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

Vol. 5 No. 2 (2024) 32-41
https://publisher.uthm.edu.my/periodicals/index.php/eeee

EEEE

An Enhanced UTHM RISC-V Processor Core Architecture Implemented on FPGA

Goh Jun Kang[1], Chessda Uttraphan[1]*

[1] *Faculty of Electrical & Electronic Engineering,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, 86400, Johor, MALAYSIA

*Corresponding Author: chessda@uthm.edu.my
DOI: https://doi.org/10.30880/eeee.2024.05.02.005

Article Info

Received: 10 July 2024
Accepted: 19 September 2024
Available online: 30 October 2024

Keywords

RISC-V, processor architecture, Verilog, open source, FPGA

Abstract

RISC-V is a completely free and open instruction set architecture (ISA) known for its flexibility, enabling diverse hardware implementations. This paper presents the design and FPGA implementation of our version of a RISC-V processor core, named the UTHM-RISC-V processor. It covers the development of the datapath and control units for the processor, supporting up to 23 essential instructions based on the RV32I ISA. The processor core's architecture is developed using Verilog Hardware Description Language (HDL), with functional verification conducted using ModelSim. FPGA implementation is executed with Intel Quartus Prime. To validate the functionality of the proposed RISC-V processor on the FPGA, a simple test program, written in assembly language were used. Simulation results confirm that the design operates as intended, effectively executing all instructions. The compilation report indicates that the design is optimized, utilizing minimal logic elements on the implemented FPGA device (Intel Cyclone V), while the test programs confirm successful FPGA implementation. The outcomes support the development of a processor core suitable for embedded systems and educational purposes. Additionally, the low logic utilization contributes to reduced power consumption, making it ideal for low-power applications.

1. Introduction

Computer chips form the foundation of the information industry, with the instruction set architecture (ISA) being the key technology behind them. Over the past few decades, the x86 and ARM ISAs have dominated the market [1]. However, these architectures are proprietary and come with high licensing fees, which restrict access and flexibility for developers and manufacturers. Rather than licensing an ISA from x86 or ARM, there is another ISA option: RISC-V [2]. Developed by the University of California, Berkeley, RISC-V is a clean, simple, and open standard ISA ideal for low-power embedded systems and high-performance computers [3]. Its standout feature is its openness and flexibility, providing a royalty-free, standardized ISA that anyone can implement [4].

RISC-V is a versatile instruction set architecture (ISA) designed to be universally applicable across various computing devices [5]. Built on the principles of reduced instruction set computing (RISC), RISC-V simplifies microprocessor design by using simple, general-purpose instructions to achieve efficiency and performance gains compared to complex instruction set computing (CISC) [6]. Moreover, RISC-V is a load-store architecture, which means all arithmetic operations use operands from and produce results in addressable registers [7]. Operands are loaded from memory to registers before ALU operations, and results are stored in registers before being written to memory. This architecture minimizes memory interactions, enhancing instruction execution performance. On the other hand, during the design stage, the ISA is structured into a core base ISA and optional extensions. The base integer ISAs provide essential functionality for general-purpose computing in embedded processors, while

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 32-41

extensions enhance computational capabilities and support multiprocessing. This modular design allows for extensive customization, enabling processors to be tailored to specific applications and use cases. RISC-V offers four primary base ISAs: RV32I, RV32E, and RV64I. RV32I and RV64I are similar in user-visible features but differ in register width and memory address space, while RV32E is a variant of RV32I with a reduced register set. This work focuses exclusively on the development of RV32I base ISA.

The RV32I base ISA consists of 47 instructions with 32 bits length each. All these instructions can be classified into six formats: the four major formats (R, I, S, & U) and two variants (SB & UJ). Each of these formats serves a specific function and has a different encoding structure. For instance, the R format refers to register-to-register functions. Similarly, the I format operates similarly to R but involves sign extensions immediate values that minimizes the descriptive complexity of the ISA [8]. The S format is utilized for store operations, while the U format manages operations with a 20-bit immediate value. Conditional branch operations are encoded in the SB format, while the UJ format refers to unconditional jump operations.

FPGAs are semiconductor devices capable of reconfiguration based on user-developed HDL code. They consist of an array of programmable blocks interconnected via prefabricated routing tracks and programmable switches [9]. Typically used for implementing digital hardware circuits, FPGAs offer a departure from traditional circuit printing methods by enabling the implementation of entire systems without needing physical design, layout, fabrication, and verification stages. This results in reduced manufacturing costs and shorter development times. Moreover, their ability to be reprogrammed allows FPGAs to receive ongoing hardware upgrades at no additional cost [10].

The work aims to extend the design of the UTHM-RISC-V processor architecture proposed in our previous work [11] by extending its instruction set and implementing it onto an FPGA device. The previous design had included 13 instructions. However, to improve the processor's functionality and flexibility, this work will add 10 new instructions. Table 1 summarizes the 23 RV32I instructions implemented in the proposed design. The outcome will deliver a reliable RISC-V processor core suitable for low-power embedded system applications and educational purposes [12].

