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

EEEE

Vol. 5 No. 2 (2024) 369-377
https://publisher.uthm.edu.my/periodicals/index.php/eeee

Design and Implementation of Non-Pipelined RISC Processor for Educational Purpose

Lau Kah Chun[1], Munirah Ab. Rahman[1]*, Chessda Uttraphan Eh Kan[1]

[1] *Faculty of Electrical and Electronic Engineering*
Universiti Tun Hussien Onn Malaysia, Johor 86400, Malaysia

*Corresponding Author: munira@uthm.edu.my
DOI: https://doi.org/10.30880/eeee.2024.05.02.044

Article Info

Received: 5 July 2024
Accepted: 29 October 2024
Available online: 30 October 2024

Keywords

Non-Pipelined, RISC, Education Purpose

Abstract

The fast pace of change in the 21st century is driving a significant shift in education, with old-school teaching methods giving way to newer approaches. Shallow understanding and decreased interest among students in the subjects such as computer architecture, which partly due to insufficient learning material and outdated teaching method, have become new challenges for educators. To address this issue, this study aims to design and implement a non-pipelined RISC processor for educational use in 'Computer Architecture' and 'Microprocessor and Microcontroller' subjects at Universiti Tun Hussein Onn Malaysia (UTHM). The 8-bit non-pipelined RISC processor which can support 16 types of instructions, is designed using Intel Quartus software and Verilog HDL, with functionality verification and simulation performed in ModelSim-Altera using various sample programs. The obtained simulation results indicate that the instructions are executed sequentially without overlapping, aligning with non-pipelining processing principles. This study is expected to enrich the educational resources in UTHM, deepening student's understanding of computer architecture, especially non-pipelining concept, and potentially boosting their academic results and engagement.

1. Introduction

Computer architectures are typically categorized into Reduced Instruction Set Computer (RISC) and Complex Instruction Set Computer (CISC). CISC architectures have a large set of powerful instructions capable of performing complex tasks but often lead to slower operation due to their complex design [1]. In contrast, RISC processors utilize a simpler set of instructions, which allows for more efficient execution through extensive pipelining and optimized compilers [2]. This simplicity allows RISC processors to execute each instruction within a single machine cycle, resulting in faster performance. Moreover, RISC are less costly and time-consuming to develop, enabling quicker updates and incorporation of the latest VLSI technology [3].

In both CISC and RISC processor, instruction execution can be carried out through two methods: non-pipelining and pipelining. In non-pipelining, the instructions are executed sequentially, with each instruction going through all stages before the next on starts, which results in a simple design but slower performance due to multiple cycles per instructions [4]. On the other hand, pipelining processor processes multiple instruction concurrently by overlapping their execution stages, enhancing overall throughput, and boosting performance but adding complexity to control unit to handle the concurrent stages and potential execution hazards [5-6]. In short, non-pipelining sacrifices speed to simply the design and control, whereas pipelining offers efficiency gains despite its added complexity.

Computer architecture is an important and widely taught subject in universities worldwide. It introduces students to the basic computer structures, CPU design, I/O systems, memory systems, non-pipelining and pipelining concept, and more. The course typically includes lectures, lab work, assignment, and projects, allowing students to acquire both theoretical knowledge and practical experience. Despite its importance, computer architecture is often perceived as a difficult subject due to the extensive range of computer components that need to be studied, understood, and memorized, such as processors, hard drives, and memory.

At Universiti Tun Hussein Onn Malaysia (UTHM), similar issue is observed. Not only computer architecture, the outcome from other subjects such as 'Digital Design' and 'Microprocessor and Microcontroller (MPMC)' are falling short of lecturers' expectations. The lack of adequate learning material and old-school teaching method seem to be the main reasons for students' lack of comprehension and interest. Therefore, to address this issue, this study aims to design and implement an 8-bit 5 stages non-pipelined RISC processor for educational purpose. This non-pipelined processor is anticipated to serve as a new and valuable learning material for 'Computer Architecture' and MPMC subject in the future.

2. Methodology

2.1 Instruction Set Architecture

Three types of instruction format are used in this non-pipelined RISC processor: R-type (Register Type), I-type (Immediate Type), and J-type (Jump/Branch Type) as illustrated in Fig. 1.

