IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024

DOI: 10.17148/IARJSET.2024.11798

RISC-V Microarchitecture Design on FPGA

Mr. Praveen A[1], Shwetha V[2], Thushar Cherian[3], Prayag Singh[4], Varshith S[5]

Assistant Professor, Department of ECE, K S Institute of Technology, Bengaluru, India[1]

Undergraduate Students, Department of ECE, K S Institute of Technology, Bengaluru, India[2,3,4,5]

Abstract: This project presents the design and implementation of a single cycle RISC-V RV32I processor on FPGA using Xilinx ISE Design Suite. RISC-V is an open standard instruction set architecture that provides flexibility, scalability and privilege from proprietary constraints, making it an excellent choice for educational purposes. The processor is designed using Verilog HDL, includes essential components such as the instruction fetch, decode, execute, memory access and write-back stages.

The entire design is synthesized and implemented on a Spartan-6 FPGA board. This project demonstrates the feasibility and effectiveness of implementing a single cycle RISC-V processor on FPGA, providing valuable insights into processor design and hardware implementation.

Keywords: RISC-V single cycle processor design, RV32I Instruction Set, Spartan 6 FPGA, Xilinx ISE Design Suite.

I. INTRODUCTION

The rapid advancement of digital technologies has promoted a growing need for efficient and flexible processors. Among these, the RISC-V instruction set architecture (ISA) has played a significant role due to its open-source nature, modularity and simplicity. The RISC-V is an open and extensible ISA developed by the University of California, Berkeley, designed to support a wide range of applications, from small embedded systems to large scale data center. This project focuses on the designing a single cycle microarchitecture for RISC-V RV32I processor using Verilog HDL and implementing it on a Spartan 6 FPGA board. The RISC-V RV32I subset is a 32-bit ISA that includes a basic integer instruction set, making it ideal for educational purposes and small-scale applications. The RV32I instruction set consists of load, store, arithmetic, logical, and control flow instructions, providing a comprehensive foundation for understanding processor functionality.

A single cycle processor carries out one instruction in a single clock cycle. Therefore, in this architecture, each instruction is executed in a single clock cycle, encompassing instruction fetch, decode, execution, memory access and write-back stages. Despite its simplicity, the single cycle design serves an excellent educational tool for understanding the fundamental concepts of processor operation and microarchitecture design. The design will be divided into datapath and control unit, each component crafted to ensure efficient execution of the instructions. The integration of these components will be carried out under a top module which will then be deployed on Spartan 6 FPGA board for validation and testing.

FPGAs are widely used for prototyping and educational purposes due to their flexibility and reconfigurability. These devices allow designers to implement and test various architectural features and optimizations directly in hardware, providing valuable insights into real-world processor design challenges and trade-offs. The Spartan-6 FPGA is a family of Field-Programmable Gate Arrays (FPGAs) developed by Xilinx, designed to deliver high performance and low power consumption at an affordable cost. One of the standout features of Spartan-6 FPGAs is their power efficiency. These FPGAs are designed to operate with minimal power consumption, which is crucial for applications where energy efficiency is a priority. This makes them particularly suitable for battery-powered devices and other power-sensitive applications.By successfully implementing a single-cycle RISC-V processor on an FPGA, this project not only reinforces the fundamental principles of computer architecture but also provides a solid foundation for exploring more advanced processor designs and optimizations in future endeavors.

IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024

DOI: 10.17148/IARJSET.2024.11798

Why RISC-V?

Table I. Comparison of ISAs

ISA	CHIPS	ARCHITECTURE LICENSE	COMMERCIAL CORE IP	ADD OWN INSTRUCTIONS	OPEN-SOURCE CORE IP
x86	Yes, three vendors	No	No	No	No
ARM	Yes, many vendors	Yes, expensive	Yes, one vendor	No	No
RISC-V	Yes, many vendors	Yes, free	Yes, many vendors	Yes	Yes, many available

II. LITERATURE SURVEY

[1] Pushpalatha K N, Anmol Singh, Arpit Kumar, Abhishek Singh, Anirudh Reddy R. “Design And Implementation Of RISC-V ISA (RV32IM) On FPGA” (2023)

The paper "Design and Implementation of RISC-V ISA (RV32IM) on FPGA" by Pushpalatha K N, Anmol Singh, Arpit Kumar, Abhishek Singh, and Anirudh Reddy R. presents an in-depth study and practical application of the RISC-V instruction set architecture, specifically the RV32IM subset, in the context of FPGA implementation. The paper begins with a comprehensive overview of the RISC-V architecture, highlighting its advantages over traditional proprietary ISAs. The RV32IM subset, which includes basic integer instructions (RV32I) and multiplication and division instructions (M), is specifically addressed. The authors discuss the modular nature of RISC-V, which allows for the selective implementation of instruction subsets tailored to specific application requirements. Through detailed design methodology, thorough validation, and performance analysis, the authors provide valuable insights into the challenges and solutions associated with FPGA-based processor design. This work not only contributes to the academic understanding of RISC-V but also serves as a valuable resource for practical applications and further research in the field of digital design and computer architecture.