Table 1 *Instruction implemented in previous design & add-on instructions implemented in new design.*

Instruction Format	Instructions implemented in previous design	Add-on instructions implemented in new design
R-TYPE	ADD, SUB, OR, AND	XOR
I-TYPE	LW, ADDI, SLLI, SRLI	LB, ANDI, ORI, XORI, JALR
S-TYPE	SW	SB
U-TYPE	LUI, AUIPC	-
SB-TYPE	BEQ	BNE, BGE, BLT
UJ-TYPE	JAL	-

2. Related Works

This section reviews the related works on implementing RISC-V ISA on FPGA devices. It examines significant findings and insights from prior research and provides a comprehensive overview of the existing works. Md. Hasanul et al. collaborated on designing an educational processor based on RISC-V architecture, successfully implementing it on an FPGA [13]. The 32-bit processor was developed using Verilog HDL and deployed on a Xilinx Spartan 6 LX16 FPGA board. Designed primarily for computational tasks, the processor supports a limited set of arithmetic and logical operations, including AND, OR, ADD, SUB, SLT (Set on Less Than), SHL (Shift Left), SHR (Shift Right), and XOR. Its datapath architecture includes essential components such as the program counter (PC), register files (RF), arithmetic logic unit (ALU), and a memory unit, which functions as the instruction memory and data memory at the same time. Moreover, the work included an assembler developed in Python 3.0. This assembler's primary role is translating assembly language instructions into machine code compatible with the RISC-V architecture. Upon receiving assembly language instructions, the assembler breaks them into tokens to determine the required operations and data access patterns. It then generates machine code based on these parameters, which the RISC-V processor subsequently processes.

Ludovico et al. had designing and implementing a RISC-V processor core at the University of Essex, UK [14]. The processor is based on the RV64I instruction set architecture (ISA). To ensure that the design is lightweight enough to be implemented on even small FPGAs, the design focused on implementing a lightweight datapath architecture supporting only seven instructions. Those instruction sets include LD (Load Double Integer), SD (Store Double Integer), ADD, SUB, AND, OR, and BEQ (Branch if Equal). On the other hand, the datapath architecture of the 64-bit RISC-V processor is illustrated in Fig. 1, which only involves five components. After fully

designing the RISC-V processor core in System Verilog and successfully simulating its functionality, the team implemented it on an FPGA. Various tests were conducted to assess the processor's FPGA utilization. According to the results, the 64-bit RISC-V processor achieved a significant performance milestone of 2.276 DMIPs/MHz in the Dhrystone benchmark.

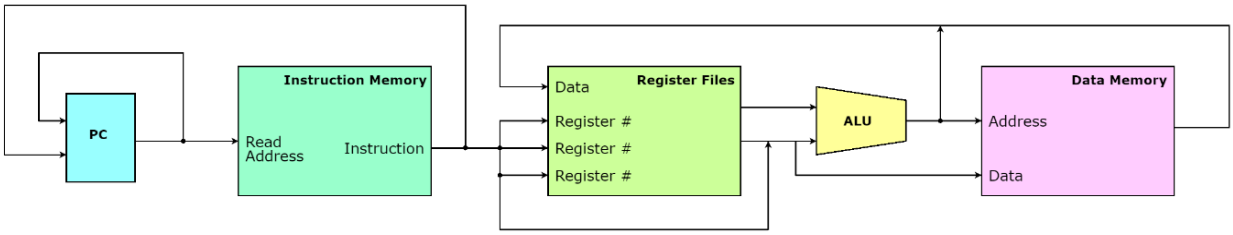

Fig. 1 *The 64-bit RISC-V processor architecture [14]*

Mr. Barriga introduces a structured methodology for developing RISC-V processor cores, illustrated through three design examples [15]. The first design consists of a description of the processor as a High-Level State Machine (HLSM), which mainly focuses on the behaviour of the processor instruction set. However, due to the large circuit area and long signal delays, this design is not suitable for hardware implementation. The general architecture of the second design comprises two blocks: the control unit and the data path. The control unit generates the control signal to the data path and external peripherals, while the data path contains the functional and storage elements required to process the data. The third design has a similar architecture to the second design but a separate memory map of instructions and data, allowing them to be accessed simultaneously and, hence, allowing the implementation of pipeline design. Furthermore, there is no centralized control unit in the third design. The control signal is generated by the instruction itself and wired directly to the corresponding data path blocks. Based on the hardware implementation results, the second design has the lowest FPGA resource utilization rate, while the third design achieves the highest throughput among the three proposed designs.

