Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Design of Low Power Control Unit for RISC-V Processor Core

Chan Jien Sern[1], Chong Li Ting[1], Warsuzarina Mat Jubadi[1,2,*], Chessda Uttraphan Eh Kan[1,2], Lai Jun Boon[3]

[1] Department of Electronic Engineering, Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Batu Pahat, Johor, Malaysia
[2] VLSI Design and System Technology (VEST) Group, Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Batu Pahat, Johor, Malaysia
[3] STATS ChipPAC Pte Ltd (SCS), 5 Yishun Street 23, Singapore 768442, Singapore

ABSTRACT

This research work focuses on the development of a low-power decode logic for a RISC-V processor core with specifications. The goal is to create a controller that performs all six groups of instruction formats outlined in the RV32I Base Integer Instruction Set. The control unit is designed to decode a total of 13 instruction sets, allowing for a comprehensive range of operations. A single instruction pipeline approach is implemented in the design to optimize performance. The synthesis of the design is carried out using the 32 nm standard library, resulting in a maximum operating frequency of 666.67 MHz. To further enhance power efficiency, clock gating techniques are employed, leading to a reduction in power consumption by 18.72 % from 112.15 µW to 91.45 µW. Additionally, the layout of the design is optimized, resulting in an area of 354.74 mm^2. The successful development of this low-power decode logic demonstrates its potential for integration into larger RISC-V processor cores. Future enhancements can include expanding the instruction decoding capability to encompass the full range of 47 instructions in the RV32I Base Integer Instruction Set, as well as exploring additional pipeline stages to further improve performance. The results achieved in this project contribute to the ongoing pursuit of power-efficient and high-performance processor designs based on the RISC-V architecture.

Keywords:
RISC-V; Control unit; Low-power; Clock gating

1. Introduction

In the last few decades, the focus on the semiconductor industry driven by the principles of Moore's Law [1] has revolved around creating smaller, faster and cheaper products. In line with this trend [2], power efficiency has emerged as a critical factor in the industry. In the year 2010, at the University of California Berkley, a new set of reduced instruction set computers called Reduced Instruction Set Computer-Five (RISC-V) was introduced by Patterson *et al.,* [3,4]. Since then, it has garnered much attention internationally. The thing about RISC-V is it is a unique, open-source instruction set architecture (ISA) that is governed by a collaborative community. The community is

* *Corresponding author.*
E-mail address: suzarina@uthm.edu.my

https://doi.org/10.37934/araset.60.2.265281

later known as the RISC-V Foundation in the year 2015 [4]. In contrast, the established microprocessor architectures such as x86 and ARM are proprietary to the manufacturers or inventors [5]. The dominance of large corporations in the technology market has limited the opportunities for small developers and manufacturers to actively participate in hardware design and development [6]. Licensing proprietary of the two dominant ISAs [7]; the x86 and ARM often require hefty royalties. Small developers and manufacturers have little to no say in designing and building the hardware. However, the emergence of RISC-V has brought a potential to even the playing field. The open-source nature allows companies of all sizes to leverage its benefits and contribute to advancing microprocessors [3].

Despite the progress made in various electronic devices, the issue of power consumption remains a persistent challenge. In most cases, achieving the desired performance comes at the expense of increased power consumption. According to a group of researchers [8], this limitation poses a significant inconvenience to users who rely on devices with extended battery longevity such as mobile phones and other smaller Internet-of-Things (IoT) devices. Incorporating an advanced power optimization technique is crucial to achieve power efficiency in the design of a control unit. In a survey by Mittal [9], power consumption in control units can be categorized into dynamic power consumption, which arises from active switching and static power consumption, which results from leakage current. The proposed approach to optimize power consumption involves utilizing the clock gating technique. This technique presents a promising opportunity to reduce power by selectively disabling the clock signal to inactive or idle components within the circuit [10]. By employing clock gating, significant power savings can be achieved, enhancing the overall power efficiency of the control unit design.

This research work embarks on objectives to design, synthesize and create the layout of a control unit for the RISC-V processor core. Then the functionality of the proposed design is simulated using ModelSim-Intel. The power consumption of the design is optimized with the Design Compiler. The work presented in this paper holds significant importance in the field of computer architectures and processor design for small devices or IoTs alike. The freedom and flexibility of RISC-V encourage innovation and customization contributing to a more even playing field for designers.

1.1 Microprocessor Design

A microprocessor can be defined as the controlling unit of a micro-computer compacted into a small chip [11]. Its primary function involves performing Arithmetic Logical Unit (ALU) operations and facilitating interactions with other peripherals connected. The microprocessor can be classified into two major units [11,12]; the Datapath Unit (DU) and the Control Unit (CU) [11,12]. The Datapath Unit (DU) is responsible for carrying out hardware-based operations. It encompasses the necessary components, such as the ALU, to execute mathematical calculations and logical operations. Additionally, it interacts with various registers and memory to process and store data. In essence, if one were to compare a microprocessor to the body of a human, the DU serves as the microprocessor's "organs" by performing computational tasks. Contrarily, the Control Unit functions as the microprocessor's command centre. It directs the operations of the DU and facilitates data movements between different components [11,12]. The CU issues commands, specifies which data should be read or manipulated and orchestrates the sequencing and timing of instructions. Comparatively, the CU can be linked to the "brain" of the microprocessor, as it governs and controls the overall activities.

While there is a clear standard describing the microprocessor, there are many variables and characteristics to be noted before categorizing a microprocessor. According to a few researchers,

[13,14] the most notable classification revolves around the type of architecture employed, such as Reduced Instruction Set Computer (RISC) or Complex Instruction Set Computer (CISC). The architecture distinction greatly influences the microprocessor's behaviour and capabilities, impacting factors like the number of bits it can process, the instruction set it supports and its system clock speed.

1.2 Reduced Instruction Set Computer

The idea behind the invention of RISC in the 1980s is to simplify the complexity of the microprocessor in executing the commands when compared to CISC [15]. Despite its name, the Reduced Instruction Set Computer - V does not fall into the same category as traditional RISC architectures. RISC-V is an open-source ISA that is known for its simplicity, modularity and flexibility [16]. The RISC-V was created to be open-source and free to use by everyone. To put it in layman's terms, RISC describes the category of CPUs while RISC-V is a language used to communicate with the microprocessor.

The integer register state in RISC-V architecture is a simple state that lacks sophisticated instructions for accessing data memory. Thus, the instruction encoded is highly consistent. RISC-V applies a standard naming convention for the ISA which is RV [{32,64,128}] [abc … … xyz] [17].

The RV is an indication of the RISC-V while the [{32,64,128}] indicates the length of the integer register which is either 32-bit, 64-bit, or 128-bit. The [abc … … xyz] is the indication of the set of extensions supported. According to the RISC-V manual [17], the RISC-V ISA consists of four fundamental base integer instruction sets at the moment: RV32I, RV32E, RV64I and RV128I. In this study, the RV32I is used as the base model of this study's design. The RV32I has a total of 47 sets of instructions in 6 formats: R-type, I-type, S-type, U-type B-type and J-type.

1.3 Clock Gating

Power optimization refers to the techniques and strategies employed to reduce power consumption in electronic devices and systems [18]. Clock gating is a technique [19-21] used to reduce power consumption in electronic devices by selectively disabling clock signals to unused or idle components or portions of a circuit according to research done by Pandey *et al*., [18]. It helps to eliminate unnecessary clock cycles and reduces dynamic power consumption. Clock gating operates by inserting gating logic in the clock path of a circuit, allowing the clock signal to be enabled or disabled based on specific conditions. In research by Attaoui *et al.,* [22] when a component or portion of the circuit is not actively processing data or performing computations, the clock signal to that component is gated off, effectively stopping the clock pulses and halting unnecessary power consumption. By selectively gating the clock, power optimization with clock gating offers several benefits such as reducing dynamic power consumption and minimizing leakage power of inactive components [19]. It improves energy efficiency, prolongs battery life and lowers energy costs.