[2] Enfang Cui, Tianzheng Li, And Qian Wei.“RISC-V Instruction Set Architecture Extensions: A Survey” (2023)

The paper "RISC-V Instruction Set Architecture Extensions: A Survey" by Enfang Cui, Tianzheng Li, and Qian Wei provides a comprehensive overview of the various extensions developed for the RISC-V instruction set architecture (ISA). As RISC-V continues to gain momentum in both academic and industrial domains, its open-source nature allows for extensive customization and enhancement through ISA extensions. The survey begins by discussing the fundamental aspect of the RISC-V architecture, emphasizing its modular design and the rationale behind the creation of various ISA extensions. The authors categorize the extensions into several groups based on their functionalities, such as performance enhancement, security, and application-specific extensions. The survey delves into the implementation challenges and strategies for integrating these extensions into RISC-V processors. It discusses the trade-offs involved in extending the ISA, such as the balance between performance gains and increased complexity or power consumption. The authors also highlight the importance of maintaining backward compatibility with the base ISA to ensure broad adoption and support.

[3] Mr.Rajkumar D.Komati1, Aditya Kolekar, Kunal Kasbekar, Ms.Avanti Godbole. “Design and Implementation Of 16-Bit RISC-V Processor On FPGA” (2020)

The paper "Design and Implementation of 16-Bit RISC Processor on FPGA" by Rajkumar D. Komati, Aditya Kolekar, Kunal Kasbekar, and Avanti Godbole presents an in-depth exploration of the development and realization of a 16-bit Reduced Instruction Set Computing (RISC) processor on an FPGA platform. The core of the research focuses on the

IARJSET
ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024
DOI: 10.17148/IARJSET.2024.11798

design methodology for a 16-bit RISC processor. The authors detail the architecture's main components, including the arithmetic logic unit (ALU), register file, instruction fetch unit, control unit, and memory interface. The implementation process on an FPGA is discussed, with specific reference to the chosen FPGA platform. The authors describe the synthesis, place, and route processes, addressing challenges such as resource constraints and timing optimization. The FPGA's reconfigurable nature allows for iterative testing and refinement of the processor design. The paper details the 16-bit instruction set designed for the processor, including arithmetic, logical, control flow, and data transfer instructions. The design of the instruction set aims to balance simplicity with functionality, ensuring that the processor can perform essential operations efficiently.

[4] Deepika R, Gopika Priyadharsini S M, Muthu Malar, Vivek Anand. "Microarchitecture Based RISC-V Instruction Set Architecture for Low Power Application" (2022)

The paper "Microarchitecture Based RISC-V Instruction Set Architecture for Low Power Application" by Deepika R, Gopika Priyadharsini S M, Muthu Malar, and Vivek Anand (2022) delves into the design and implementation of a RISC-V microarchitecture tailored specifically for low-power applications. The authors emphasize the significance of the RISC-V ISA's open-source nature, which provides the flexibility needed to optimize the architecture for power efficiency without the constraints of licensing fees associated with proprietary ISAs. The research focuses on the development of a microarchitecture that balances performance with power consumption, making it suitable for applications where energy efficiency is paramount, such as in wearable devices, IoT sensors, and other embedded systems. The authors outline their approach to reducing power usage by incorporating various techniques, such as clock gating, dynamic voltage scaling, and power gating. These techniques are integrated into the RISC-V processor design to minimize active and standby power consumption. Overall, this research contributes to the ongoing efforts to develop energy-efficient computing solutions by leveraging the flexibility of the RISC-V ISA. The findings and methodologies presented in this paper can serve as a foundation for further advancements in low-power microarchitecture design, potentially influencing the development of next-generation processors for various low-power applications.