Tian Zheng et al. proposed a 32-bit RISC-V softcore processor that supports a base integer instruction set and multiplication division extension (RV32IM), along with configurable pipelining [16]. Extra features are added to the design to improve the reliability of the proposed work. For instance, hierarchical decoding is introduced to minimize FPGA resource consumption, while optimization schemes like shortening the critical path and reducing the stall cycle are used to improve the performance of the processor. The evaluation performed between the proposed work and the other four commercial softcore processors: MicroBlaze, PicoRV32, VexRiscv, and Taiga indicates that the proposed work has the lowest resources consumption rate but provides the second highest instructions per cycle (IPC), only 3% lower than Taiga, which is well-known as a high-performance processor.

At the National Institute of Technology, Hussain and Sarkar designed a 5-stage pipeline RISC-V processor that supports up to 12 RV32I instructions and implemented it on a Basys-3 FPGA board [17]. A hazard control unit is introduced to the processor's architecture to eliminate any possible pipeline hazard. For a single cycle processor (non-pipeline design), the processor achieved a maximum operating frequency of 31.6MHz, and the obtained power consumption is 244mW. On the other hand, the maximum operating frequency of the pipeline design can be 87.86MHz (2.8 times higher), on top of lower power consumption, 96mW.The comparison between the pipeline and single cycle (non-pipeline) design.

3. Proposed Design

3.1 RV32I ISA

As mentioned before, 23 instructions from RV32I base ISA were implemented in the design for the datapath architecture of the proposed RISC-V processor. The implemented RV32I instructions and their corresponding assembly formats are presented in Table 2.

Table 2 *Mnemonic & RTL operation of selected RV32I instructions*

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 32-41

Instruction	Mnemonic	RTL Operation
Load Byte	LB rd, imm12(rs1)	rd ← mem[rs1 + imm12] (7:0)
Load Word	LB rd, imm12(rs1)	rd ← mem[rs1 + imm12] (31:0)
Store Byte	SB, rs2, imm12(rs1)	rs2(7:0) → mem[rs1 + imm12]
Store Word	SW, rs2, imm12(rs1)	rs2(31:0) → mem[rs1 + imm12]
Addition	ADD rd, rs1, rs2	rd ← rs1 + rs2
Subtraction	SUB rd, rs1, rs2	rd ← rs1 - rs2
Logical XOR	XOR rd, rs1, rs2	rd ← rs1 ^ rs2
Logical OR	OR rd, rs1, rs2	rd ← rs1 | rs2
Logical AND	AND rd, rs1, rs2	rd ← rs1 & rs2
Add Immediate	ADDI rd, rs1, imm12	rd ← rs1 + imm12
Logical XOR Immediate	XORI rd, rs1, imm12	rd ← rs1 ^ imm12
Logical OR Immediate	ORI rd, rs1, imm12	rd ← rs1 | imm12
Logical AND Immediate	ANDI rd, rs1, imm12	rd ← rs1 & imm12
Shift Left Logical Immediate	SLLI rd, rs1, shamt	rd ← rs1 << shamt
Shift Right Logical Immediate	SRLI rd, rs1, shamt	rd ← rs1 >> shamt
Branch if Equal	BEQ rs1, rs2, imm12	if rs1 = rs2, PC ← PC + imm12
Branch if Not Equal	BNE rs1, rs2, imm12	if rs1 ≠ rs2, PC ← PC + imm12
Branch if Less Than	BLT rs1, rs2, imm12	if rs1 < rs2, PC ← PC + imm12
Branch if Greater or Equal	BGE rs1, rs2, imm12	if rs1 ≥ rs2, PC ← PC + imm12
Load Upper Immediate	LUI rd, imm20	rd ← imm20 << 12
Add Upper Immediate to PC	AUIPC rd, imm20	rd ← PC + imm20 << 12
Jump and Link	JAL rd, imm20	rd ← PC + 4, PC ← PC + imm20
Jump and Link Register	JALR rd, im12(rs1)	rd ← PC + 4, PC ← rs1 + imm12

3.2 Top-Level Module of the Designed RISC-V Processor

The top module of the designed RISC-V processor comprises two submodules: the datapath unit and the control unit. These submodules interact closely with each other to interpret, process, and execute instructions from the program running on the processor. Fig. 2 illustrates the block diagram for the top-level module of the proposed RISC-V processor.