1.4 RISC-V Development and Current Trend

Recent research on RISC-V has shown a growing interest in integrating RISC-V into high-performance computing (HPC) systems due to its flexibility and ability to be customized for specific tasks. Studies have been focusing on developing RISC-V-based processors that can handle parallel processing efficiently for scientific computations [23,24].

Embedded and IoT devices, for example, have extremely strict requirements on the area and power consumption of the processor because of the limitation on its working environment [25,26]. To reduce the overhead of the embedded processor as much as possible, a configurable 32-bit in-order RISC-V processor core based on the 16-bit data path and units, named RV16 has been designed and implemented [26]. It was found that RV16 consumes fewer hardware resources and less power consumption as compared to a traditional 32-bit RISC-V processor with similar features. For example, the maximum performance of RV16 running Dhrystone benchmarks is 0.92 DMIPS/MHz, reaching 75% of traditional 32-bit processors.

As RISC-V becomes more widely adopted, there is an increased focus on enhancing its security features [27-29]. An enhanced 3D RECTANGLE algorithm is proposed to improve confusion and diffusion properties through a new 3D array block rotation method for 4x4 plaintext, based on a 128-bit key and 16 rounds [28]. This new lightweight algorithm named enhanced 3D RECTANGLE [28], is designed to deliver robust security for IoT applications and is optimized for cell phones with minimal memory usage, low power consumption and efficient performance. Yang, Sen *et al.,* [30] designed a low-cost RISC-V processor for IoT applications with an integrated hybrid encryption accelerator to efficiently secure encryption and decryption of data transmitted between IoT devices. The hybrid encryption accelerator used the SM3 and SM4 as hash and symmetric encryption algorithms to achieve a balance between encryption security, high speed and key-management convenience. The proposed processor is compared with Hummingbird E203 and the XuanTie E902 with a total resource utilization rate reduced by 39.1~66.2% on the FPGA platform.

RISC-V architectures are also increasingly being used to address edge AI workloads. The flexibility of the RISC-V ISA and the inclusion of vector extensions make it suitable for specific AI and machine learning workloads, such as computer vision and natural language processing [31,32]. Studies have highlighted the benefits of using RISC-V with vector extensions and accelerators tailored for AI operations like matrix multiplications and deep learning tasks [32,33]. Recent network architecture search (NAS) has been widely used to simplify deep learning neural networks. However, this often produces multi-precision networks. Many multi-precision accelerators have also been created to facilitate manual computation of multi-precision networks. A Resistive Random Access Memory (RRAM) technology has been studied as a possible future candidate for these accelerators since crossbar implementations allow for the evaluation of matrix-vector multiplications [34].

A software-hardware interface is thus required to automatically map multi-precision networks to multi-precision accelerators. For example, EXTREM-EDGE is a hardware/software co-design technique developed to add bespoke extensions to the open-source RISC-V to create a scalable and adaptable ML processor architecture [35]. EXTREM-EDGE enhances the RISC-V processor with hardware AI functional units (AFU) and ISA extensions that directly target these AFUs. An agile hardware/software co-design for RISC-V-based multi-precision deep learning microprocessor is developed by He *et al.,* [36]. This custom RISC-V instruction was designed with a framework to automatically compile multi-precision CNN networks onto multi-precision CNN accelerators, demonstrated on FPGA. The researcher shows that with NAS-optimized multi-precision CNN models (LeNet, VGG16, ResNet, MobileNet), the RISC-V core with multi-precision accelerators can reach the highest throughput in 2,4,8-bit precisions respectively on a Xilinx ZCU102 FPGA [35]. All these trends suggest that RISC-V will continue to play a critical role in the evolution of AI, edge computing and server platforms, with ongoing research and industry support driving its adoption forward.

2. Methodology

2.1 Design Methodology

In this work, a control unit for the RISC-V processor is developed following the datapath unit. The design of the control unit is divided into several parts as shown in Figure 1:

i. Part 1: literature review of the microprocessor, RISC-V, the challenges and power optimization;
ii. Part 2: Verilog coding and functional verification
iii. Part 3: Synthesis of the design
iv. Part 4: The power optimization via clock gating
v. Part 5: Layout of the design.

Three (3) design tools are used in this project: Intel Quartus Prime 20.1 Lite Edition, Synopsys Design Compiler and Synopsys Integrated Chip Compiler.

