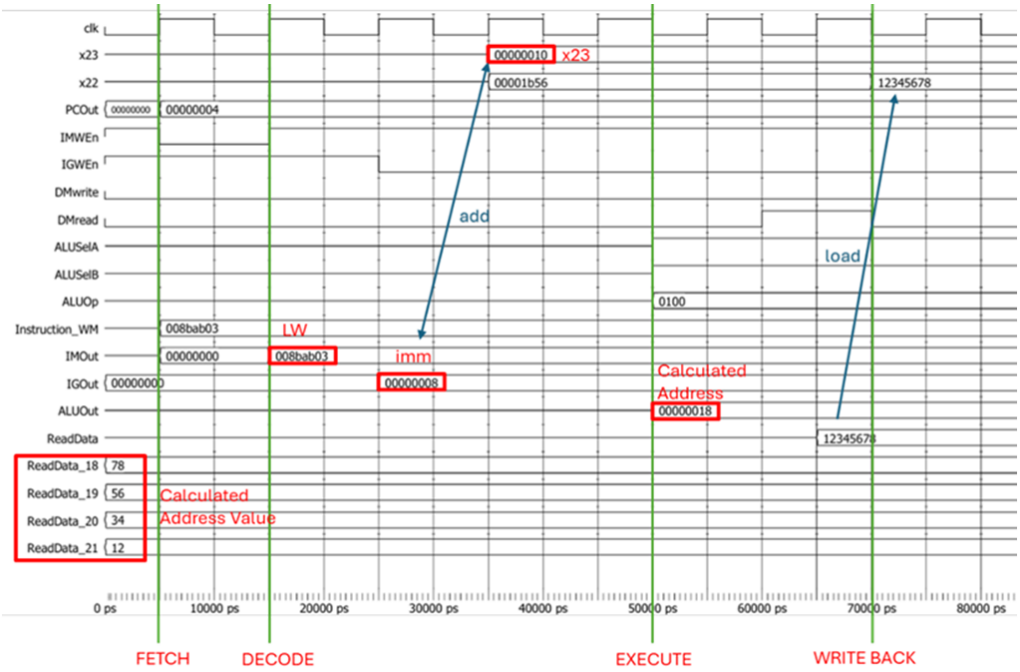

Fig. 7 *The LW Instruction Simulation Waveform*

Table 6 *Operation of LW Instruction*

Instructions:		LW x22, 8(x23)
Math Operations:	x23	0x00000010
	+ #8	0x00000008
	x22	0x00000018

4.2.5 Control Flow Operation (B-type)

Fig. 8 shows the simulation waveform of the BEQ x17, x18, #16 instruction. The BEQ instruction compares the contents of registers x17 and x18. If the values are equal, the program counter PC jumps to a new address computed by adding the specified offset to the current PC. Otherwise, the PC is incremented by 4 to fetch the subsequent instruction. The immediate value #16 corresponds to hexadecimal 32'h0000010. Table 7 presents the detailed operation of this instruction.