III. METHODOLOGY

A. Block Diagram

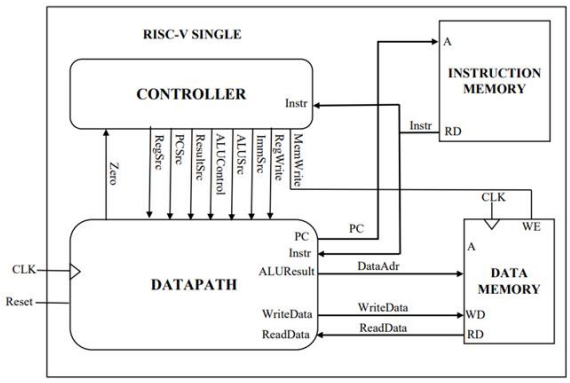

Figure 1.Block Diagram of RISC-V single cycle processor

The Figure.1, shows the RISC-V single cycle processor block diagram which consists of the fundamental components necessary for executing RISC-V instructions within a single clock cycle. The central component is the Datapath, which mainly includes the Program Counter (PC) for tracking instruction addresses, the Instruction Memory (IMEM) for storing the program's instructions, and the Register File for holding the processor's working data. An Arithmetic Logic Unit (ALU) performs arithmetic and logical operations, while Data Memory (DMEM) is used for data storage during execution.

The Control Unit generates the necessary signals to orchestrate the Datapath operations, ensuring each instruction is fetched, decoded, executed, and results are written back correctly. Together, these components interact within a single clock cycle, ensuring streamlined instruction execution.

IARJSET
ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024
DOI: 10.17148/IARJSET.2024.11798

B. Working

Datapath

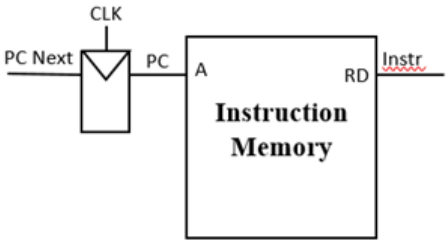

Figure 2. Instruction Memory

The program counter contains the address of the instruction to execute. The first step is to read this instruction from instruction memory. The PC is connected to the address input of the instruction memory as shown in Figure.2. The instruction memory fetches the 32-bit instruction, labelled Instr. The processor's actions depend on the specific instruction that was fetched.

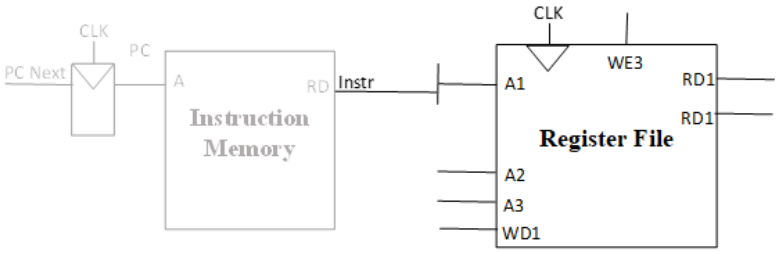

Figure 3. Read source from Register File

The next step is to read the source register containing the base address. The base register is specified in the rs1 field of the instruction. These bits of the instruction connect to the A1 address input of the register file, as shown in Figure 3. The register file reads the register value onto RD1. The offset is stored in the 12-bit immediate field of the instruction. It is a signed value, so it must be sign-extended to 32 bits. Sign extension is performed by an Extend unit. It receives the 12-bit signed immediate and produces the 32-bit sign-extended immediate.

The ALU can perform many operations, as was described in Table 3, ALUControl specifies the operation. The ALU receives 32-bit operands and generates a 32-bit ALUResult. The result is sent to the data memory as the address to read.

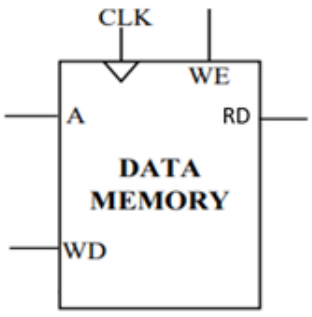

Figure 4. Data Memory

The memory address from the ALU is provided to the address (A) port of the data memory as shown in Figure 4. The data is read from the data memory onto the ReadData bus and then written back to the destination register at the end of the cycle. A control signal called RegWrite (register write) is connected to WE3 in Figure 4., port 3's write enable input, and is asserted so that the data value is written into the register file. The write takes place on the rising edge of the clock at the end of the cycle. While the instruction is being executed, the processor must also compute the address of the next instruction, that is PCNext, therefore PC is incremented to point next instruction address. The new address is written into the program counter on the next rising edge of the clock.

IARJSET
ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024
DOI: 10.17148/IARJSET.2024.11798

Control Unit

The control unit, is also referred to as the controller or the decoder, because it decodes what the instruction should do. It is partitioned into two major parts: the Main Decoder, which produces most of the control signals, and the ALU Decoder, which determines what operation the ALU performs.

The Main Decoder determines the instruction type from the opcode and then produce the appropriate control signals for the datapath. The Main Decoder generates most of the control signals for the datapath.