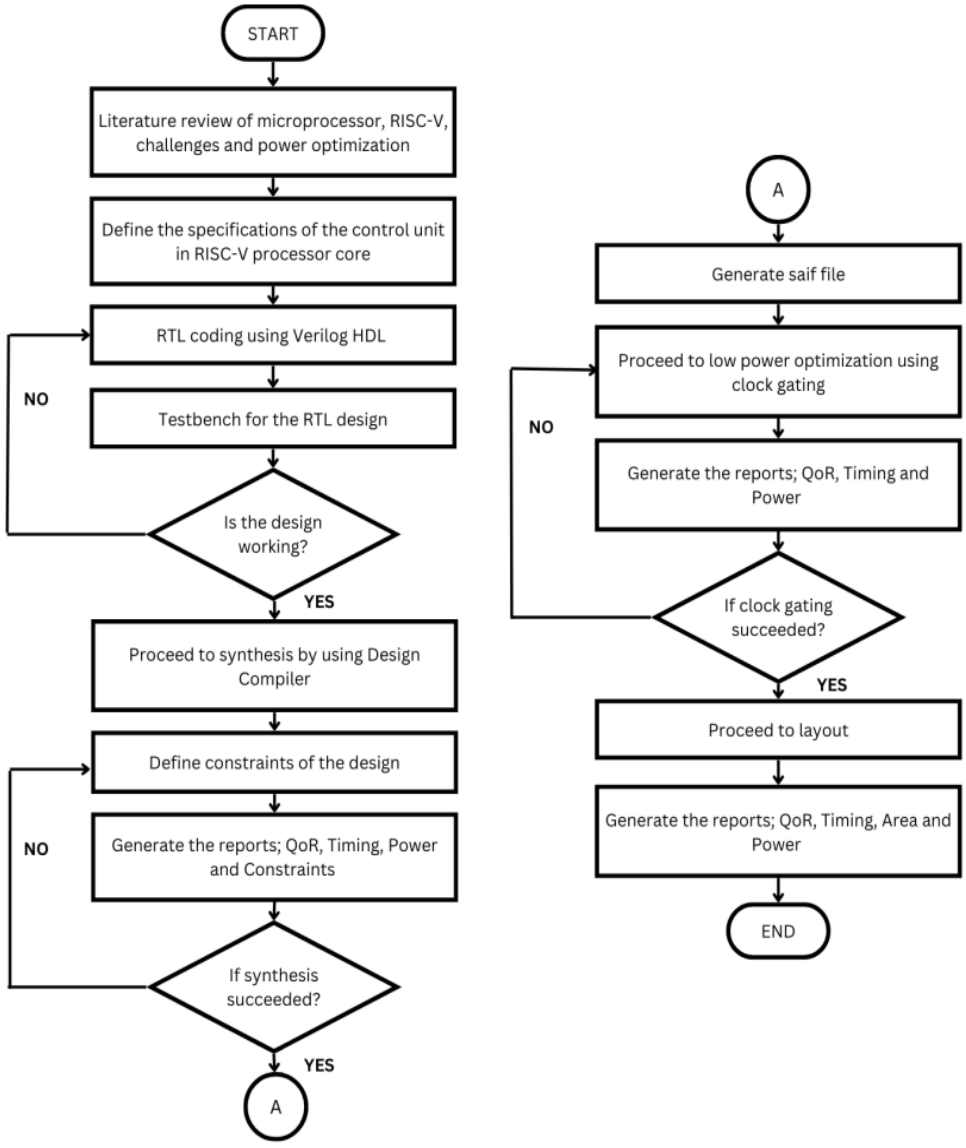

Fig. 1. Flowchart of the design methodology

2.1.1 Intel Quartus Prime 20.1 lite edition

Intel Quartus Prime is a programmable logic device design software. The software is used for analysis and synthesis of HDL designs and has many other features. Prior to this, the software is known as Altera Quartus Prime. The Quartus Prime Lite Edition is a free version of the software with limited features as compared to the paid version. In this study, this software will be used for scripting purposes which is to develop the Verilog HDL coding of the control unit and also the test-bench. In order to simulate the results to have a better visual for debugging purposes, an extension of the Quartus Prime will be used; ModelSim-Intel. The ModelSim is a multi-language environment for the simulation of hardware description languages (HDL). It allows the functionality of the design to be simulated visually, usually in waveform. This allows easier verification of the design or for debugging purposes.

2.1.2 Synopsys design compiler

Synopsis Inc. is an Electronic Design Automation (EDA) tool provider. The company's portfolio comprises a wide range of IC design solutions in various fields. The Design Compiler is a synthesis solution tool provided by the company. In this study, the Synopsys Design Compiler is used. Design Compiler is a compact synthesis tool that concurrently optimize timing, area and power as well as generating reports needed. This tool will also be used for power optimization.

2.1.3 Synopsys integrated chip compiler (ICC)

Synopsys ICC (Integrated Chip Compiler) is a tool used for the physical design of integrated circuits. It handles tasks like floor planning, placement, clock tree synthesis and routing. ICC optimizes factors such as area, power, timing and performs checks for design rule compliance. It enables designers to achieve high-performance, low-power designs through its advanced algorithms and optimization techniques.

2.2 Design of Control Unit

The design contains a total of 4 input pin lists and 13 output pin lists as in Figure 2. For the input pin list, the operation code (Opcode), function 3 and function 7 are the variables that determine the type of instruction to perform. The BrEq input is used to check if the condition allows for branching. Depending on the combinations of inputs, the output will vary accordingly.

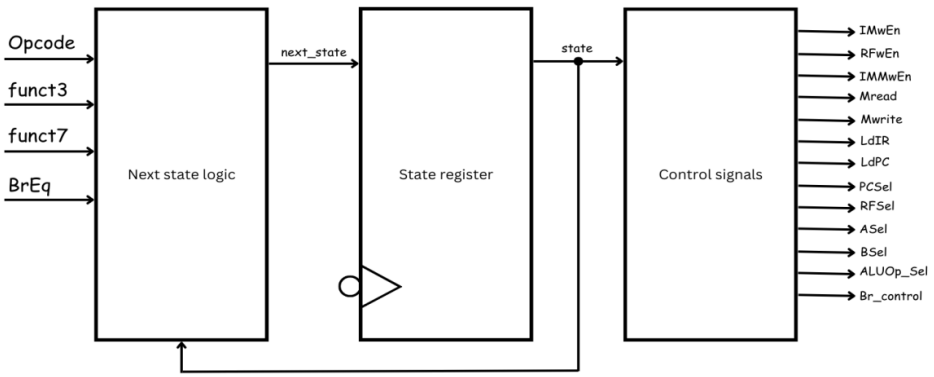

Fig. 2. Block diagram of the control unit

Figure 3 shows the overall block diagram of RISC-V Processor Core. Each of the inputs and outputs represents a signal that is used to control the blocks in the RISC-V processor core.

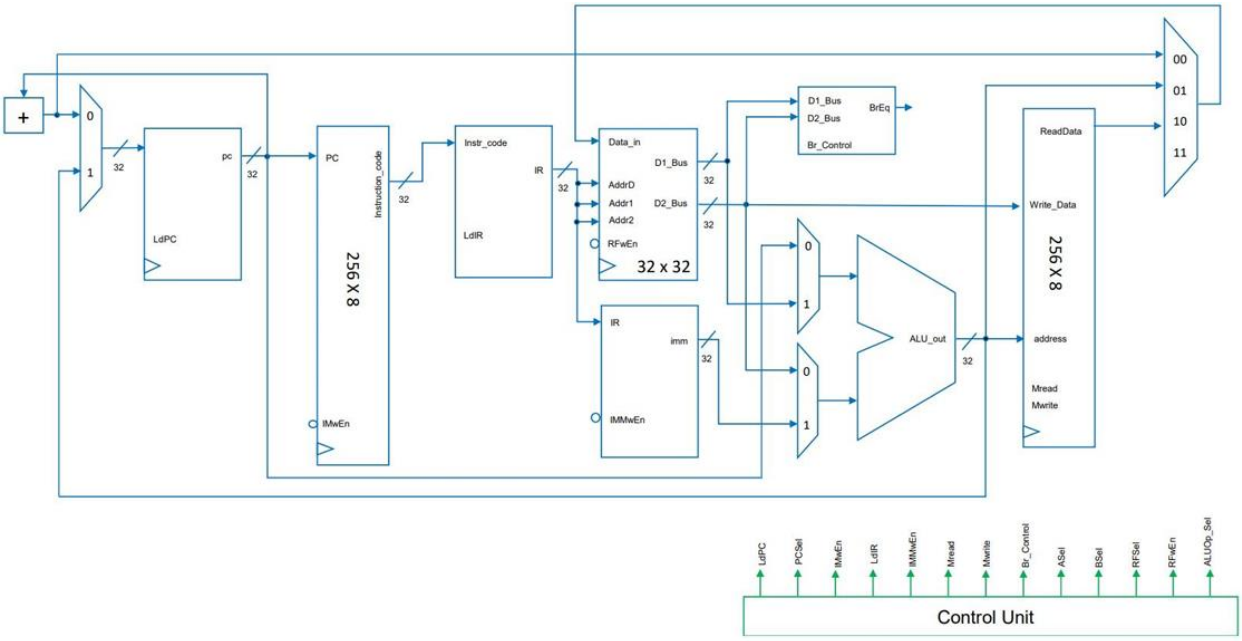

Fig. 3. Block diagram of RISC-V processor core

The naming of the inputs and outputs are as defined in Table 1.

Table 1
The inputs and outputs of control unit

Parameter Category	Parameter Symbol	Parameter Function
Input	Opcode	Operation code
Input	funct3	Unsigned immediate for 3-bit function code
Input	funct7	Unsigned immediate for 7-bit function code
Input	BrEq	Branch if equal
Output	IMwEn	Instruction memory enable
Output	RFwEn	Register file enable
Output	IMMwEn	Immediate generator enables
Output	Mread	Memory reading enable
Output	Mwrite	Memory writing enable
Output	LdIR	Instruction memory enable
Output	LdPC	Program counter enable
Output	PCSel	Program counter selection
Output	RFSel	Register file multiplexer enable
Output	ASel	Multiplexer A enable
Output	BSel	Multiplexer B enable
Output	ALUOp_Sel	ALU control enable
Output	Br_control	Branch control enable

2.3 The RV32I

The control unit that decodes the high-level language to machine code has a certain distinct format for each of the group instructions. Based on the RV32I of the RISC-V ISA, there are a total of 47 sets of instructions that are classified into 6 groups. Out of which, in this project, only 13

instructions from all the 6 groups will be implemented. From Table 2, the format of each group of RISC-V instructions is portrayed.