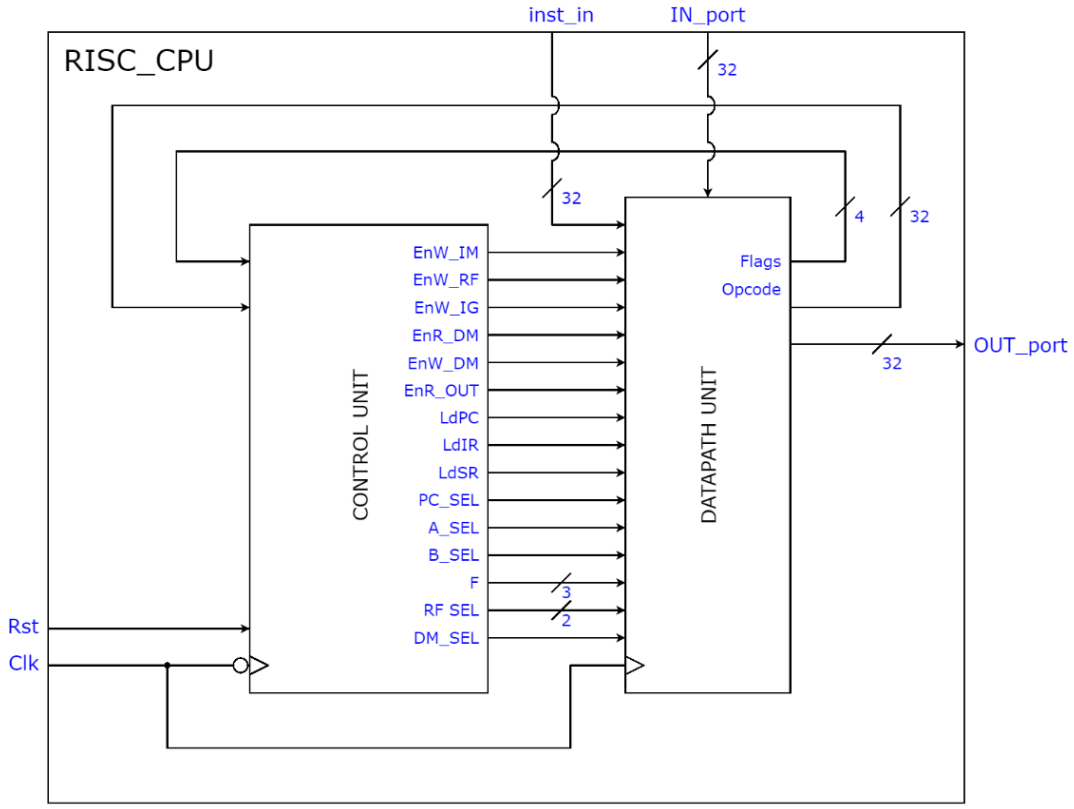

Fig. 2 *Block diagram for the top module of the proposed RISC-V processor*

3.3 Design of the Datapath Unit

The datapath unit consists of all the components required to execute the selected RV32I instructions. Each element is implemented as a Verilog module, emphasizing coding conventions and high readability. A Verilog testbench is developed for each component, enabling the datapath unit's behaviour to be verified in simulation. Fig. 3 depicts the block diagram for the datapath of the proposed RISC-V processor.

The program counter (PC) keeps track of the memory address of the next instruction to be executed. On the other hand, the instruction memory (IM) stores the machine codes of the instructions that form the program running on the processor. In contrast, the instruction register (IR) temporarily holds the binary representation of the currently executed instruction. During the execution of an instruction, the instruction is fetched from the IM into the IR based on the memory address determined by the PC. Simultaneously, the PC is incremented by 4, pointing to the next sequential instruction. The IM consists of 256 cells with 8-bit width each, providing a memory size of 256 bytes.

The Arithmetic Logic Unit (ALU) executes arithmetic, logic, and shift operations on binary data. An 8-to-1 multiplexer is implemented in the ALU to control the operations performed. The status register (SR) shows the status of the processor when executing arithmetic instructions and provides the conditions required by branch instructions. The immediate generator (IM) produces constant values that serve as operands for computational operations and act as the offset value for memory access and control flow operations. All the generated values are sign-extended.

The data memory (DM) serves to store data during program execution. Like the IM, the size of the DM is also defined as 256 bytes. In addition, specific memory cells of the DM are mapped to the output and input ports of the processor, allowing data transmission between the processor and external peripherals. The register files (RF) serve two functions in the datapath unit. One function will be to supply the ALU with operands required for the operation. Another function is to hold data transferred from the DM, results generated from the ALU, and the instruction address from the PC. It contains 32 general-purpose registers, with each register being 32-bit wide.