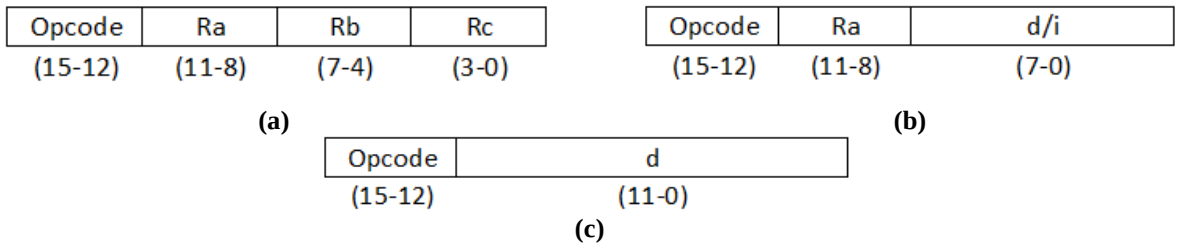


Fig. 1 *Instruction formats (a) R-type (Register-type); (b) I-type (Immediate-type); (c) J-type (Jump/Branch-type)*

Fig. 2 (a) illustrates the instruction format of Register-type which has a total of four fields. Opcode width is of 4-bit, which is used to determine the type of instruction. Destination registers (Ra) and two source registers (Rb and Rc) are indicated by 4-bit fields. The instruction which are supported by this instruction format are ADD, SUB, AND, OR, XOR, ROR, and ROL.

Fig. 2 (b) shows the instruction format of Immediate-type. There are a total of three fields in this format. Similar to R-type, the Opcode and destination register are of 4-bit width. Meanwhile, there is another field which used for immediate data (i) or address (d) and has an 8-bit width. The instruction which are supported by I-type instruction format are LD, ST, and MOV.

Fig. 2 (c) depicts the Jump/Branch type instruction format. In this format, there are only two fields, an Opcode with 4-bit width and a 12-bit width field which called address (d). The instruction which are supported by this instruction format are BRA, BRZ, BRNZ, BRGT, and BRLT.

The designed pipelined processor can support up to 16 types of instructions, including addition (ADD), subtraction (SUB), logical AND (AND), logical OR (OR), logical XOR (XOR), Rotate Right by 1-bit (ROR), Rotate Left by 1-bit (ROL), Load (LD), Store (ST), Move Immediate (MOV), unconditional branching (BRA), Branch if zero (BRZ), Branch if not equal zero (BRNZ), Branch if greater than (BRGT), and Branch if lower than (BRLT).The instruction set of the processor is listed down in Table 1.

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 369-377

Table 1 *Instruction set of 8-bit 5 stages pipelined RISC processor*

No.	Assembler Mnemonics	Operation
1	LD Ra, d	Load data from memory address **d** into register **Ra**
2	ST d, Ra	Store data from register **Ra** into register **Ra**
3	Mov Ra, #i	Move immediate complement value into register Ra
4	ADD Ra, Rb, Rc	ADD the content in register **Rb** with **Rc** and store the result in register **Ra**
5	SUB Ra, Rb, Rc	Subtract the content in register **Rb** with **Rc** and store the result in register **Ra**
6	BRA d1	Branch to address **d1**
7	BRZ d2	Branch to address **d2** if Z flag equals 1
8	BRNZ d3	Branch to address **d3** if Z flag equals 0
9	BRGT d4	Branch to address **d4** if N flag equals 0
10	BRLT d5	Branch to address **d5** if N flag equals 1
11	ROL Ra, Rb	Rotate the content in register **Rb** left by 1 bit and store the result in register **Ra**
12	ROR Ra, Rb	Rotate the content in register **Rb** right by 1 bit and store the result in register **Ra**
13	AND Ra, Rb, Rc	AND the content in register **Rb** with **Rc** and store the result in register **Ra**
14	OR Ra, Rb, Rc	OR the content in register **Rb** with **Rc** and store the result in register **Ra**
15	XOR Ra, Rb, Rc	XOR the content in register **Rb** with **Rc** and store the result in register **Ra**
16	NOP	No Operation

2.2 Harvard Architecture

The non-pipelined RISC processor is designed based on Harvard architecture where the data and instruction are stored in different memory space separately with an independent bus connection. The Harvard architecture and pipeline architecture are illustrated in Fig. 2.