Table 2
The 6 groups instruction format of RISC-V [16]

Group	31	30 - 25	24 - 21	20	19 - 15	14 - 12	11 - 8	7	6 - 0
R	Funct7		Register source 2		Register source 1	funct3	Register destination		Opcode
I	Imm[11:0]				Register source 1	funct3	Register destination		Opcode
S	imm[11:5]		Register source 2		Register source 1	funct3	imm[4:0]		Opcode
B	imm[21]	imm[10:5]	Register source 2		Register source 1	funct3	imm[4:1]	imm[17]	Opcode
U	imm[31:12]						Register destination		Opcode
J	imm[20]	imm[10:1]		imm[17]	imm[19:12]		Register destination		Opcode

2.4 List of Instructions the Control Unit Decodes

The control unit in this work is designed to decode a specific subset of instructions from the RV32I instruction set. The selected instructions are based on the function and specification of the data path unit. The 13 instructions that will be decoded are as listed; R- type - add, sub and, or; I-type - addi (add immediate), slli (shift left logical immediate), srli (shift right logical immediate), lw (load word); S-type - sw (store word); U-type - lui (load upper immediate), auipc (add upper immediate with pc); J-type - jal (unconditional jump); B-type - beq (conditional jump). The Table 3 shows the instruction format of each of the 13 instructions.

Table 3
The instruction format of the 13 instructions

Group	31 30 - 25	24 - 21	20	19 - 15	14 - 12	11 - 8	7	6 - 0
ADD	00000000	rs2		rs1	000	rd		0110011
SUB	01000000	rs2		rs1	000	rd		0110011
AND	00000000	rs2		rs1	111	rd		0110011
OR	00000000	rs2		rs1	110	rd		0110011
ADDI	imm[11:0]			rs1	000	rd		0010011
SLLI	000000X	shamt		rs1	001	rd		0010011
SRLI	000000X	shamt		rs1	101	rd		0010011
LW	offset[11:0]			rs1	010	rd		0000011
SW	offset[11:5]	rs2		rs1	010	offset[11:0]		0100011
LUI	imm[31:12]					rd		0110111
AUIPC	imm[31:12]					rd		0010111
JAL	imm[20|10:1|11|19:12]					rd		1101111
BEQ	offset[12|10:5]	rs2		rs1	000	offset[4:1|11]		1100011

2.5 The Control Sequence

The working principle of the control unit is demonstrated in an algorithmic state machine (ASM) chart as shown in Figure 4.

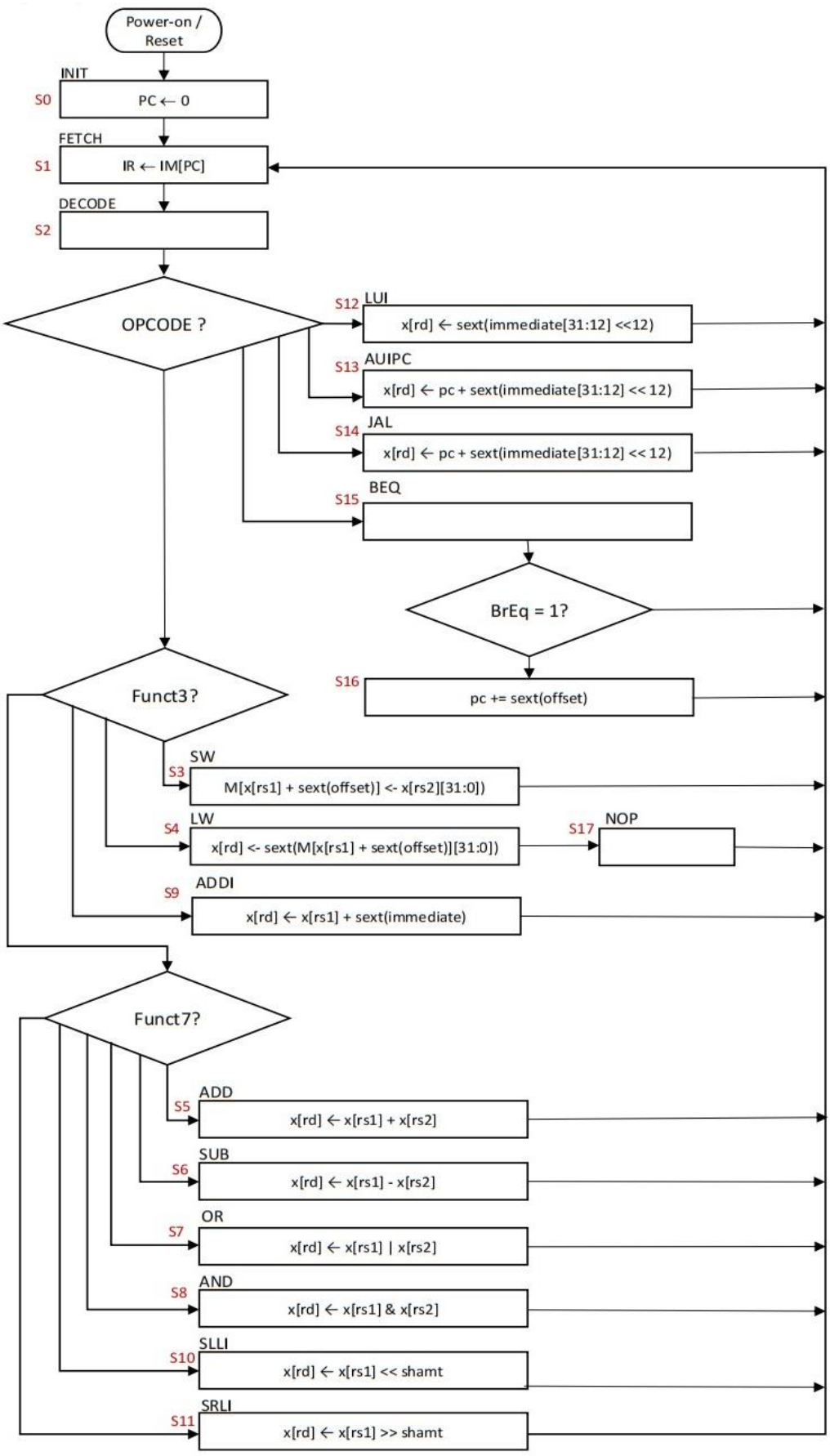

Fig. 4. The ASM chart

The program has a total of 17 states to demonstrate the functionality of each of the control sequences as in in Table 4. The Combined Value (CV) is the combination value of the 13 outputs in hexadecimal value in a total of 5 bytes. Each CV is unique following the instructions. Each instruction is programmed to start from state 1 (S1), the fetch stage and proceed to state 2 (S2), the decode stage before the execution stage. After completing the execution of one instruction, the program will restart at S1 and repeat the cycle for the execution of the next instruction.