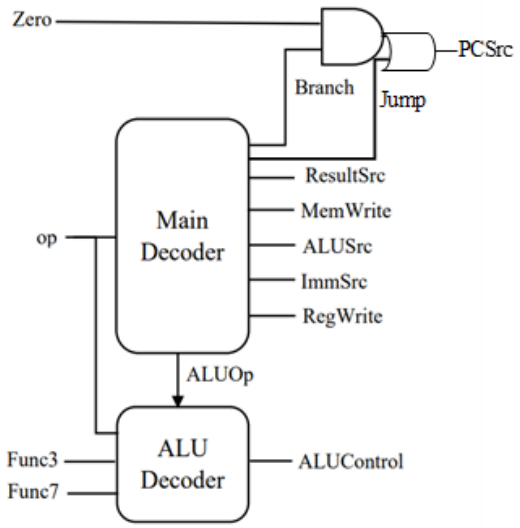

Figure 5. Controller

Table II. Main Decoder Truth Table

Instruction	Op	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp
lw	0000011	1	00	1	0	01	0	00
sw	0100011	0	01	1	1	xx	0	00
R-type	0110011	1	xx	0	0	00	0	10
beq	1100011	0	10	0	0	xx	1	01
I-type ALU	0010011	1	00	1	0	00	0	10
jal	1101111	1	11	x	0	10	0	xx

It produces internal signals Branch and ALUOp, signals used within the controller. The logic for the Main Decoder can be developed from the truth table shown in Table II, using various techniques for combinational logic design. The ALU Decoder produces ALUControl based on ALUOp and func3. In the case of the sub and add instructions, the ALU Decoder also uses funct7(5) and op(5) to determine ALUControl, as given in the Table III. ALUOp of 00 indicates add (e.g., for loads or stores). ALUOp of 01 indicates subtract (e.g., to compare two numbers for branches). ALUOp of 10 indicates an R-type ALU instruction where the ALU Decoder must look at the func3 field and sometimes also the op(5) and func7(5) bits to determine which ALU operation to perform (e.g., add, sub, and, or, slt).

Each component is designed using Verilog HDL, ensuring modularity and ease of integration. The datapath and control unit are integrated under a top-level module, which coordinates their interactions to ensure the correct execution of instructions.

IARJSET
ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024
DOI: 10.17148/IARJSET.2024.11798

Table III. ALU Decoder Truth Table

ALUOp	Func3	{op(5),func7(5)}	ALUControl	Instruction
00	x	x	0000	lw,sw
01	x	x	0001	beq
10	000	00,01,10	0000	add, addi
	000	11	0001	sub
	001	xx	0100	sll, slli
	010	xx	0101	slt,slti
	011	xx	0110	sltu, sltiu
	100	xx	0111	xor, xori
	101	00,01,10	1000	srl, srli
	101	11	1001	sra, srai
	110	xx	0011	or, ori
	111	xx	0010	and, andi

The entire design is synthesized and implemented on a Spartan 6 FPGA board. The assembly code is converted into binary or hexadecimal format and loaded into the instruction memory of the FPGA-implemented processor. Upon execution, the results are stored in the data memory, referred to as result (1). The same assembly programs are executed on an online RISC-V simulation platform, and the results are recorded as result (2).

Finally, result (1) from the FPGA implementation is compared with result (2) from the online RISC-V platform to validate the correctness of the processor design. Matching results confirm the proper functioning of the designed processor.

IV. HARDWARE & SOFTWARE USED

A. Hardware used - Spartan 6 FPGA Kit

Figure 6. Spartan 6 FPGA

An FPGA, Field Programmable Gate array, is a semiconductor device, that is based on matrix of configurable logic blocks, connected via programmable interconnects. FPGAs are subset of logic devices referred to as programmable logic devices. A FPGA configuration is generally written using a hardware description language (HDL). The logic blocks of an FPGA can be configured to perform complex combinational functions, or act as simple logic gates like AND and XOR. FPGAs also have a role in embedded system development due to their capability to start system software development simultaneously with hardware, enable system performance simulations at a very early phase of the development, and allow various system trials and design iterations before finalizing the system architecture.

Xilinx produced the first commercially viable field-programmable gate array in 1985 – the XC2064. The Spartan-6 FPGA board, developed by Xilinx, is a versatile and widely used platform in the area of digital logic design, embedded systems, and rapid prototyping. The Spartan-6 family includes various models with different capacities and features, catering to a broad spectrum of requirements. The Spartan-6 family is designed to deliver low power consumption, making it ideal for battery-powered and portable applications.