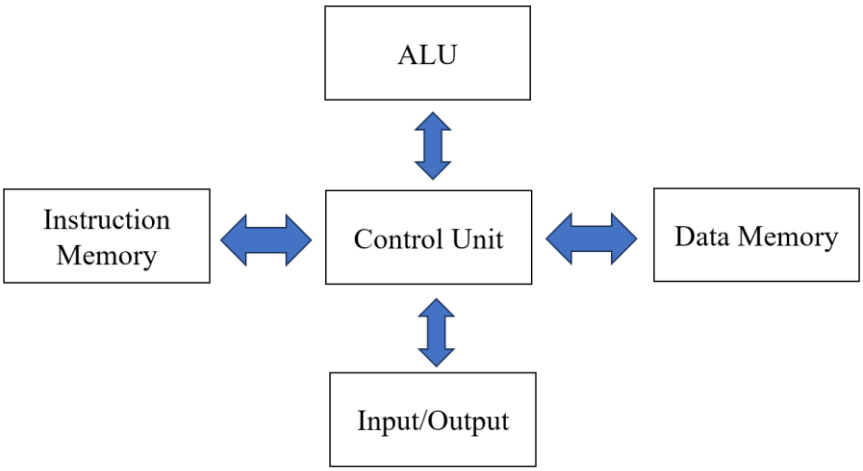


Fig. 2 *Harvard architecture [7]*

3. Result and Discussion

3.1 Design of Non-Pipelined Processor

An 8-bit non-pipelined RISC processor is designed by using Intel Quartus Prime 20.1 Lite Edition and Verilog HDL. The non-pipeline processor is a multi-cycle processor where different instructions can take a different number of cycles, allowing the processor to handle complex instructions more efficiency. Its design is consisting of various components such as Program Counter (PC), Instruction Memory (IM), Instruction Register (IR), Control Unit (CU), Register File (RF), an Arithmetic Logical Unit (ALU) which can operate arithmetic, logical, and rotate instructions, Status Register (SR), Data memory (DM), an increment, and two multiplexers. Its block diagram and RTL view are illustrated in Fig.3 and Fig. 4 respectively.