Table 4
The instruction format of the 13 instructions

		RFwEn	IMMwEN	IMwEn	Mread	Mwrite	LdIR	LdPC	PCsel	RFsel	Asel	Bsel	ALUOp_Sel	Br_control	CV
50	PC <- 0	1	1	1	0	0	0	0	0	01	1	0	1000	0	20'h1C0D0
S1	IR <- IM[PC]	1	1	0	0	0	1	0	0	01	1	0	1000	0	20'h188D0
S2		x	x	x	x	x	x	x	x	xx	X	X	Xxx	X	20'hxxxx
S3	SW	1	0	0	0	1	1	1	0	10	1	1	0000	0	20'11D60
S4	LW	0	0	0	1	0	0	0	0	10	1	1	0000	0	20'h02160
S5	ADD	0	1	0	0	0	1	1	0	01	1	0	0000	0	20'h08CC0
S6	SUB	0	1	0	0	0	1	1	0	01	1	0	0001	0	20'h08CC2
S7	OR	0	1	0	0	0	1	1	0	01	1	0	0010	0	20'h08CC4
S8	AND	0	1	0	0	0	1	1	0	01	1	0	0011	0	20'h08CC6
S9	ADDI	0	0	0	0	0	1	1	0	01	1	1	0000	0	20'h00CE0
S10	SLLI	0	0	0	0	0	1	1	0	01	1	1	0100	0	20'hOOCE8
S11	SRLI	0	0	0	0	0	1	1	0	01	1	1	0101	0	20'h00CEA
S12	LUI	0	0	0	0	0	1	1	0	01	0	1	0110	0	20'h00CAC
S13	AUIPC	0	0	0	0	0	1	1	0	01	0	1	0000	0	2o'h01CA0
S14	JAL	0	0	0	1	0	1	1	1	00	0	1	0000	0	20'h02E20
S15	BEQ [check]	0	0	0	0	0	0	0	0	01	1	0	0000	1	20'h00C1
S16	BEQ [false]	1	1	1	0	0	1	1	0	01	1	0	0000	0	20'h1CCC0
	BEQ [true]	0	0	0	1	0	1	1	1	00	0	1	0000	1	20'h02E21
S17	NOP (LW)	0	0	0	1	0	1	1	0	10	1	1	0000	0	20'h02D60

2.6 The Synthesis of Control Unit

During the synthesis process, the Verilog file is read and processed. This file contains the design description in a hardware description language (HDL). Additionally, a SAIF (Signal Activity Interchange Format) file is generated from VCS (Verilog Compiler Simulator) specifically for implementing clock gating techniques. The SAIF file, which contains information about signal activities and timing, is then read and utilized in the synthesis flow. Finally, various reports are generated, providing valuable insights into the synthesis results, such as timing analysis, Quality of Results (QoR) and power estimation.

2.7 Clock Gating Technique

Clock gating is a technique used to reduce power consumption in electronic devices by selectively disabling clock signals to unused or idle components or portions of a circuit [10]. Clock gating helps to eliminate unnecessary clock cycles and reduces dynamic power consumption. Clock gating operates by inserting gating logic in the clock path of a circuit, allowing the clock signal to be enabled or disabled based on specific conditions. In this project, by selectively gating the clock, power optimization can be achieved by reducing the dynamic power consumption and minimizing the leakage power of inactive components.

2.8 The Layout of Control Unit

During the layout phase, several steps are performed in Synopsys ICC to ensure the correctness and reliability of the layout. First, the Synopsys Design Constraints (SDC) files generated from the Design Compiler are read to obtain the necessary design constraints for the layout. These constraints define various aspects such as timing, power and area requirements. Once the constraints are in place, the layout compilation process takes place, where the physical representation of the circuit is created. After the layout is generated, it undergoes a series of checks, including Design Rule Checking (DRC) and Layout versus Schematic (LVS) checks. These checks verify the compliance of the layout with the design rules and the consistency between the layout and the schematic representation.

3. Results

3.1 Functional Verification

The testbench simulation of the control unit is performed in ModelSim. For the verification of the design, one instruction from six groups of the RV32I ISA was analysed and presented in this section. The results of the simulations are shown from Figure 5 to Figure 10. The simulation output for each instruction is compared with the instruction set in Table 4 to verify that the control sequence of each instruction is correct.

Figure 5 shows the simulation result at 720 ns for instruction *ADD x1, x2, x3*. The CV output is 08CC0 and the corresponding state is 5. The Control Unit accurately interprets the inputs of Opcode (0110011), function 3 (000) and function 7 (0000000), producing appropriate control signals for the execution of the AND operation. This validation demonstrates the reliable functionality of the Control Unit in executing the AND instruction.

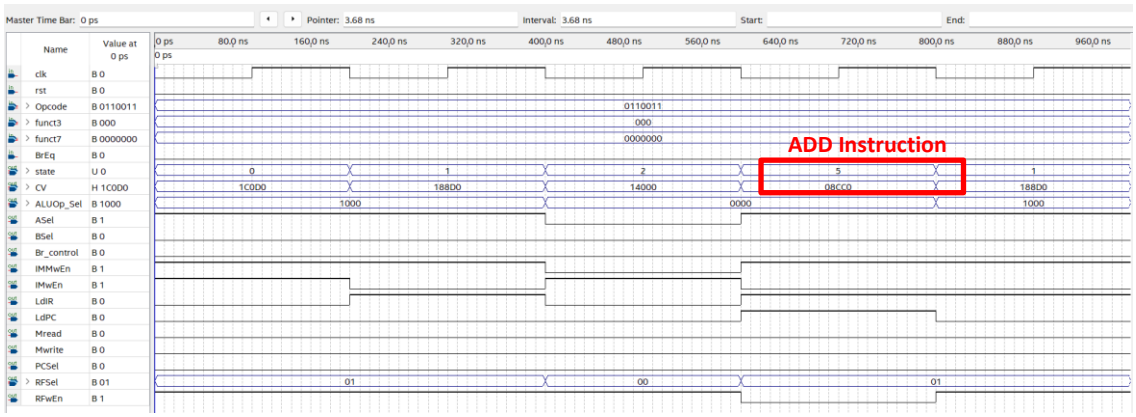

Fig. 5. Testbench simulation of R-type, ADD

In the output simulation of Load Word (LW) instruction shown in Figure 6, two observations can be made at different time points. The output of the CV is recorded as 02160 and the corresponding state is 4 as shown in Figure 6. At 880 ns, the output is 02d60 and the state is 17. The inputs for this instruction are Opcode (0000011) and function 3 (010). It is important to note that the LW instruction requires an additional cycle to store the data in the Instruction Memory (IM) and hence state 17 represents an NOP (No Operation) instruction that provides the extra time needed for storing the instruction. This behaviour is consistent with the single pipeline design of the CU.

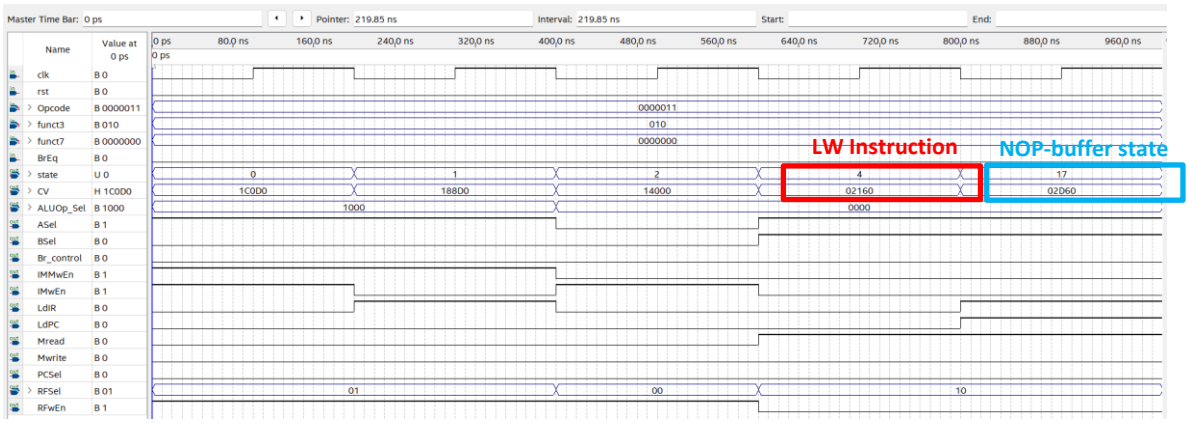

Fig. 6. Testbench simulation of I-type, LW to execute LW x2, #4(x1) and NOP(LW)

Figure 7 depicts the timing diagram of Store Word (SW) instruction where it stores a 32-bit value from the LSB of register source 2 (rs2). The output of the CV is recorded as 11D60 (Hex value is with the corresponding state determined as 3.