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 32-41

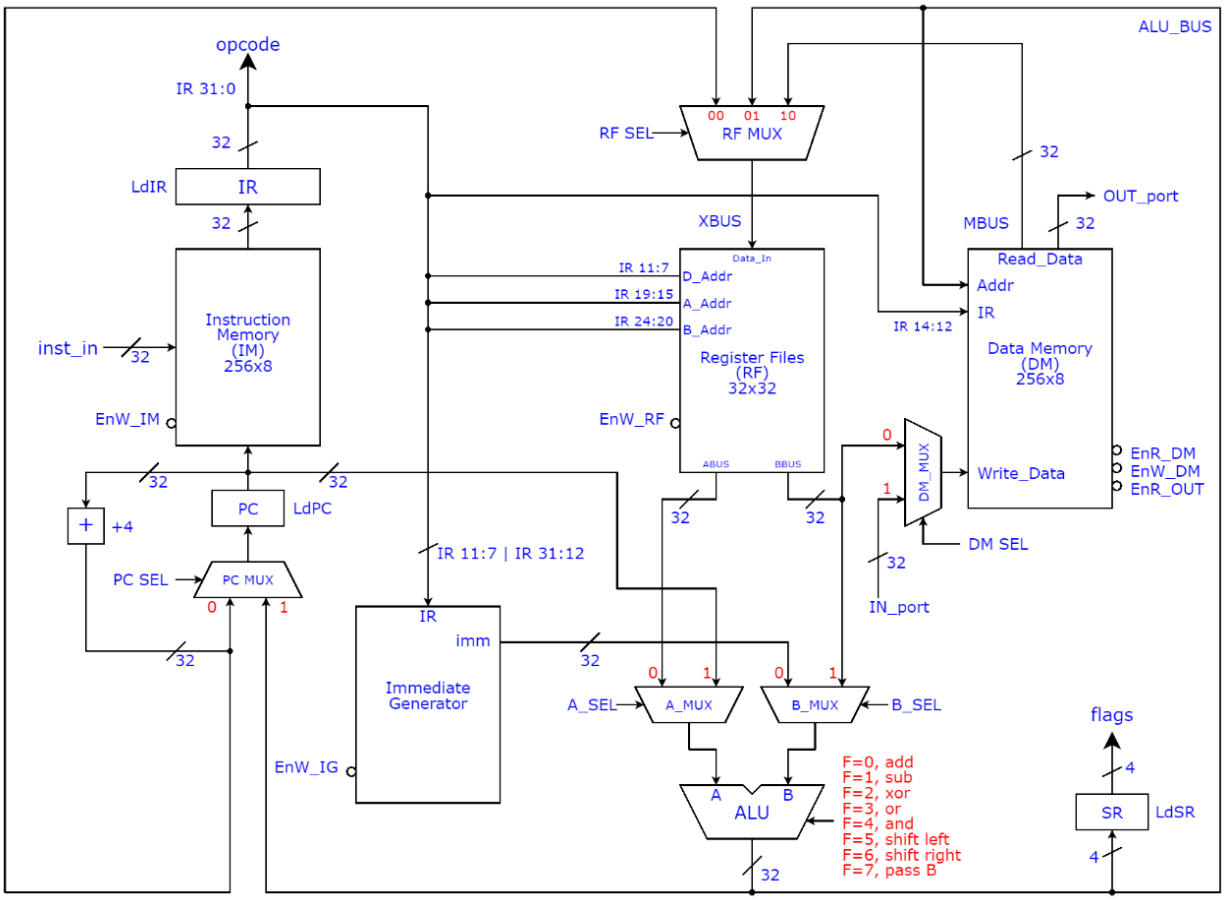

Fig. 3 *Block diagram for the datapath of the proposed RISC-V processor*

3.4 Design of the Control Unit

The control unit generates control signals to control the datapath operations, ensuring each operates in a well-defined sequence and maintains the correct flow of data and operations within the processor. In the proposed design, the control unit is implemented as a finite state machine (FSM) [18-19], with each state corresponding to a specific stage in the instruction execution process. The current state and inputs, including the instruction opcode and status flags, determine state transitions. Each state generates a specific set of control signals required for the corresponding operation. Fig. 4 depicts the block diagram of the designed control unit.