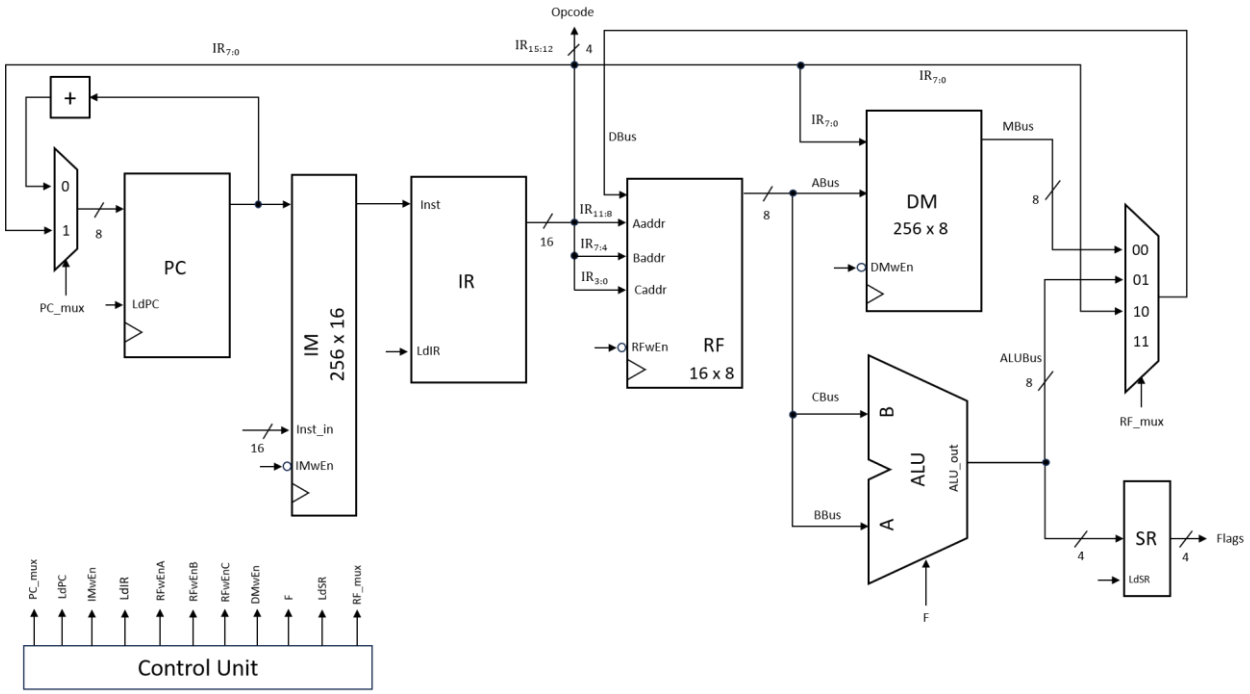


Fig. 3 *Block diagram of the 8-bit non-pipelined RISC processor design*

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 369-377

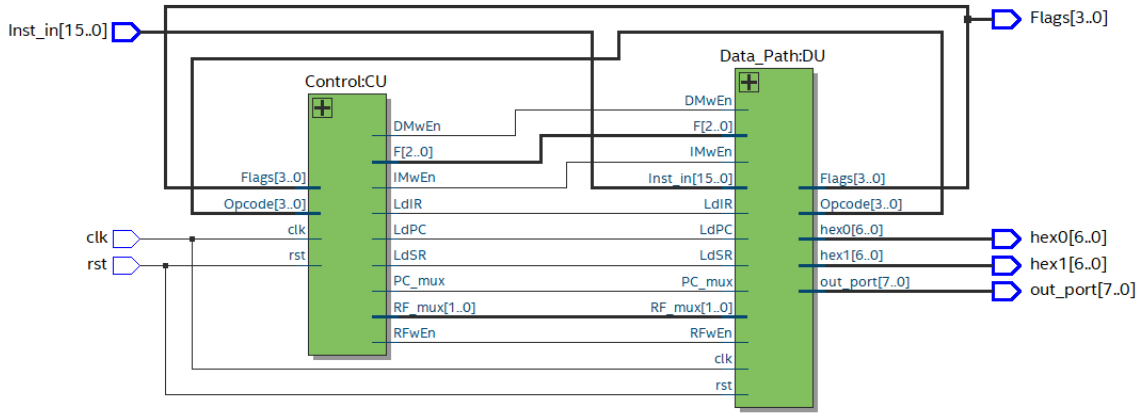
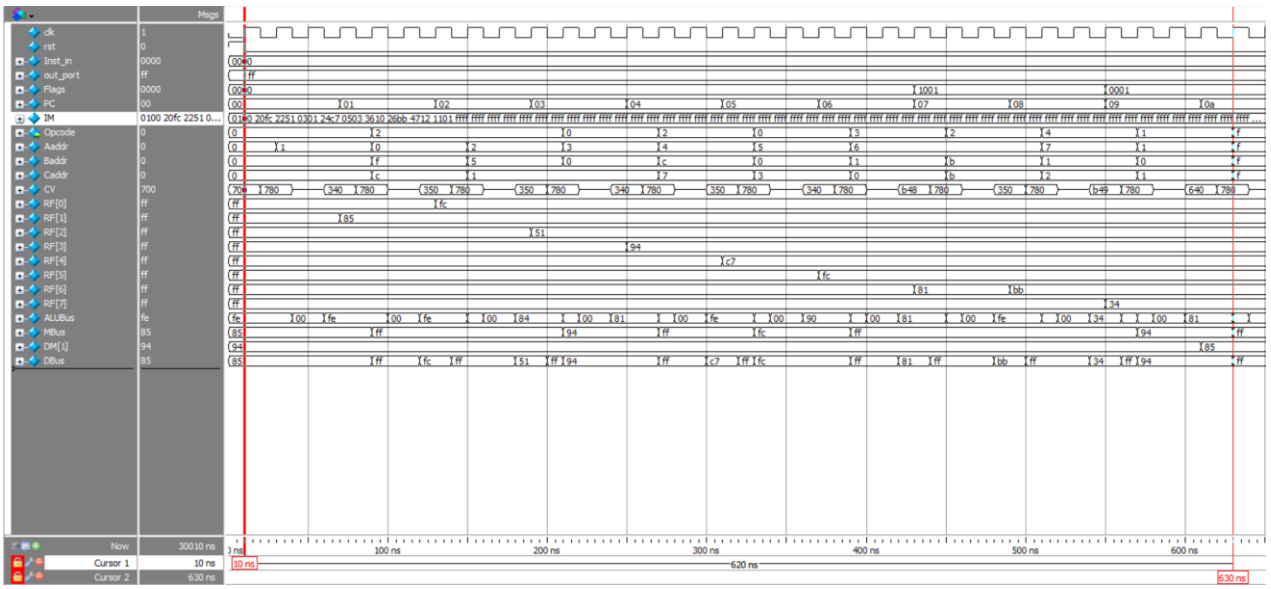