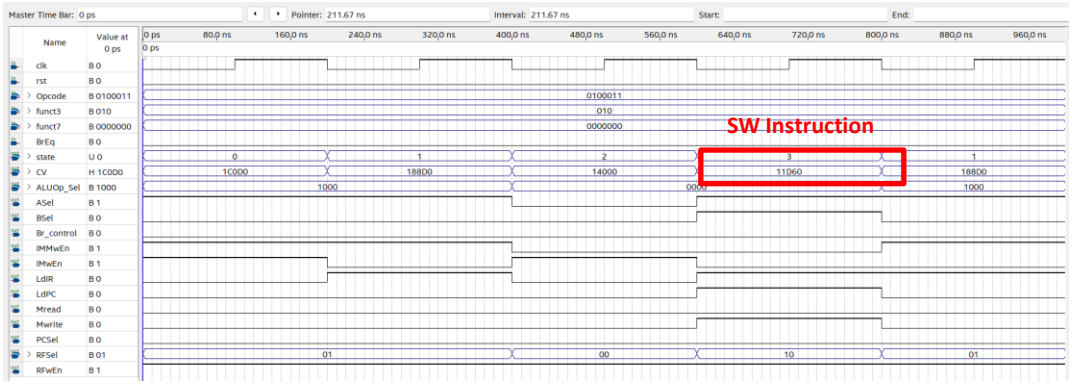

Fig. 7. Testbench simulation of S-type, SW for instruction *SW x3,#20(x2)*

Comparing these values with the control sequence of SW provided in Table 5 (refer to state S3), it is validated all the control sequence is correct. This validates the accuracy and proper functioning of the Control Unit in executing the SW instruction according to the given inputs of Opcode (0100011) and function 3(010).

Table 5
The control sequence of SW operation

		RFwen	IMMwEN	IMwEn	Mread	Mwrite	LdIR	LdPC	PCsel
S3	SW x3, #20 (x2)	1	0	0	0	1	1	1	0
		RFSel	Asel	Bsel	ALUOP-Sel	Br_control			
		10	1	1	0000	0			

Figure 8 shows the result when instruction *JAL x18, #4* is performed. This allows a jump to address and place the return address in the register destination (rd). The CV output is documented as 02E20, signifying the corresponding state as 14. These output values are compared with the instruction provided in Table 4 to confirm that the control sequence for the JAL instruction is accurate. This validation affirms the precise and effective operation of the Control Unit in executing the JAL instruction based on the provided Opcode (1101111) input.

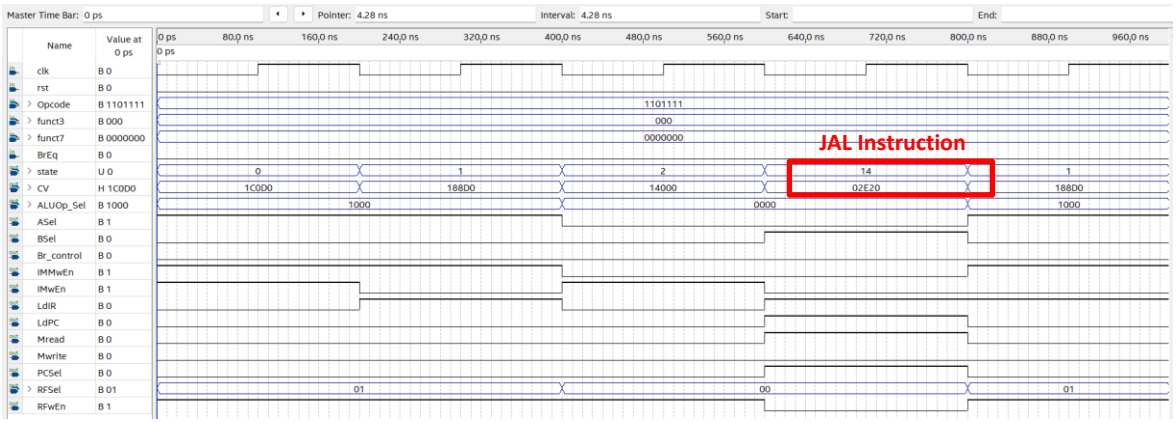

Fig. 8. Testbench simulation of J-type, JAL for instruction JAL x18, #4

Figure 9 depicts the simulation result of BEQ instruction where the registers are checked for equality. At 320 ns, the CV is 000C1 and the state is 15. If the registers are found to be equal, the CU receives a high input signal for BrEq. However, if the registers are found to be not equal, the CU receives a low input signal for BrEq. This validation is labelled as BEQ (true) and BEQ (false) in Figure 9, respectively.

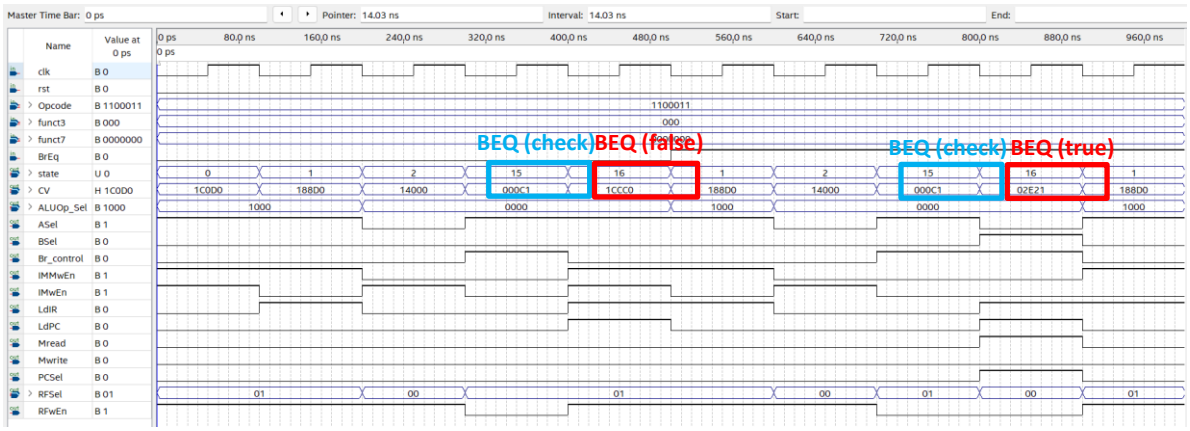

Fig. 9. Testbench simulation of B-type, BEQ

The state 15 (S15) plays a crucial role in the subsequent determination of the S16 stage. At 400 ns, the output of the CV at S16 is recorded as 1CCC0. During this time, the BrEq signal is active low, indicating that the registers are not equal and the condition for branching is not satisfied. On the other hand, when the output of CV at S16 is 02E21, it is indicated that the condition for branching is met. These observations confirm the proper functioning of the Control Unit in executing the BEQ instruction based on the provided inputs of Opcode (1100011) as in Table 6.