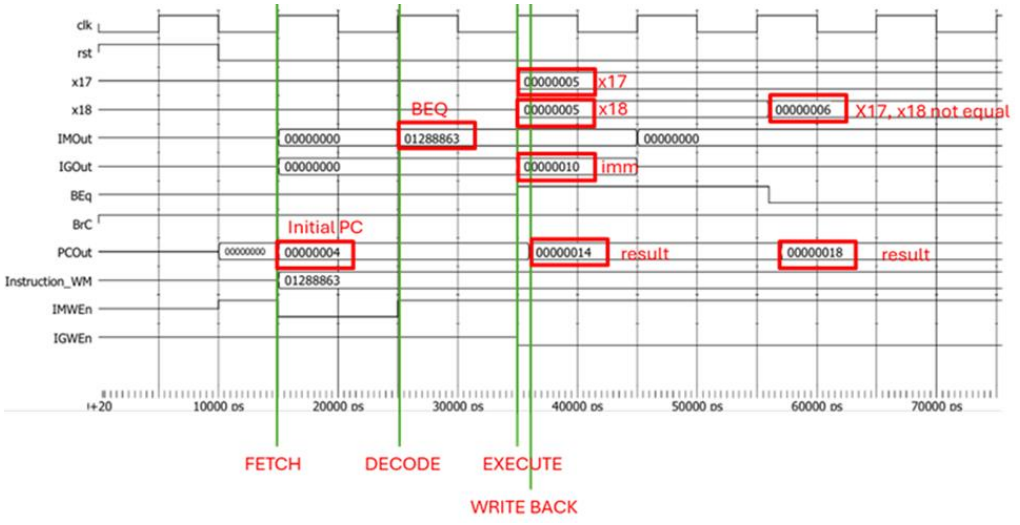

Fig. 8 *The BEQ Instruction Simulation Waveform*

Table 7 *Operation of BEQ Instruction*

Instructions:		BEQ x17, x18, #16	
Compare Operations:	x17	0x00000005	0x00000005
	x18	0x00000005	0x00000006
		Equal	Not equal
Math Operations:	Current PC value	0x00000004	0x00000014
	Offset value	0x00000010	0x00000004
	New PC value	0x00000014	0x00000018

4.2.6 Immediate Value Operation (U-type)

Fig. 9 illustrates the execution of the LUI x13, #100 instruction, in which the 20-bit immediate value #100 is loaded into the upper 20 bits of register x13, while the lower 12 bits are reset to zero. The immediate value #100 corresponds to the hexadecimal value 32'h0000064. Table 8 presents the equivalent operation, where the immediate value is left shifted by 12 bits and stored in register x13, with bits 0-11 filled with zeros.

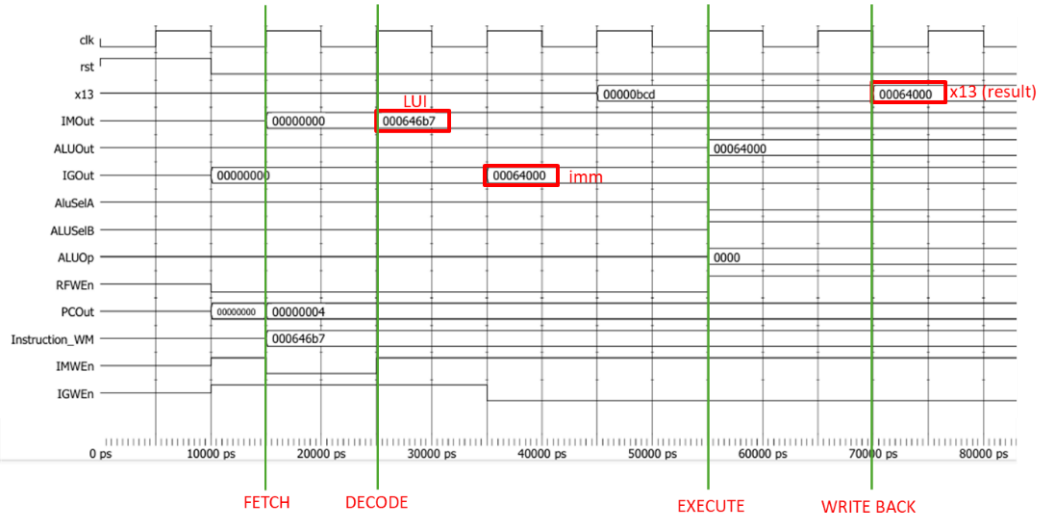

Fig. 9 *The BEQ Instruction Simulation Waveform*

Table 8 *Shift Operation of LUI Instruction*

Instructions:	LUI x13, #100
Shift Operations:	0x00000064
Shifted 12bits	0x000064000

4.2.7 Performance Analysis of RISC-V Architecture

Table 9 presents the performance of the proposed design for various optimization modes. In the balanced mode, the design achieves a maximum operating frequency, Fmax of 40.88 MHz. The architecture is implemented using 21,822 logic cells and consumes 193.87 mW of power.

The Balanced mode provides a general optimization approach, balancing speed, area, and power without emphasizing any single metric. This mode is appropriate for early-stage development or general-purpose applications where stable and consistent results are preferred.

In Performance (High Effort) mode, the compiler applies additional timing optimizations, resulting in an operating frequency of 62.64 MHz, a logic cell utilization of 21,843, and a power consumption of 193.77 mW. This mode is suitable for designs where achieving maximum speed is crucial, without significantly increasing area or power. Similarly, Performance (Aggressive) mode also aims to enhance speed, but at the expense of higher area and power consumption. The design achieved a frequency of 58.14 MHz, with the highest logic cell usage of 24,623 and a maximum power usage of 197.8 mW. This mode is designed for high-throughput processing systems where performance takes precedence over power efficiency and area constraints.

Evolution in Electrical and Electronic Engineering Vol. 6 No. 2 (2025) p. 465-476

Table 9 *Comparison of Various Compiler Optimization Modes*

Optimization Mode	Fmax (speed)	Logic Cells	Total Power Consumption
Balanced	40.88MHz	21822	193.87mW
Performance (high effort - increase runtime)	62.64MHz	21843	193.77mW
Performance (aggressive - increase runtime and area)	58.14MHz	24623	197.80mW
Power (high effort - increase runtime)	42.08MHz	21822	186.41mW
Power (aggressive - increase runtime and area)	43.92MHz	21018	183.24mW
Area (aggressive - increase runtime and area)	40.88MHz	21822	193.87mW

For designs prioritizing power reduction, Power (High Effort) mode optimizes dynamic power, achieving 42.08 MHz with 21,822 logic cells and reduced power consumption of 186.41 mW. This mode is suitable for low-power or battery-operated embedded systems, maintaining a reasonable balance between area and performance. Additionally, Power (Aggressive) mode further minimizes power consumption to 183.24 mW which is the lowest among all modes while maintaining a frequency of 43.92 MHz and utilizing 21,018 logic cells, which is the smallest logic cells. This mode is appropriate for designs where both power and chip area minimization are critical.

Lastly, the Area (Aggressive) mode yields results identical to those of the Balanced mode, with a 40.88 MHz frequency, 21,822 logic cells, and 193.87 mW power consumption. This indicates that the design has already achieved area efficiency, and further reduction is constrained by timing and functional requirements. This mode is typically employed in scenarios where fitting the design into a limited chip area is essential.

As a result, the Performance (High Effort) mode is the most suitable for improving the design's performance in terms of speed. This is because this mode produced the highest Fmax of 62.64 MHz, which shows the design can operate at a faster clock speed compared to other optimization modes. Additionally, the Power (Aggressive) is the most suitable option in terms of logic cell usage and power consumption, with values of 21018 and 183.24 mW, respectively.

In conclusion, the simulation and timing results confirm that the processor meets its design objectives in terms of both functionality and performance. Functional verification using ModelSim-Altera showed correct output for all eleven implemented instructions, covering R, I, S, B, U, and J formats. Timing analysis using Quartus Prime Lite initially revealed a maximum frequency (Fmax) of 61.69 MHz, which was later adjusted to 40.88 MHz after resolving timing violations. Further optimization revealed that the Performance (High Effort) mode produced the highest Fmax of 62.64 MHz, ideal for high-speed operation. In contrast, the Power (Aggressive) mode minimized logic cell usage (21,018) and power consumption (183.24 mW), making it more efficient for power-constrained applications. These results demonstrate the design’s flexibility and ability to adapt to varying embedded system needs by balancing speed, area, and energy efficiency.

5 Conclusion

In conclusion, the 32-bit RISC-V architecture has been successfully designed and validated using ModelSim-Altera. The processor has correctly executed instructions for arithmetic and logical operations, arithmetic operations with immediate values, memory access operations, control flow operations, and immediate value operations. This indicates that the architecture is capable of performing real-world computing tasks. Furthermore, using the built-in feature in Quartus Prime Lite, the timing violation can be resolved by lowering the operating frequency, where the new design timing constraints are encapsulated in an SDC file. The study revealed that the Performance (High Effort) mode is the most suitable for speed optimization due to the highest Fmax of 62.64 MHz, which shows the design can operate at a faster clock speed, while Power (Aggressive) is the most suitable for optimization in terms of logic cells usage and power consumption, where the values are 21018 and 183.24 mW respectively.