Fig. 4 *RTL view of the 8-bit non-pipelined RISC processor's top-level design*

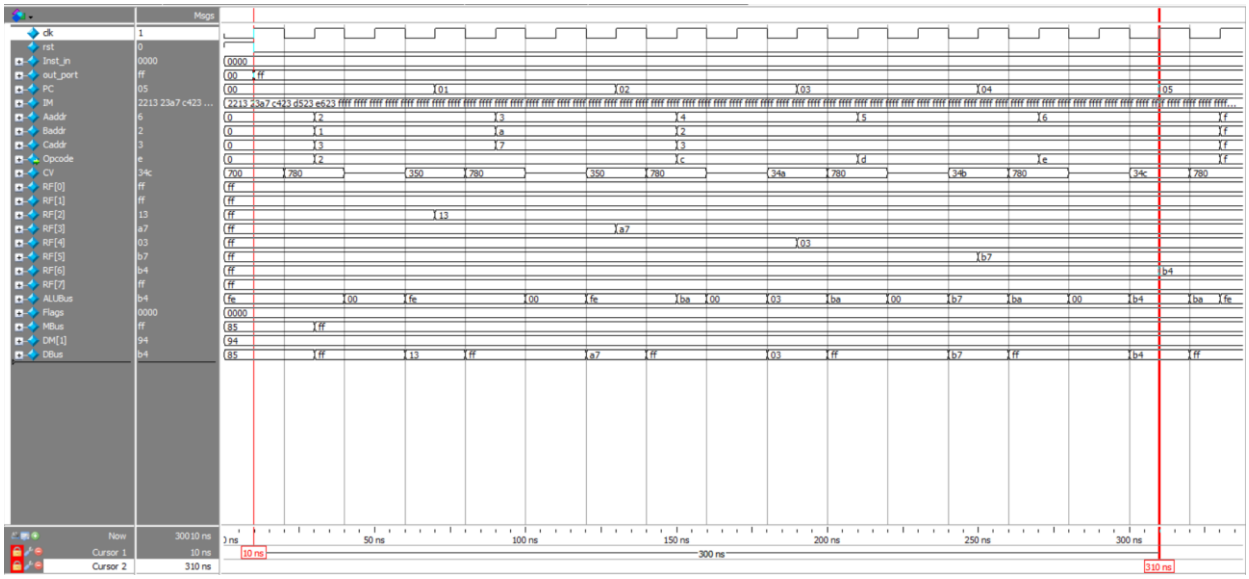
3.2 Simulation Results

The functionally verification and performance of the designed non-pipelined processor are simulated in ModelSim-Altera simulator using the 5 types of sample programs which including all the 16 instructions. The simulation results of all the five sample programs are illustrated in Fig. 5. Based on the results, it shows that for non-pipelined processor, the instructions are executed sequentially, one after one, without overlapping, which indicates that the designed processor consistent with the behavior of non-pipelining processor.

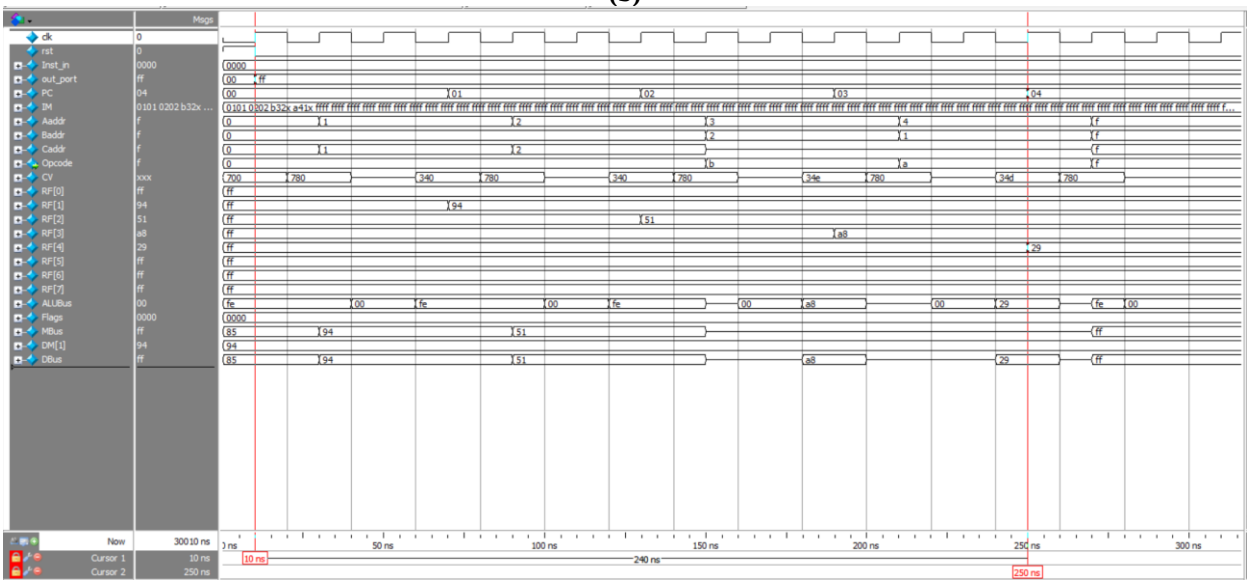


(a)