IARJSET
ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024
DOI: 10.17148/IARJSET.2024.11798

B. Software used – Xilinx ISE

Figure 7. Xilinx ISE Design Suite

Xilinx ISE (Integrated Software Environment) is a comprehensive software tool, provided by Xilinx for the synthesis and analysis of HDL (Hardware Description Language) designs, enabling developers to design, simulate, and implement FPGA and CPLD (Complex Programmable Logic Device) designs. Xilinx ISE was the flagship IDE from Xilinx for FPGA design for almost two decades since its introduction in 1992. It enabled the complete FPGA design flow from design entry to bitstream generation using integrated tools for synthesis, place and route, timing analysis, simulation and debugging.

The Xilinx ISE supports the Spartan 6, as well as older devices which include CPLDs (CoolRunner and XC9500). It also supports various methods of design entry, including schematic capture, HDL (VHDL, Verilog), and state machines. Various integrated tools for functional simulation, timing analysis, and formal verification to ensure the correctness of the design can be used. Xilinx's algorithms for synthesis allow designs to run up to 30% faster than competing programs, and allows greater logic density which reduces project time and costs.

V. RESULTS

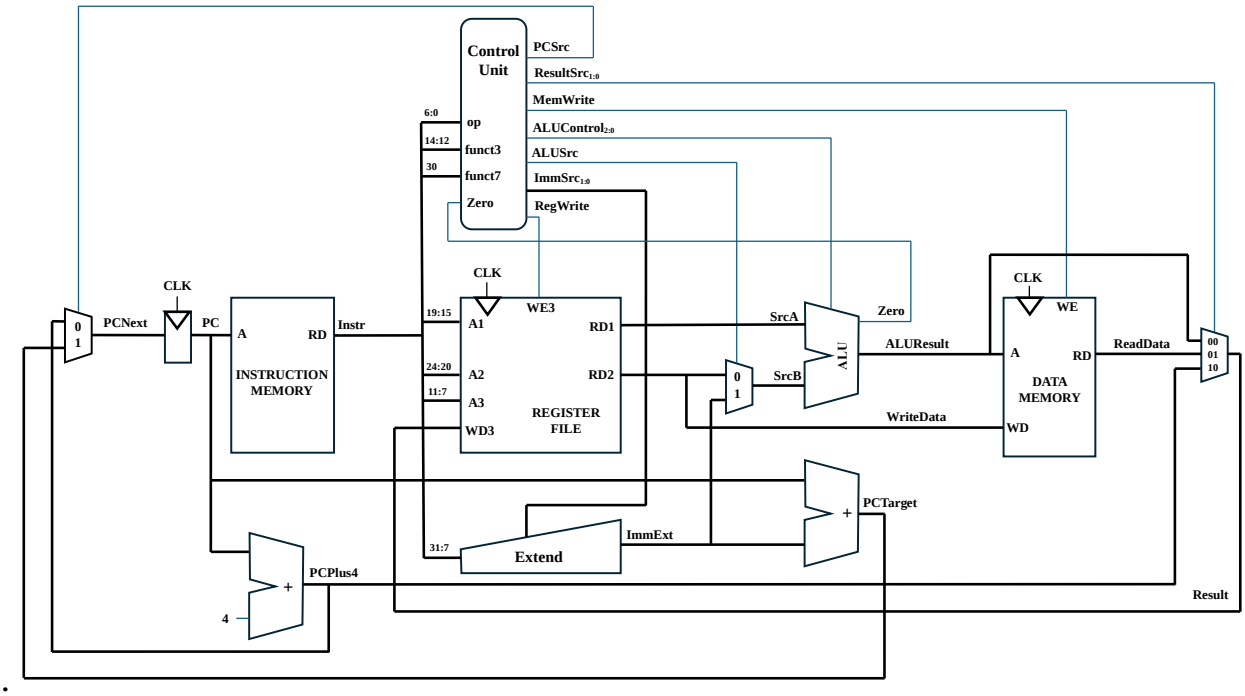

Figure 8. RISC-V single cycle processor

The above complete single cycle processor is designed using Verilog HDL on Xilinx ISE. This design is tested using the ALP program shown in Table IV. A test bench program is written to simulate the functionality of the designed single cycle RISC-V processor. The simulation is successful and it is as shown in the Figure 9.

IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024

DOI: 10.17148/IARJSET.2024.11798

Table IV. RISC-V ALP and its equivalent hexadecimal

RISC-V ASSEMBLY	HEXADECIMAL EQUIVALENT
addi x2, x0, 5	00500113
addi x3, x0, 12	00C00193
addi x7, x3, -9	FF718393
or x4, x7, x2	0023E233
and x5, x3, x4	0041F2B3
add x5, x5, x4	004282B3
beq x5, x7, end	02728863
slt x4, x3, x4	0041A233
beq x4, x0, around	00020463
addi x5, x0, 0	00000293
slt x4, x7, x2	0023A233
add x7, x4, x5	005203B3
sub x7, x7, x2	402383B3
sw x7, 84 (x3)	0471AA23
lw x2, 96 (x0)	06002103
add x9, x2, x5	005104B3
jal x3, end	008001EF
addi x2, x0, 1	00100113
add x2, x2, x9	00910133
sw x2, 0x20 (x3)	0221A023
beq x2, x2, done	00210063

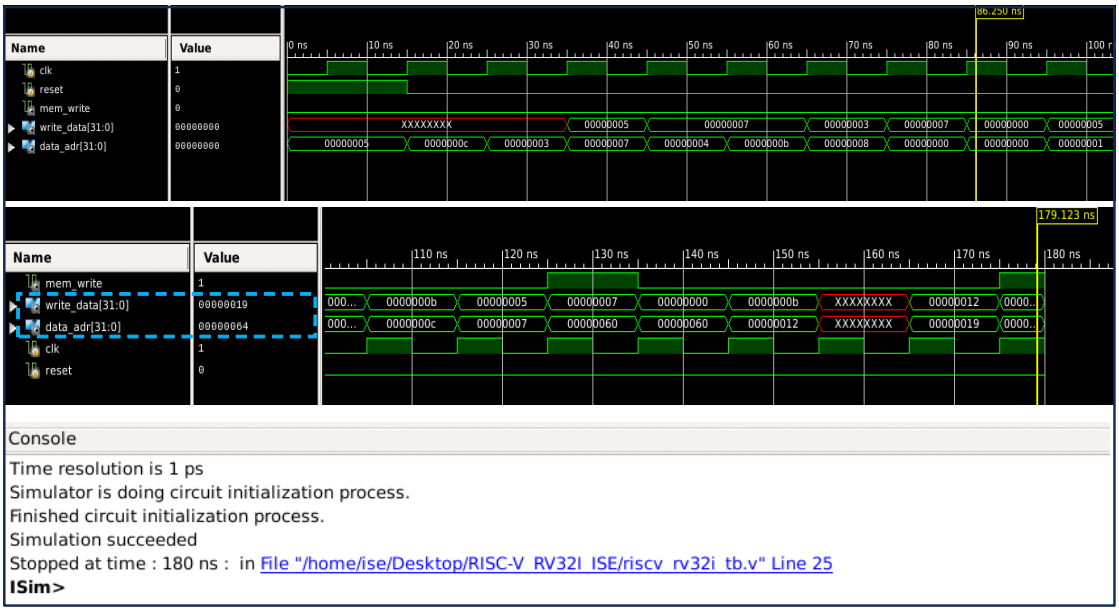

Figure 9. Simulation Output

IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024

DOI: 10.17148/IARJSET.2024.11798

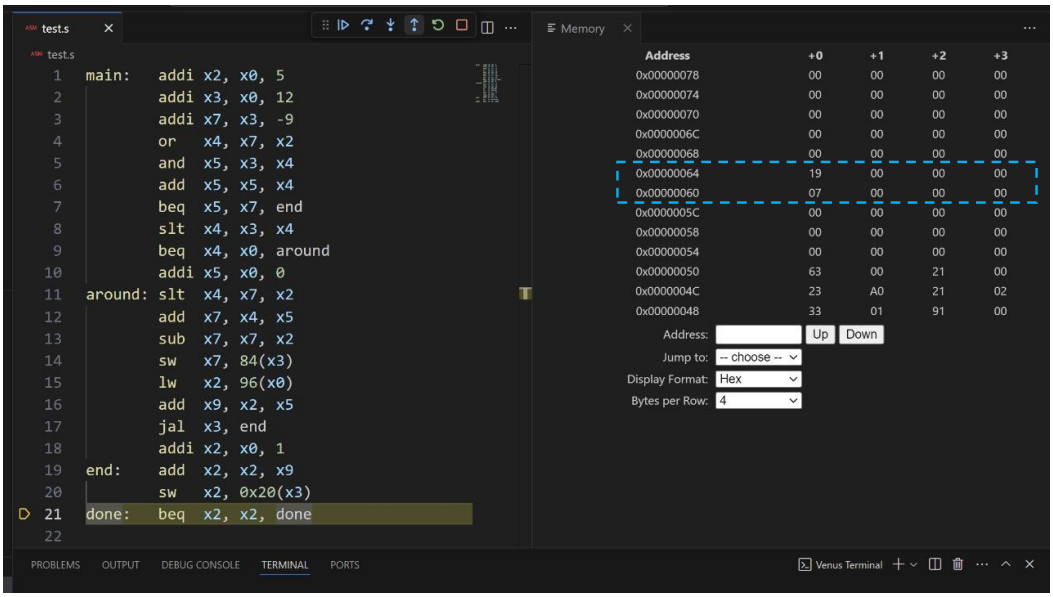

Figure 10. Venus RISC-V simulator output

The same ALP code is executed on Venus RISC-V simulator on Visual Studio Code to compare and validate the designed RISC-V processor output. The output of the RISC-V simulator is shown in Figure 10. Note that the ALP contains two sw instructions that writes data into data memory as follows,