Thus, the 32-bit RISC-V processor developed in this study achieves a balanced trade-off between performance, power, and area through configurable synthesis options. Compared to the asynchronous design in [10], which reduces power and delay through micro-pipelining, the synchronous approach presented here offers a higher operating frequency (62.64 MHz) and a simpler implementation using standard design tools. Unlike the use of custom instruction formats in [10], this design supports the standard RV32I instruction set. While the processor in [11] emphasizes pipelining and hazard mitigation, it lacks analysis of power-performance trade-offs. The design in [9] presents a basic RISC-V implementation without optimization data. Overall, the design

addresses key limitations of previous approaches by supporting standard instructions, providing flexible design settings, and being ready for use in real-world embedded systems.

Acknowledgement

The authors gratefully acknowledge the support of the Faculty of Electrical and Electronic Engineering at Universiti Tun Hussein Onn Malaysia.

Conflict of Interest

Authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Lim Huang Hong, Chua King Lee;* ***data collection:*** *Author Y;* ***analysis and interpretation of results:*** *Lim Huang Hong, Chua King Lee;* ***draft manuscript preparation:*** *Lim Huang Hong, Chua King Lee. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] F. Masood, "RISC AND CISC Computer Architecture," *Computing Research Respository - CORR,* p. 21, 2011.
[2] A. Waterman, K. Asanovic, Y. Lee and D. A. Patterson, "The RISC-V Instruction Set Manual, Volume I: Base Iser-Level ISA," p. 32, 2011.
[3] P. Arul, N. Abirami, S. S, V. P and A. V, "Implementation of RISC-V Instruction Set Architecture for edge IoT computing platform," in International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Chennai, 2024.
[4] Ali and Wajid, "Exploring Instruction Set Architectural Variations: x86, ARM, and RISC-V in Compute-Intensive Applications," p. 5, 2023.
[5] C. E. Fang, L. T. Zheng and W. Qian, "Advanced Digital System Design using SoC FPGAs," IEEE ACCESS, vol. 11, p. 16, 2023.
[6] Jasmina Saidova, "RISC-V ARCHITECTURE AND ITS POLE IN THE NEAR FUTURE," Journal of Advanced Scientific Resarch (ISSN: 0976-9595), Vol.5. p. 54-67, 2024.
[7] J. -Y. Lai, C. -A. Chen, S. -L. Chen and C. -Y. Su, "Implement 32-bit RISC-V Architecture Processor using Verilog HDL," 2021 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Hualien City, Taiwan, 2021, pp. 1-2, doi: 10.1109/ISPACS51563.2021.9651130.
[8] T. J. David Ngu, "DESIGN AND SIMULATE RISC-V PROCESOR USING VERILOG," Master dissertation, Fac. Of Eng. & Green Tech., UTAR, Malaysia, August 2023.
[9] B. Maneesha, G. Sriram, G. Sai, M. Siddhu, and S. E. Jaya, "Design of RISC-V processor using Verilog," Int. J. Res. Publication and Reviews, vol. 5, no. 4, pp. 2097–2116, 2024. [Online]. Available: https://ijrpr.com/uploads/V5ISSUE4/IJRPR24834.pdf.
[10] G. Rajesh Babu, M. Bhanu Prakash, M. Vijaya Kumari, Ch. V. D. Ashok Kumar, and B. G. Sai, "Design of 32-bit asynchronous RISC-V processor using Verilog," J. Emerg. Technol. Innov. Res., vol. 7, no. 3, pp. 1717–1722, 2020. [Online]. Available: https://www.jetir.org/papers/JETIR2003247.pdf.
[11] Sandeep P., "A Novel 32 Bit RISC-V Based Pipelined Processor Design," Degree dissertation, Dept. of Elect. & Comm. Eng., School of E&E, Institute Science & Tech., Chennai, March 2022.
[12] L. T. Chong, "Low-Power Design of a Datapath Architecture Based on RISC-V Processor Core," Degree dissertation, Fac. of E & E Eng., Dept. Comp. Eng., UTHM, Parit Raja, B.P., Johor, Malaysia, June 2023.
[13] J. K. Goh and C. Uttraphan, "An enhanced UTHM RISC-V processor core architecture implemented on FPGA," Emerging Evolutionary Engineering Exploration (EEEE), vol. 5, no. 2, pp. 32–41, Nov. 2024. [Online]. Available: https://publisher.uthm.edu.my/periodicals/index.php/eeee/article/view/17719.