Table 6
The control sequence of BEQ operation

		RFwen	IMMwEN	IMwEn	Mread	Mwrite	LdIR	LdPC	PCsel	RFSel	Asel	Bsel	ALUOP-Sel	Br_control	Hew
S15	BEQ x19, x20, #8 [check]	0	0	0	0	0	0	0	0	01	1	0	0000	1	20'hOOOC1
S16	BEQ x19, x20, #8 [false]	1	1	1	0	0	1	1	0	01	1	0	0000	0	20'h1CCC0
	BEQ x19, x20, #8 [true]	0	0	0	1	0	1	1	1	00	0	1	0000	1	20'h02E21

3.2 Synthesis and Power Gating Results

During synthesis, a set of constraints was applied to evaluate the design and ensure it meets various requirements, such as timing constraints and power consumption. The design was evaluated with a timing period of 1.5 nanoseconds successfully with a positive slack of 0.09. This means that the design can operate at a frequency that allows for a maximum operating frequency of 666.67 MHz. Regarding the power consumption, the synthesis was performed twice to obtain the results; first without clock gating and the latter with the clock gating technique as shown in Table 7. The design initially consumed 112.1507 µW. After applying the clock gating technique, the power consumption was reduced to 91.4526 µW. This technique effectively reduces the power consumption by a total of 18.72 %.

Table 7
Power report after synthesis

Power	Before Clock Gating	After Clock Gating
Internal power (µW)	0.0000	25.0191
Switching power (µW)	6.4342	7.3084
Leakage power (pW)	1.0572e+08	5.9125e+07
Total power (µW)	112.1507	91.4526

3.3 Layout Results

After the layout process, the CU occupies a total area of 354.72 mm^2. It consists of 62 ports, 125 nets, 133 cells, 128 combination cells, 5 sequential cells, 26 buffers/inverters and 21 references. Figure 10 shows the layout after the routing process, which determines the precise paths and connections between different components. It is worth noting that the layout of the CU successfully passed the LVS (Layout vs. Schematic) and DRC (Design Rule Check) verification without any violations. This indicates that the layout adheres to the specified design rules and accurately represents the intended circuitry.

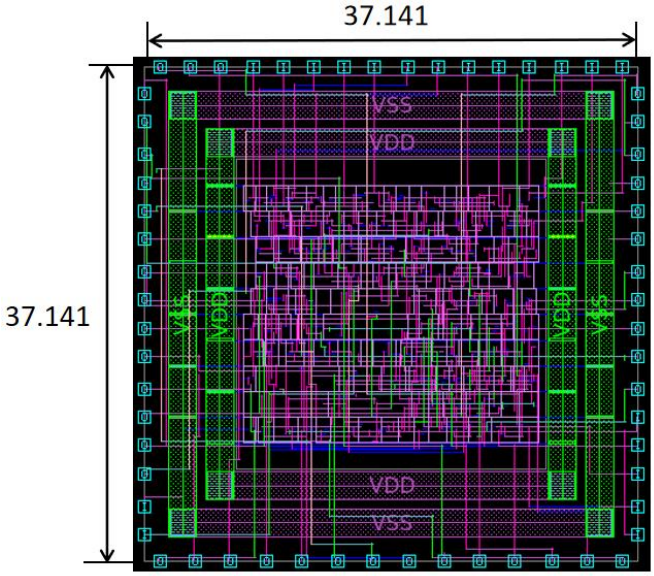

Fig. 10. Layout of the control unit after placement and routing with 37.141 x 37.141 mm^2 area

4. Conclusions

In summary, the objective of designing, synthesizing and creating the layout has been achieved. The functional verification phase involved utilizing ModelSim-Intel to successfully validate all 13 instructions across the 17 stages, ensuring the correctness of the design. Moving on to synthesis, the Control Unit was synthesized using Synopsys Design Compiler (DC) and achieved an impressive maximum operating frequency of 666.67 MHz, highlighting the design's efficiency. The clock gating technique proved particularly effective, which initially consumed 112.1507 µW power and is reduced to 91.4526 µW resulting in an 18.72 % reduction in power usage. Furthermore, the layout of the CU was created using Synopsys IC Compiler yielding a total area of 354.72 mm^2. The successful completion of these stages demonstrates the meticulousness and effectiveness of the design process in meeting functional, timing, power and area requirements. The completion of this project marks

the beginning of further enhancements and integration into a larger design, as the current implementation only focuses on the CU component. To make improvements, the design can be upgraded to support the decoding of additional instructions based on the RV32I integer instruction set, which encompasses a total of 47 instructions. By expanding the instruction decoding capability, the CU can handle a broader range of operations.

Acknowledgment
The authors would like to thank Universiti Tun Hussein Onn Malaysia for the technical and financial support.

References

[1] Xiu, Liming. "Time Moore: Exploiting Moore's Law from the perspective of time." *IEEE Solid-State Circuits Magazine* 11, no. 1 (2019): 39-55. https://doi.org/10.1109/MSSC.2018.2882285
[2] Yang, Fengwei and Sai Gu. "Industry 4.0, a revolution that requires technology and national strategies." *Complex & Intelligent Systems* 7 (2021): 1311-1325. https://doi.org/10.1007/s40747-020-00267-9
[3] Waterman andrew, Yunsup Lee, David A. Patterson and Krste Asanovic. "The RISC-V instruction set manual, volume I: User-level ISA, version 2.0." *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54* (2014): 4. https://doi.org/10.21236/ADA605735
[4] Höller, Roland, Dominic Haselberger, Dominik Ballek, Peter Rössler, Markus Krapfenbauer and Martin Linauer. "Open-source risc-v processor ip cores for fpgas—overview and evaluation." In *2019 8th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1-6. IEEE, 2019. https://doi.org/10.1109/MECO.2019.8760205
[5] Zhang, Yunrui, Zichao Guo, Jian Li, Fan Cai and Jianyang Zhou. "Annikacore: Risc-v architecture processor design and implementation for iot." In *2021 IEEE 15th International Conference on Anti-counterfeiting, Security and Identification (ASID)*, pp. 200-203. IEEE, 2021. https://doi.org/10.1109/ASID52932.2021.9651690
[6] Moseman, A. "Scientists broke a major computer design barrier - and it could change tech as we know it." *Inverse*, (2023). https://www.inverse.com/science/riscv-computer-chip-autonomous-cars
[7] Jolly, Brad. "IoT device battery life: Go slow for fast insights into challenging conditions." In *2021 IEEE international midwest symposium on circuits and systems (MWSCAS)*, pp. 680-683. IEEE, 2021. https://doi.org/10.1109/MWSCAS47672.2021.9531705
[8] Qu, Pei-Yao, Guang-Ming Tang, Xiao-Chun Ye, Dong-Rui Fan, Zhi-Min Zhang and Ning-Hui Sun. "Design of datapath circuits for a bit-parallel 8-bit RSFQ microprocessor." In *2019 IEEE International Superconductive Electronics Conference (ISEC)*, pp. 1-4. IEEE, 2019. https://doi.org/10.1109/ISEC46533.2019.8990911
[9] Mittal, Sparsh. "A survey of techniques for dynamic branch prediction." *Concurrency and Computation: Practice and Experience* 31, no. 1 (2019): e4666. https://doi.org/10.1002/cpe.4666
[10] Nguyen, Phuc-Vinh, Phuoc-Loc Diep and Duc-Hung Le. "A Low-Power ASIC Implementation of Multi-Core OpenSPARC T1 Processor on 90nm CMOS Process." In *2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pp. 95-100. IEEE, 2018. https://doi.org/10.1109/MCSoC2018.2018.00027
[11] John, L. *Computer Organization and Design Risc-v Edition-the Hardware Software Int*. Elsevier Science & Technology, 2017.
[12] Jang, Hyeonguk, Kyuseung Han, Sukho Lee, Jae-Jin Lee, Seung-Yeong Lee, Jae-Hyoung Lee and Woojoo Lee. "Developing a multicore platform utilizing open RISC-V cores." *IEEE Access* 9 (2021): 120010-120023. https://doi.org/10.1109/ACCESS.2021.3108475
[13] Dogan Ibrahim. "Chapter 1 - Microcomputer systems," *Arm-Based Microcontroller Multitasking Projects*, Oxford, UK: Newnes. (2021): 1-12. https://doi.org/10.1016/B978-0-12-821227-1.00001-3
[14] Burrell, Mark and Mark Burrell. "CISC and RISC Architectures: An Overview." *Fundamentals of Computer Architecture* (2004): 299-304. https://doi.org/10.1007/978-1-137-11313-9_19
[15] Mishra, Sadhna K., K. Mishra, Arvind Rajawat and R. P. Singh. "Processor Architecture Design Practices : survey & Issues SADHNA." (2010).
[16] Pinyotrakool, Kan and Boonchuay Supmonchai. "Design of a low power processor for embedded system applications." In *2020 8th International Electrical Engineering Congress (iEECON)*, pp. 1-4. IEEE, 2020. https://doi.org/10.1109/iEECON48109.2020.229544
[17] Waterman andrew, Krste Asanovic and John Hauser. "The RISC-V instruction set manual, volume II: Privileged architecture." *RISC-V Foundation* (2019): 1-4.