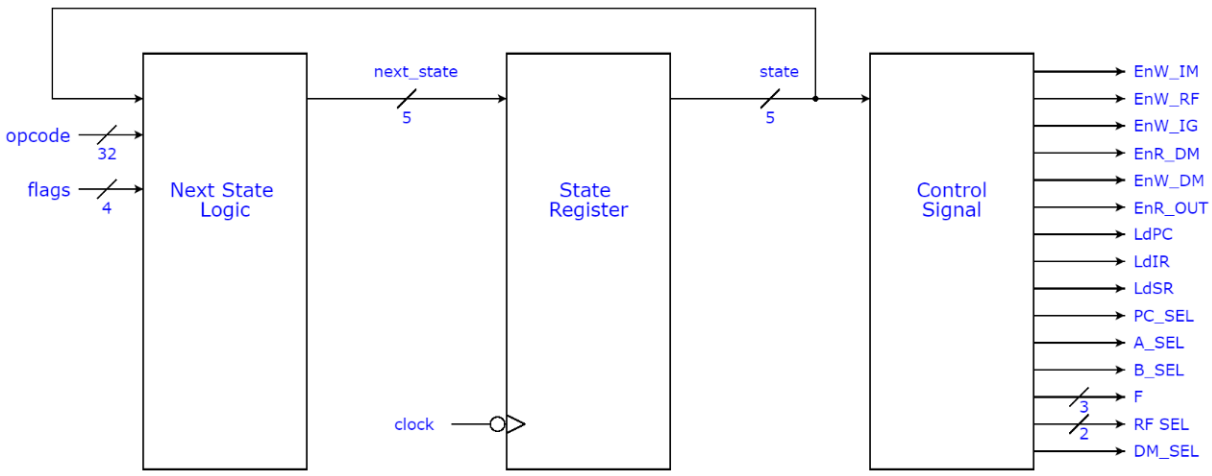

Fig. 4 *Block diagram for the control unit of the proposed RISC-V processor.*

3.5 FPGA Implementation

Once the design of the proposed RISC-V processor is completed and the simulation for all selected RV32I instructions is successful, the proposed design can be programmed onto the target FPGA device, which is the Intel Cyclone V FPGA on the DE1-SoC development board [20]. Moreover, a Verilog module for the BCD to the 7-segment decoder is developed to utilize the 7-segment display available on the DE1-SoC board.

4. Results & Discussion

Fig. 5 shows the output waveform for the simulation via ModelSim. Initially, the PC is reset to zero, followed by the fetch of the first instruction from the IM, which is SB x4, #3(x3). After decoding, the SB instruction is executed. The value in register x3, 0x00000003, is added to the immediate value #3, generating the effective memory address #6. This means the least significant byte of register x4 is stored in the data memory location #6. For the next instruction, SUB x9, x5, x2, subtraction is carried by subtracting the content of x5 (0x0000000a) to x2 (0x0000000a) which will produce 0x00000000 and be written to register x9. For the third instruction, for the instruction ADDI x8, x5, #7, a constant value, 7, is added to the content of register x5. The result of the operation which is 0x00000011 is then be written to register x8, replacing the initial value, 0x0A0B0C00.

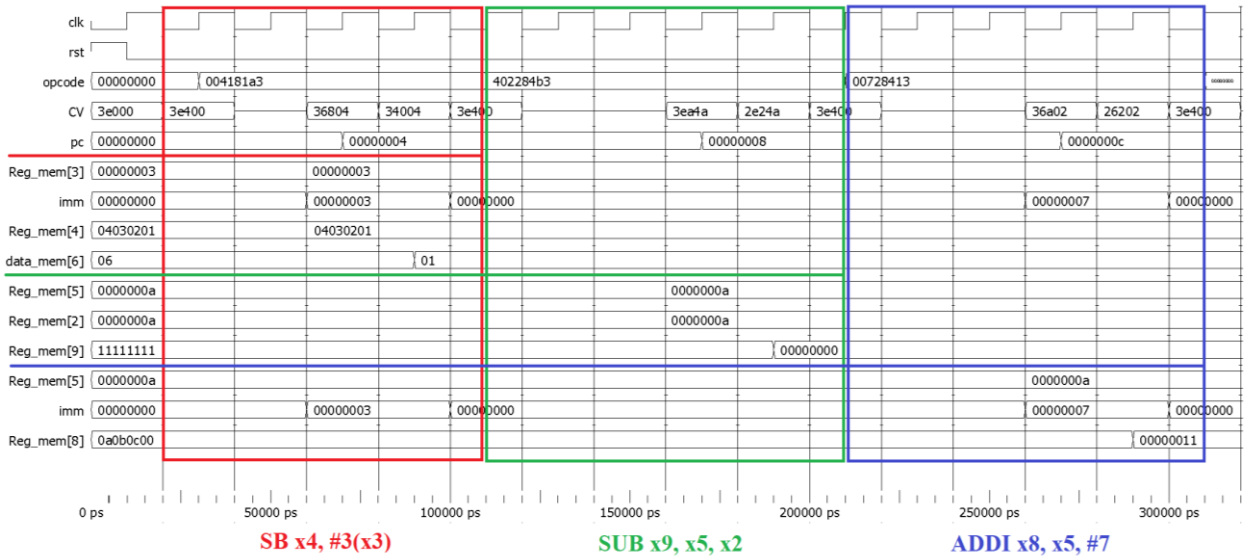

Fig. 5 *Output waveform for the simulation of is SB x4, #3(x3), SUB x9, x5, x2, and ADDI x8, x5, #7*

Upon the design of the proposed RISC-V processor is complete and the successful simulations for all the selected RV32I instructions, a simple program was designed and run on the FPGA implementation to ensure its functionality. Table 3 presents the assembly program for an 8-bit binary counter, while Fig 6 illustrates the program test outcome.

Table 3 *The 8-bit binary counter assembly program*

Address	Instruction	Machine Code	Operation
0	ANDI x1, x1, #0	0x0000F093	Initialization
4	ANDI x2, x2, #0	0x00017113	
8	ADDI x2, x2, #255	0x0FF10113	Define sequence
12	SW x1, #200 (x0)	0x0C102423	Increment by 1
16	ADDI x1, x1, #1	0x00108093	
20	ANDI x5, x5, #0	0x0002F293	Delay routine
24	ANDI x6, x6, #0	0x00037313	
28	ADDI x6, x6, #58	0x03A30313	
32	SLLI x6, x6, #16	0x01031313	

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 32-41

36	BEQ x5, x6, #8	0x00628463	
40	ADDI x5, x5, #1	0x00128293	
44	JAL x3, #-12	0xFF5FF1EF	
48	BEQ x1, x2, #4	0x00208263	Sequence end?
52	JAL x3, #-44	0xFD5FF1EF	
56	JALR x4, #0(x0)	0x00000267	Loop back

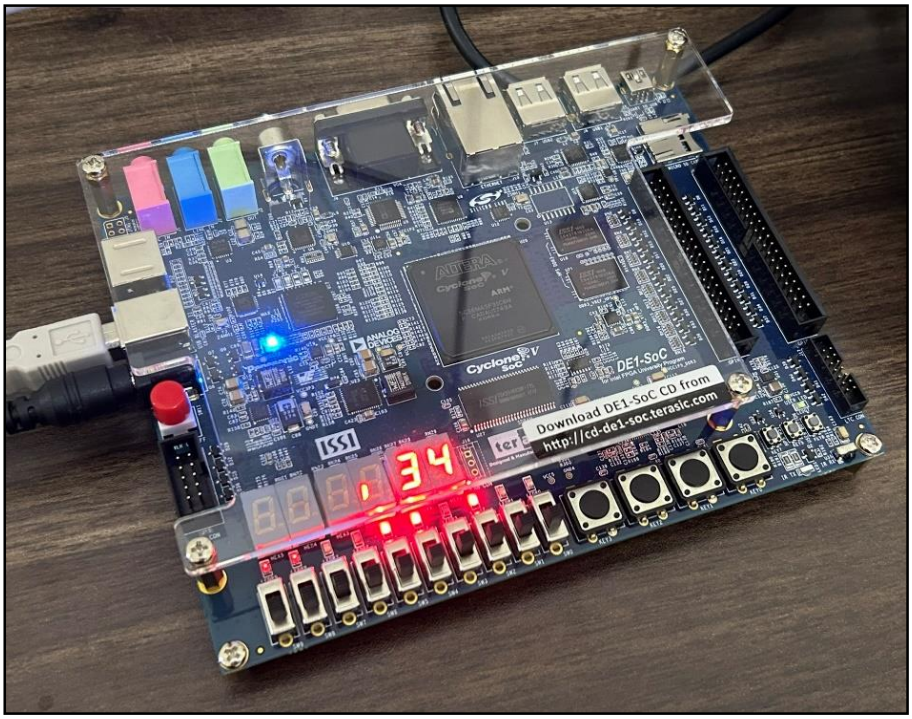

Fig. 6 *8-bit binary counter program test outcome*

Table 4 summarizes the resource utilization, maximum operating frequency, and total power consumption obtained for the proposed RISC-V processor. With a 50MHz clock signal, the power consumption is 424.45mW and achieved a maximum operating frequency of 30.35MHz. From the perspective of resource utilization, the complete design of the proposed RISC-V processor occupies 5,796 logic elements, 116 I/O pins, and 2,048 block memory, which is relatively low compared to the total available resources.

Table 4 *Resource utilization, maximum operating frequency, and total power consumption obtained for the proposed RISC-V processor*

Resource Utilization	Logic Elements	5,796 / 32,070 (18%)
	I/O Pins	116 / 457 (25%)
	Block Memory	2,048 / 4,065, 280 (<1%)
Maximum Operating Frequency	30.35 MHz	
Total Power Consumption	424.45 mW	

5. Conclusions

In conclusion, the UTHM-RISC-V processor core supporting up to 23 RV32I instructions is successfully designed and implemented on a Field Programmable Gate Array (FPGA). Simulation results validated the functionality of each processor element, ensuring accurate execution of a wide range of instructions. Additionally, FPGA testing demonstrated that the design performs efficiently with relatively low resource utilization leading to low power consumption processor. Pipelining can be implemented in the proposed design to improve the processor's

throughput. Furthermore, extensions like compiler and assembler can be added to the proposed design to convert the high-level language and assembly language into machine code in RISC-V instruction format, simplifying the assembly program development process. Moreover, the RV32I design can be replaced with alternative RISC-V ISA, such as RV64I and RV128I, to meet the requirements for greater data size application.

Acknowledgement

Execution of this work is made possible through the support of various staff from Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia.

Conflict of Interest

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

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Goh Jun Kang and Chessda Uttraphan;* ***data collection:*** *GOH JUN KANG;* ***analysis and interpretation of results:*** *Goh Jun Kang and Chessda Uttraphan;* ***draft manuscript preparation:*** *Goh Jun Kang. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Cui, E., Li, T., & Wei, Q. (2023). RISC-V instruction set architecture extensions: A survey. *IEEE Access, 11*, 24696-24711, https://doi.org/10.1109/2023.3246491

[2] G. S. Nicholas, Y. Gui. & F. Saqib (2020). A survey and analysis on SoC platform security in ARM, Intel and RISC-V architecture, *IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 63,* 718-721, https://doi.org/10.1109/MWSCAS48704.2020.9184573

[3] A. Waterman (2016). *Design of the RISC-V Instruction Set Architecture*. University of California at Berkeley. Retrieved from http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.html

[4] Asanović, K., & Patterson, D. A. (2014). *Instruction sets should be free: The case for RISC-V*. University of California, Berkeley. Retrieved from http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.html

[5] Waterman, A., & Asanović, K. A. (2017). *The RISC-V Instruction Set Manual Volume I: User-Level ISA Document Version 2.2*. University of California, Berkeley. Retrieved from https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-118.pdf

[6] Patterson, D. A., & Hennessy, J. L. (2018). *Computer organization and design: The hardware/software interface: RISC-V Edition*. Morgan Kaufmann Publishers Inc. Retrieved from http://home.ustc.edu.cn/~louwenqi/reference_books_tools/Computer%20Organization%20and%20Design%20RISC-V%20edition.pdf

[7] Atkins, & Ong, S.-C. (1979). Time-component complexity of two approaches to multioperand binary addition. *IEEE Transactions on Computers, 28*(12), 918-926. https://doi.org/10.1109/TC.1979.1675281

[8] He, Y., & Chen, X. (2023, August). *Survey and comparison of pipeline of some RISC and CISC system architectures.* In Proceedings of the International Conference on Computer and Communication Systems (ICCCS) (Vol. 8, pp. 785-790). https://doi.org/10.1109/ICCCS57501.2023.10150975

[9] Boutros, A., & Betz, V. (2021, April). FPGA architecture: Principles and progression. *IEEE Circuits and Systems Magazine, 21(2),* 4–29.

[10] Gandhare, S., & Karthikeyan, B. (2019, February). *Survey on FPGA architecture and recent applications.* In International Conference on Vision Towards Emerging Trends in Communication and Networking (pp. 1-4). https://doi.org/10.1109/ViTECoN.2019.8899550

[11] Ting, C. L. (2023). *Low-power design of a datapath architecture based on RISC-V processor core*. Universiti Tun Hussein Onn Malaysia.

[12] Eassa, H., Adly, I., & Issa, H. H. (2019, December). *RISC-V based implementation of programmable logic controller on FPGA for Industry 4.0.* In 2019 31st International Conference on Microelectronics (ICM) (pp. 98-102). https://doi.org/10.1109/ICM48031.2019.9021939

[13] Saif, M. H. B., Sadad, N. U., & Mondal, M. N. I. (2023, April). *FPGA implementation of educational RISC-V processor suitable for embedded applications.* In Proceedings of the International Conference on Electrical, Computer and Communication Engineering (ECCE) (pp. 1-5). https://doi.org/10.1109/ECCE57851.2023.10101508.

[14] Poli, L., Saha, S., Zhai, X., & McDonald-Maier, K. D. (2021, June). *Design and implementation of a RISC-V processor on FPGA.* In 2021 International Conference on Mobility, Sensing and Networking (MSN) (pp. 161-166). https://doi.org/10.1109/MSN53354.2021.00037
[15] Barriga, A. (2020). *RISC-V processors design: A methodology for cores development.* In Conference on Design of Circuits and Integrated Systems (DCIS) (pp. 1-6). https://doi.org/10.1109/DCIS51330.2020.9268639
[16] Zheng, T., Cai, G., & Huang, Z. (2022). A soft RISC-V processor IP with high-performance and low-resource consumption for FPGA, *IEEE International Symposium on Circuits and Systems (ISCAS), 2022,* 2538-2541, https://doi.org/10.1109/ISCAS48785.2022.9937742
[17] Hussain, F., & Sarkar, S. (2024). *Design and FPGA implementation of five stage pipelined RISC-V processor.* In 2024 IEEE 9th International Conference for Convergence in Technology (I2CT) (pp. 1-6). https://doi.org/10.1109/I2CT61223.2024.10544184
[18] Alsubaei, S., Qaisar, S. M., & Alhalabi, W. (2017). *A VHDL based Moore and Mealy FSM example for education.* In 2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP) (Vol. 2, pp. 456-459). https://doi.org/10.1109/SIPROCESS.2017.8124583
[19] Wilson, P., & Mantooth, H. A. (2013). Block diagram modeling and system analysis. In P. Wilson & H. A. Mantooth (Eds.), *Model-based engineering for complex electronic systems* (pp. 169-196). Newnes. https://doi.org/10.1016/B978-0-12-385085-0.00006-3
[20] Terasic Technologies Inc. (2003-2013). *DE1-SoC user manual*. Retrieved from https://www.merriam-webster.com/dictionary/manual