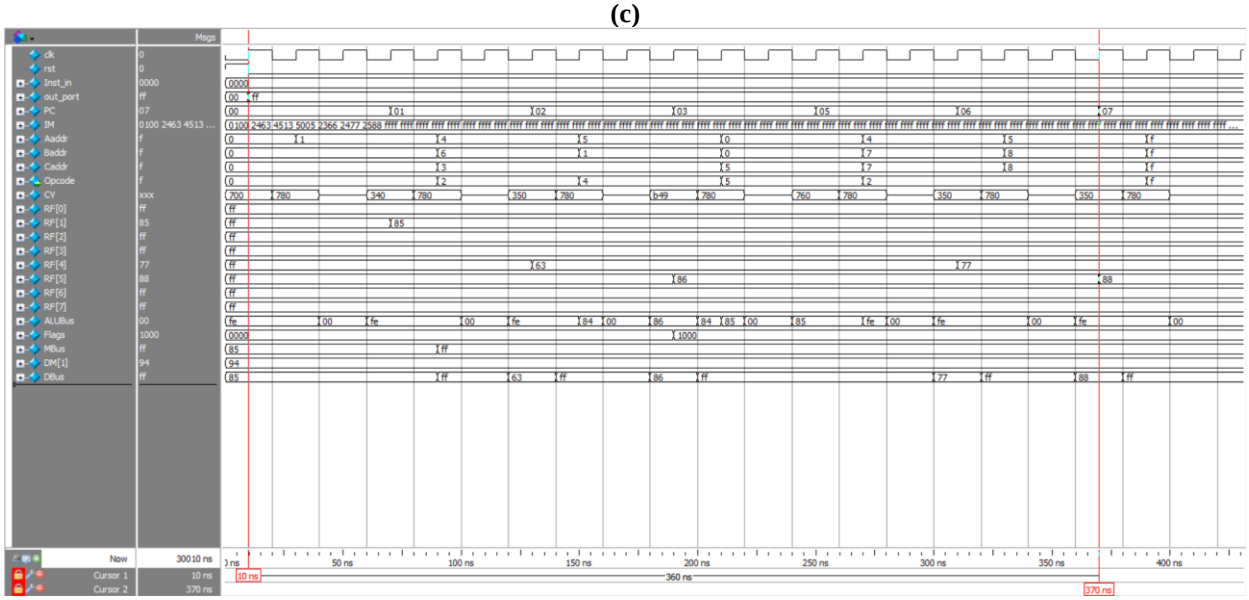
Fig. 5 *Simulation results of non-pipelined processor (a) Sample program 1; (b) Sample program 2; (c) Sample program 3; (d) Sample program 4; (e) Sample program 5*



(b)



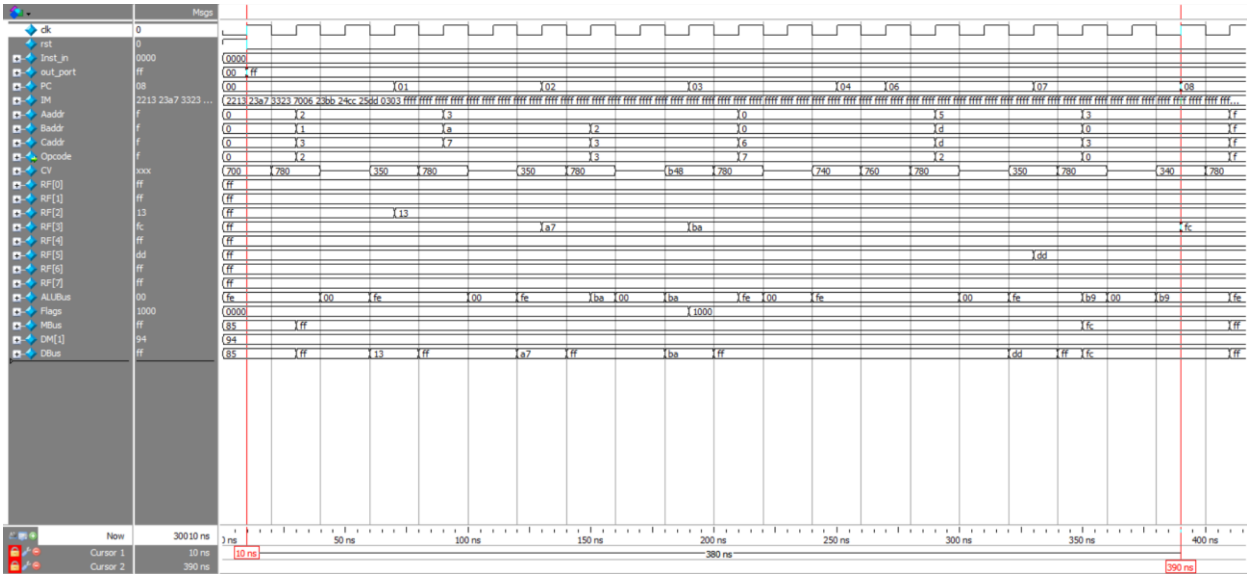
(c)



(d)

Fig. 5 *continued*

Evolution in Electrical and Electronic Engineering Vol. 5 No. 2 (2024) p. 369-377



(e)

Fig. 5 *continued*

The performance of the designed non-pipelined processor is illustrated in Table 2 in term of the types of sample program, total number of instructions in the program, total program instruction execution time, clock period, total number of clock cycle required to execution the program, and Cycle Per Instruction (CPI).

Table 2 *Summary of non-pipelined processor's performance*

RISC Processor	Sample Program	No. of Instruction Count	Program Execution Time	Clock Period	No. of Clock Cycles	CPI
Non-pipelined	Sample 1	10	620 ns	20 ns	31	3.1
	Sample 2	5	300 ns	20 ns	15	3
	Sample 3	4	240 ns	20 ns	12	2
	Sample 4	7	360 ns	20 ns	18	2.571
	Sample 5	8	380 ns	20 ns	19	2.375

Based on the summarization, it shows that the total number of instructions in a program has a relationship with the CPI. The more the instructions in a program, the higher the CPI of the processor for that program. It is due to the non-pipelining concept where the instruction execution sequentially, without overlapping, which results in lower throughput and efficiency.

Additionally, comparing with the other related works such as [8-11], although the designed processor may not surpass them in certain metrics, its primary focus on educational utility is evident. Prioritizing simplicity, this processor facilitates easy understanding of fundamental computer architecture concepts, especially non-pipelining. Its straightforward design allows students, especially those new to processor architecture, to grasp the concepts and gain hands-on experience in processor design.

3.3 FPGA Testing

Two programs, 'Binary Counter' and 'Blinking LED', has been developed to demonstrate the capabilities of the non-pipelined processor on the DE1 SoC FPGA (Field Programmable Gate Array) board. These programs showcase the functionality and performance of the designed processor. The results are shown in Fig.6 and Fig.7.

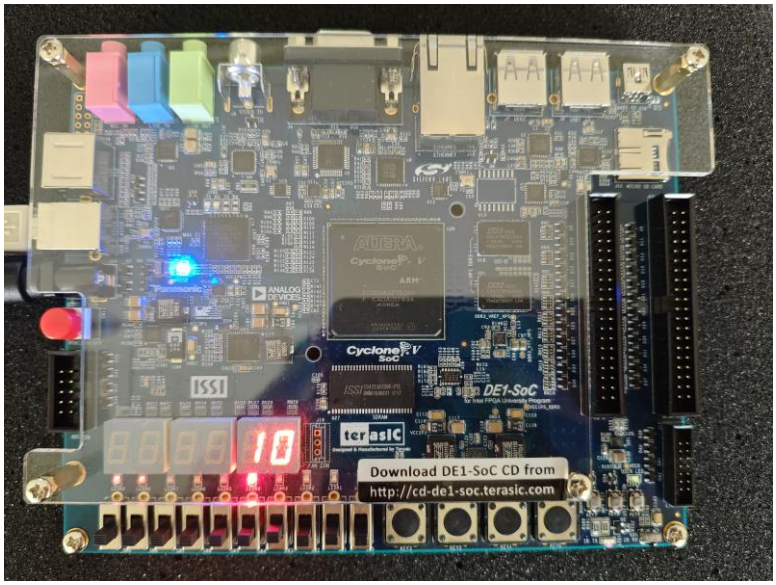
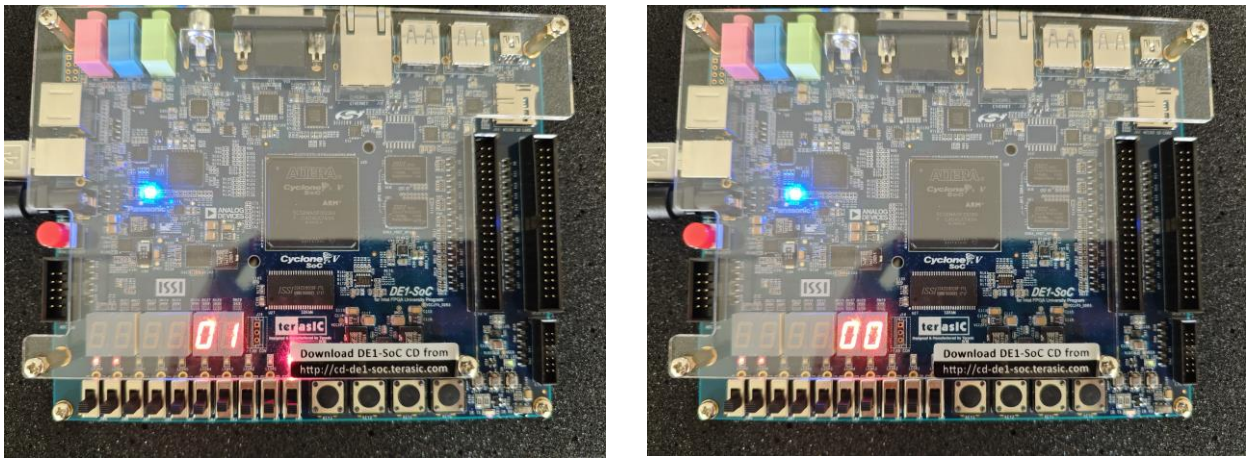


Fig. 6 *'Binary Counter' program*



(a) **(b)**

Fig. 7 *'Blinking LED' program (a) Turn on LED; (b) Turn off LED*

4. Conclusion

An 8-bit non-pipelined RISC processor that can support 16 types of instructions has been designed and implemented based Harvard architecture using Intel Quartus Prime software. The design is simulated with ModelSim-Altera simulator using five types of sample programs and its performance based on the sample programs are summarized in table. Despite the design has some limitations, this study still provides a valuable learning material for 'Computer Architecture' and 'Microprocessor and Microcontroller' subject in UTHM, which allows students to have experience to perform various instruction execution, leading to gain better and deeper understanding of computer architecture, especially non-pipelining concept.

For future work, this design can be enhanced by modifying it to a pipelined processor with hazard unit, forward unit, and branch predictor, to prevent hazards and for a better performance and faster instruction execution. Moreover, optimizing the designed processor by increasing the number of instructions, expanding the bit width of the instruction set, and modifying it to a RISC-V processor can provide greater functionality and enable more complex operations, thereby making it more suitable for educational purpose.

Acknowledgement

The author would like to thank Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, for providing the facility of DE1 SoC FPGA board.

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:*** *Lau Kah Chun;* ***analysis and interpretation of results:*** *Lau Kah Chun;* ***supervision:*** *Chessda Uttraphan, Munirah Ab. Rahman;* ***draft manuscript preparation:*** *Lau Kah Chun. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Wilmshurst, T. (2010). Tiny computers, hidden control. Elsevier EBooks, 3–23. https://doi.org/10.1016/b978-1-85617-750-4.10002-2
[2] Ibrahim, D. (2019). The ARM Microcontrollers. Elsevier EBooks, 25–41. https://doi.org/10.1016/b978-0-08-102969-5.00003-3
[3] Lars Wanhammar. (1999). DSP Architectures. Elsevier EBooks, 357–385. https://doi.org/10.1016/b978-012734530-7/50008-8
[4] Pipelining vs Non-Pipelining. (2020, April 30). GeeksforGeeks. https://www.geeksforgeeks.org/pipelining-vs-non-pipelining/
[5] Vishwakarma, A. (2021, August 24). PIPELINING — AN EASY WAY. Geek Culture. https://medium.com/geekculture/pipelining-an-easy-way-583ca48ab3d0
[6] Harris, S. L., & David Money Harris. (2016). Digital design and computer architecture. Elsevier, Cop.
[7] Bhagya, V. (2021, March 5). Have you heard about Von Neumann architecture, Harvard Architecture and Non Von Neumann…. Medium. https://vimeshibhagya.medium.com/have-you-heard-about-von-neumann-architecture-harvard-architecture-and-non-von-neumann-8b8ce9da720a
[8] Hernandez Zavala, A., Camacho Nieto, O., Huerta Ruelas, J. A., & Carvallo Dominguez, A. R. (2015). Design of a General Purpose 8-bit RISC Processor for Computer Architecture Learning. Computación Y Sistemas, 19(2). https://doi.org/10.13053/cys-19-2-1941
[9] Becvar, M., Alois Pluhacek, & Jiri Danecek. (2003). DOP: A CPU FOR TEACHING BASICS OF COMPUTER ARCHITECTURE. https://doi.org/10.1145/1275521.1275527
[10] Raed Alqadi and Luai Malhis, "An Educational Processor: A Design Approach," Mağallaẗ ğāmi'aẗ al-nağāḥ al-abḥāṭ. A, Al-'ulūm al-tabī'iyyaẗ/Mağallaẗ ğāmi'aẗ al-Nağāḥ li-l-abḥāṭ. Al-ʿulūm al-tabī'iyyaẗ, vol. 20, no. 1, pp. 1–40, Mar. 2006, doi: https://doi.org/10.35552/anujr.a.20.1.600.
[11] Banna, H., Nahin Ul Sadad, & Islam, N. (2023). FPGA Implementation of Educational RISC- V Processor Suitable for Embedded Applications. https://doi.org/10.1109/ecce57851.2023.10101508