[18] Pandey, Bishwajeet, Deepa Singh, Deepak Baghel, Jyotsana Yadav and Manisha Pattanaik. "Clock gated low power memory implementation on virtex-6 FPGA." In *2013 5th International Conference and Computational Intelligence and Communication Networks*, pp. 409-412. IEEE, 2013. https://doi.org/10.1109/CICN.2013.90

[19] Shanmugasundaram, N. "Clock gating techniques: an overview." In *2018 conference on emerging devices and smart systems (ICEDSS)*, pp. 217-221. IEEE, 2018. https://doi.org/10.1109/ICEDSS.2018.8544281

[20] Sahu, Preeti and Santosh Kumar Agrahari. "Comparative Analysis of Different Clock Gating Techniques." In *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 1-6. IEEE, 2020. https://doi.org/10.1109/ICRAIE51050.2020.9358375

[21] Samanth, Rashmi, C. V. S. Chaitanya and G. Subramanya Nayak. "Power reduction of a functional unit using rt-level clock-gating and operand isolation." In *2019 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, pp. 1-4. IEEE, 2019. https://doi.org/10.1109/DISCOVER47552.2019.9008025

[22] Attaoui, Yassine, Mohamed Chentouf, Zine El Abidine Alaoui Ismaili and Aimad El Mourabit. "Clock gating efficiency and impact on power optimization during synthesis flow." In *2021 International Conference on Microelectronics (ICM)*, pp. 13-16. IEEE, 2021. https://doi.org/10.1109/ICM52667.2021.9664896

[23] Suárez, Daniel, Francisco Almeida and Vicente Blanco. "Comprehensive analysis of energy efficiency and performance of ARM and RISC-V SoCs." *The Journal of Supercomputing* (2024): 1-19. https://doi.org/10.21203/rs.3.rs-3405993/v1

[24] Brown, Nick and Maurice Jamieson. "Performance characterisation of the 64-core SG2042 RISC-V CPU for HPC." *arXiv preprint arXiv:2406.12394* (2024).

[25] Torres-Sánchez, Enrique, Jesús Alastruey-Benedé and Enrique Torres-Moreno. "Developing an AI IoT application with open software on a RISC-V SoC." In *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*, pp. 1-6. IEEE, 2020. https://doi.org/10.1109/DCIS51330.2020.9268645

[26] Cheng, Yuan-Hu, Li-Bo Huang, Yi-Jun Cui, Sheng Ma, Yong-Wen Wang and Bing-Cai Sui. "RV16: An Ultra-Low-Cost Embedded RISC-V Processor Core." *Journal of Computer Science and Technology* 37, no. 6 (2022): 1307-1319. https://doi.org/10.1007/s11390-022-0910-x

[27] Haj-Yahya, Jawad, Ming Ming Wong, Vikramkumar Pudi, Shivam Bhasin and Anupam Chattopadhyay. "Lightweight secure-boot architecture for risc-v system-on-chip." In *20th International Symposium on Quality Electronic Design (ISQED)*, pp. 216-223. IEEE, 2019. https://doi.org/10.1109/ISQED.2019.8697657

[28] Kumar, Vinay BY, Suman Deb, Naina Gupta, Shivam Bhasin, Jawad Haj-Yahya, Anupam Chattopadhyay and Avi Mendelson. "Towards designing a secure RISC-V system-on-chip: ITUS." *Journal of Hardware and Systems Security* 4 (2020): 329-342. https://doi.org/10.1007/s41635-020-00108-8

[29] Ali, Tasnuva, Azni Haslizan Ab Halim and Nur Hafiza Zakaria. "3D Lightweight Cryptosystem Design for IoT Applications Based on Composite S-Box." *International Journal of Computational Thinking and Data Science* 3, no. 1 (2024): 40-54. https://doi.org/10.37934/ctds.3.1.4054

[30] Yang, Sen, Lian Shao, Junke Huang and Wanghui Zou. "Design and Implementation of Low-Power IoT RISC-V Processor with Hybrid Encryption Accelerator." *Electronics* 12, no. 20 (2023): 4222. https://doi.org/10.3390/electronics12204222

[31] Himeur, Yassine, Aya Sayed, Abdullah Alsalemi, Faycal Bensaali and Abbes Amira. "Edge AI for Internet of Energy: Challenges and perspectives." *Internet of Things* (2023): 101035. https://doi.org/10.1016/j.iot.2023.101035

[32] Kovačević, Nikola, Đorđe Mišeljić and Aleksa Stojković. "RISC-V vector processor for acceleration of machine learning algorithms." In *2022 30th Telecommunications Forum (TELFOR)*, pp. 1-4. IEEE, 2022. https://doi.org/10.1109/TELFOR56187.2022.9983779

[33] Li, RS., Peng, P., Shao, ZY. *et al.,* "Evaluating RISC-V Vector Instruction Set Architecture Extension with Computer Vision Workloads." *Journal of Computer Science and Technology* 38, (2023): 807–820. https://doi.org/10.1007/s11390-023-1266-6

[34] Fritscher, Markus, Alessandro Veronesi andrea Baroni, Jianan Wen, Thorsten Spätling, Mamathamba Kalishettyhalli Mahadevaiah, Norbert Herfurth *et al.,* "Prototyping Reconfigurable RRAM-Based AI Accelerators Using the RISC-V Ecosystem and Digital Twins." In *International Conference on High Performance Computing*, pp. 500-514. Cham: Springer Nature Switzerland, 2023. https://doi.org/10.1007/978-3-031-40843-4_37

[35] Verma, Vaibhav, Tommy Tracy II and Mircea R. Stan. "EXTREM-EDGE—EXtensions To RISC-V for Energy-efficient ML inference at the EDGE of IoT." *Sustainable Computing: Informatics and Systems* 35 (2022): 100742. https://doi.org/10.1016/j.suscom.2022.100742

[36] He, Zicheng, Ao Shen, Qiufeng Li, Quan Cheng and Hao Yu. "Agile hardware and software co-design for RISC-V-based multi-precision deep learning microprocessor." In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, pp. 490-495. 2023. https://doi.org/10.1145/3566097.3567871