1. 0x07 at the address 0x60
2. 0x19 at the address 0x64

The program produces the correct results (above specified sw operations) only if all the functions are functioning correctly. If the designed hardware is buggy, the expected result may not be obtained.

Hardware Utilizations:

```
================================================================
HDL Synthesis Report

Macro Statistics
# RAMs                                               : 4
 128x32-bit single-port RAM                          : 1
 32x32-bit dual-port RAM                             : 2
 64x32-bit single-port Read Only RAM                 : 1
# Adders/Subtractors                                 : 5
 21-bit adder                                        : 1
 32-bit adder                                        : 4
# Registers                                          : 5
 21-bit register                                     : 1
 32-bit register                                     : 4
# Comparators                                        : 2
 32-bit comparator greater                           : 2
# Multiplexers                                       : 34
 1-bit 2-to-1 multiplexer                            : 22
 1-bit 3-to-1 multiplexer                            : 2
 32-bit 11-to-1 multiplexer                          : 1
 32-bit 2-to-1 multiplexer                           : 7
 8-bit 2-to-1 multiplexer                            : 2
# Logic shifters                                     : 3
 32-bit shifter logical left                         : 1
 32-bit shifter logical right                        : 2
# Xors                                               : 1
 32-bit xor2                                         : 1

================================================================
```

Figure 11. HDL synthesis report

IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024

DOI: 10.17148/IARJSET.2024.11798

```
Device Utilization Summary:

Slice Logic Utilization:
  Number of Slice Registers:                    52 out of   4,800    1%
    Number used as Flip Flops:                  52
    Number used as Latches:                      0
    Number used as Latch-thrus:                  0
    Number used as AND/OR logics:                0
  Number of Slice LUTs:                        685 out of   2,400   28%
    Number used as logic:                      682 out of   2,400   28%
      Number using O6 output only:             588
      Number using O5 output only:              47
      Number using O5 and O6:                   47
      Number used as ROM:                        0
    Number used as Memory:                       0 out of   1,200    0%
    Number used exclusively as route-thrus:      3
      Number with same-slice register load:      0
      Number with same-slice carry load:         3
      Number with other load:                    0

Slice Logic Distribution:
  Number of occupied Slices:                   212 out of     600   35%
  Number of MUXCYs used:                       160 out of   1,200   13%
  Number of LUT Flip Flop pairs used:          685
    Number with an unused Flip Flop:           633 out of     685   92%
    Number with an unused LUT:                   0 out of     685    0%
    Number of fully used LUT-FF pairs:          52 out of     685    7%
    Number of slice register sites lost
      to control set restrictions:               0 out of   4,800    0%
```

Figure 12. Device Utilization Summary

RTL Schematic:

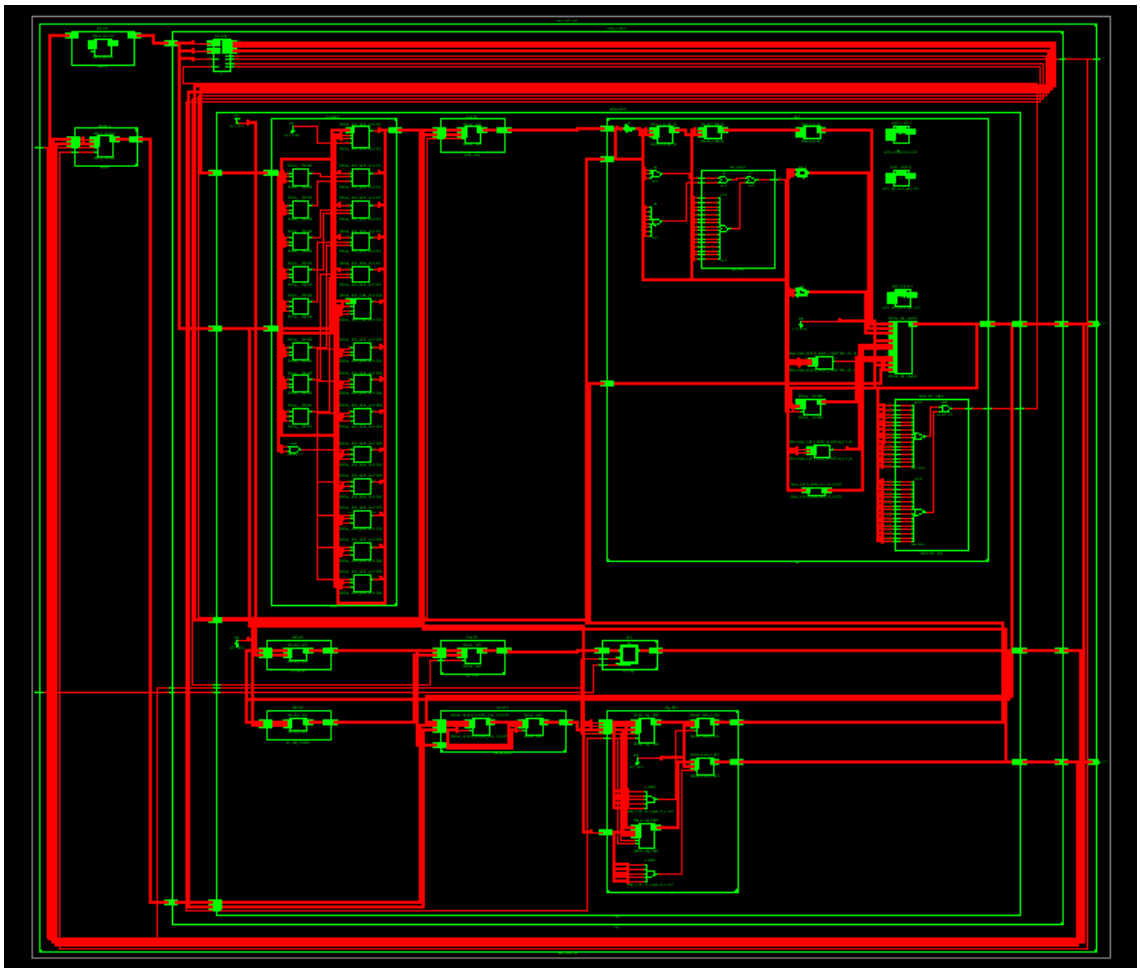

Figure 13. RTL Schematic

IARJSET
ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 11, Issue 7, July 2024
DOI: 10.17148/IARJSET.2024.11798

VI. CONCLUSION & FUTURE SCOPE

This project was a significant undertaking that provided us a hands-on experience in digital design of a RISC-V processor on FPGA platform. The project successfully demonstrated the feasibility of designing and deploying a single-cycle processor on an FPGA. Key learning outcomes included understanding the RISC-V instruction set architecture, digital design of a processor using Verilog Hardware Description language (VHDL), and gaining proficiency in using Xilinx ISE for FPGA synthesis and implementation.

The project highlights the flexibility and efficiency of the single-cycle RISC-V architecture and the programmability of FPGA platforms like the Spartan 6. It also underscores the advantages of open-source architectures in education and prototyping, allowing for customization and optimization to meet specific needs.

In the Future work, we intend to extend the single-cycle processor to a pipelined architecture to improve performance by overlapping instruction execution. To develop a multi-core RISC-V processor on the FPGA to explore parallel processing and multi-threading capabilities.

REFERENCES

[1] Pushpalatha K N, Anmol Singh, Arpit Kumar, Abhishek Singh, Anirudh Reddy R. “Design and Implementation of RISC-V ISA (RV32IM) On FPGA” (2023)
[2] Enfang Cui, Tianzheng Li, and Qian Wei. “RISC-V Instruction Set Architecture Extensions: A Survey” (2023)
[3] Mr. Rajkumar D. Komati1, Aditya Kolekar, Kunal Kasbekar, Ms. Avanti Godbole. “Design and Implementation of 16-Bit RISC Processor On FPGA” (2020)
[4] Deepika R, Gopika Priyadharsini S M, Muthu Malar, Vivek Anand. “Microarchitecture Based RISC-V Instruction Set Architecture for Low Power Application” (2022)
[5] Akshay Birari et al., “A RISC-V ISA Compatible Processor IP,” 24th International Symposium on VLSI Design and Test (2020).
[6] Aneesh Raveendran et al., “A RISC-V Instruction Set Processor Microarchitecture Design and Analysis,” International Conference on VLSI Systems, Architectures, Technology and Applications (2016).
[7] Jikku Jeemon, “Low power pipelined 8-bit RISC processor design and implementation on FPGA”, ICCICCT 2015.
[8] https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf