Design and FPGA Implementation of Five Stage Pipelined RISC -V Processor

Faeq Hussain
Electronics and Communication Engineering
National Institute of Technology Rourkela
Rourkela, India
120ec0420@nitrkl.ac.in

Dr. Santanu Sarkar
Electronics and Communication Engineering
National Institute of Technology Rourkela
Rourkela, India
sarkars@nitrkl.ac.in

Abstract— **This paper describes the design and implementation of RISC-V 5-Stage pipelined processor on Basys-3 FPGA Board. The RISC-V core is based on RV32I instruction set architecture. The five-stages of pipelines are namely, instruction fetch, instruction decode, execute, memory access and write back stages with a hazard control unit which contains a stall controller. The proposed RISC-V processor is designed with a Harvard storage structure. RISC-V processor was pipelined to increase the throughput and the maximum operating frequency. For single cycle RISC-V processor maximum 31.6MHz operating frequency is achieved. The five-stage pipelined RISC-V processor is implemented on the Basys-3 board at a clock frequency of 50MHz and it had worst net slack (WNS) of 8.6ns which indicates that the maximum operating frequency can be 87.86MHz and nearly 2.78 times the maximum frequency of the single cycle processor. The power consumption also improved from 244mW to 96mW.**

Keywords— ***RISC-V, FPGA, RISC, RV32I ISA, Pipelined***

I. INTRODUCTION

The 21st century is the era of fast computing with high speed processors. Currently the industry leaders like the ARM or Intel x86 Architectures are intellectual property (IP) protected and hence are costly and it is difficult to access them for learning and research purposes. The study of processors and further implementation is highly required for the growth of the research in the fast-growing field of processor design. An open-source architecture which can support all the applications can be beneficial for research and growth for both academic and industries. As RISC-V is an open-source ISA it provides a better opportunity to learn [1]. Using RISC-V architecture anybody can create their own microprocessor core for research or study and can construct any embedded devices. RISC-V is based upon the basics of the RISC architecture [2]. The development of RISC-V processors like SHAKTI processor from IIT Madras and VEGA processors from CDAC in India has been great inspiration for this research work. The instructions encoding style is highly regular, simple and has made huge compromises for achieving a fast processing core [1-2].

Our primary objective of this research work is to design a RISC-V Processor capable of executing instructions from RV32I ISA manual. This offers a great learning experience and as well as we the designers can execute various operations from simple to complex programs using the application specific processor.
RISC based architectures have been used in both low-level applications and mobile systems by the beginning of the 21st century. The low power and low-cost embedded markets are dominated by the RISC based ARM architectures. Most of the android-based devices, Apple iPhone, or iPad and most of the hand-held devices use the ARM architectures. Due to its multiple versions that support 16bit, 32bit, and 64bit computing, ARM is now the industry leader in its sector. The MIPS line can currently be found in games like PlayStation Portable game consoles, Nintendo 64. RISC-V has a feature of open-source ISA whereas other architectures lack some features like ARM and MIPS are proprietary standard and are royalty based which makes it difficult for research and academic developments.

In this research, initially a single cycle RISC-V processor is designed. The basics of the design of a single cycle architecture can be referred to in [3-4]. To improve the operating frequency and throughput, finally five stage pipelining architecture is implemented. Pipelining helps in reducing the cycle time and improving the time delay between execution of different instructions in the processor [5-9]. The hazard associated with pipelining can be understood from [10]. This paper also discusses the different types of hazards faced during pipelining and its possible solution to overcome it. The design is finally implemented in Basys-3 board and the proposed processor is verified with results for different inputs.

The paper is organized as follows. The RV32I architecture is discussed in Section II. Section III describes the design of the proposed RISC-V processor. The simulation results and the FPGA implementations of the proposed processor are discussed in Section IV. Conclusions are drawn in Section V. Finally, the future scope of the work is described in Section VI.

II. RV32I ARCHITECTURE

A. Instruction Set Architecture(ISA)

RV32I is designed to form a compiler target and support modern operating system environments [2]. It is also designed to reduce the hardware requirement for a minimal implementation. RV32I contains 47 unique instructions. The RISC-V ISA is defined as a base integer ISA which is a necessicity and must be present in all the designs and along with it one can have other optional extensions. RV32I contains 47 instructions as mentioned in the manual [1] and in this work we are able to implement 37 instructions successfully.

The program counter holds the address to the next instruction which is further used to access the instruction from instruction memory, then it is further decoded to get the values from the register file. The ALU is repsonsible for the execution of arithmatic and logical functions then depending upon the type of instruction the the result will be stored in the data memory or the register file.

There are 6 types of instruction format:

- R-type (register-register instructions)
- I-type (Register-Immediate and load instructions)
- B-type (Conditional branching instructions)
- J-type (Unconditional branching instructions)
- S-type (Store instructions)
- U-type (Upper Immediate instructions)

These instructions include ADD, SUB, SLL, SLT, ADDI, SLTI, ANDI, ORI and many more. Table I, decribes the major instructions of the RV32I ISA.

Table I: Some Major Instructions of RV32I

Table Head	**SOME MAJOR INSTRUCTIONS**	
	Name	***Function***
1	***ADD***	***Addition***
2	***SUB***	***Subtraction***
3	***SLL***	***Shift Logic Left***
4	***AND***	***Bitwise AND Logic***
5	***OR***	***Bitwise OR Logic***
6	***BEQ***	***Branch if equal***
7	***BNE***	***Branch if not equal***
8	***LW***	***Load word***
9	***SW***	***Store word***
10	***JAL***	***Jump and link***
11	***LUI***	***Load upper immediate***
12	***AUIPC***	***Add upper immediate to pc***

B. Processor Microarchitecture

The proposed architecture is sbased on the designs as mentioned in [2]. The Verilog code of the top module has been made very modular and readable. The main builing blocks of the processor are program counter, Instruction Memory, Data Memory, Register file and ALU. Fig. 1, shows the complete architecture of a single cycle RISC-V processor. Further this designed is pipelined into five stages to achieve better clock frequency [3-5].

C. Pipelining:

The single cycle RISC-V processor is pipelined to 5-Stages RISC-V processor. The 5 Stages consists of Instruction Fetch, Instruction Decode, Execution, Memory Access and Write-Back [4-6]. The order of the stages is as shown in Fig. 2.

Ideally it can be assumed that each stage takes about T units of time and so for example 7 instruction in a non-pipelined architecture will take 7*5T whereas in a pipelined architecture it will take 11T. So with pipelining the aim is to increase the throughput. Along with that the number of instructions executed will also increase significantly so the performance of the processor also increases.

IF
ID
EX
MA
WB

Figure 2. 5-Stages of Pipeline

D. Hazards[2][7]:

The problems faced while pipelining are called Hazards. Hazards prevent the next instruction to be executed in the next clock cycle. The 3 types of Hazards generally face are Structural Hazard, Data Hazard and Control Hazard[9].

III. RISC-V PROCESSOR DESIGN

The module was designed in such a way that it takes an input of 8-bit and stores it in the data memory location '0' and code is written such that the final output is stored at location '1'. With the help of input switches input is given by the user and output is displayed using the LEDs, and the final 10-bit output can be observed in the LEDs on the board. For this design we have used 64KB (2048*32) of ROM and 16KB (512*32) of RAM. The design is based on Harvard memory structure means there are different memories for instructions and data. The 5-Stages of pipelined RISC-V processor are Instruction Fetch, Instruction Decode, Execution, Memory Access and Write Back. Each stage is separated by a register

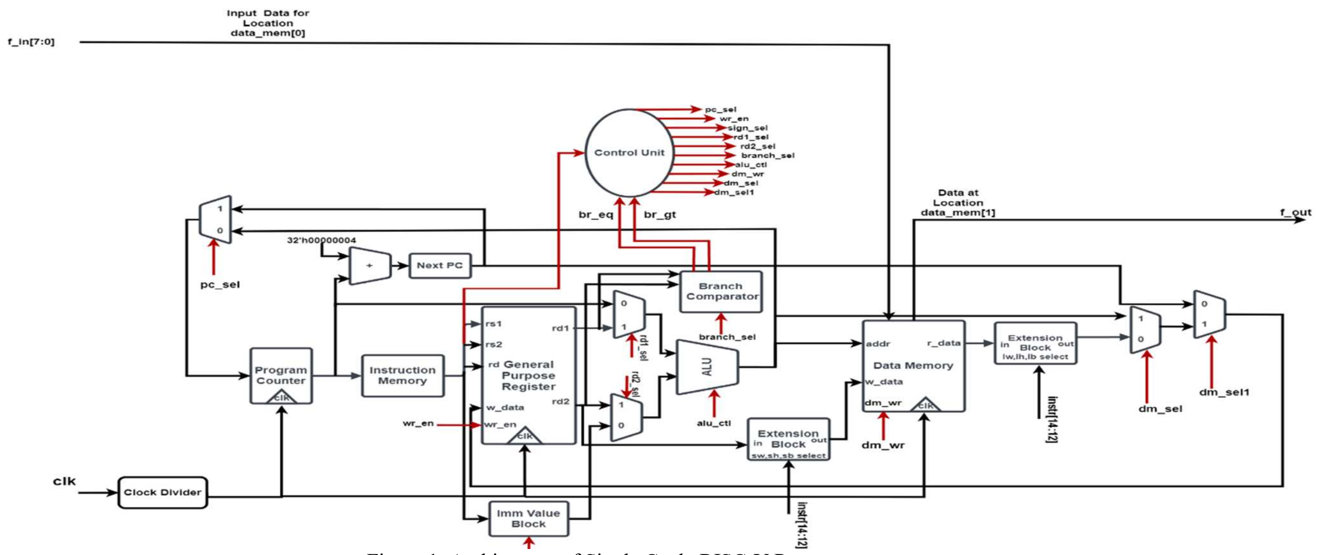


Figure 1. Architecture of Single Cycle RISC-V Processor

and along with that a hazard control unit is also designed to perform Data forwarding and Stall controlling operation.

Major Blocks used in the design:

A) Program Counter: This is a 32 bit register used to hold the address of the next instruction to be executed. The program counter (pc) value can be updated to either pc+4 or branching address depending on the instructions.

B) PC Adder: This is the program counter adder and is used to calculate the address of the next instruction.

C) Instruction Memory: It is read only memory of size 64KB (2048*32) is used to store the instructions. It is used to continuously read values based on the address given by the program counter.

D) Register file: It has 2 asynchronous read ports, 1 synchronous write port, one data input port, one write enables port and 2 data output port. The register file contains 32 registers, each with a width of 32 bits. The write enable signal is controlled by the instruction decoder block. The register r0 is set to 32'h00000000. The rest of the 31 registers can be used for reading and writing purposes.

E) Immediate Value Block: Its main purpose is to calculate the immediate value for the given instruction and then send it to the next stage.

F) ALU: The design uses a 32-bit ALU and takes in a 4-bit ALU control signal. It is responsible to perform all the arithmetic and logical operations in the processor from addition of two numbers to calculation of the next instruction address for jump statements.

G) Extension Block: This block is essential to select the data to format to be stored or loaded in the memory. If the instruction is lw or sw then complete 32-bit data is passed or for sh or lh the 16-bit data is selected and then sign extended after which it is passed or for sb or lb the 8-bit data is selected and then sign extended after which it is passed.

H) Data Memory: It consists of 1 asynchronous read port and 1 synchronous write port. It is a read and write memory of size 16KB (512*32).

I) Branch Comparator Block: It takes 2 inputs from data read from register file and returns br_gt and br_eq to the instruction decoder to decide if the branching should be done or not for branch type of instructions.

J) Instruction Decoder: This block is responsible for generating the control signals for the processor. The instruction decoder consists of a control unit responsible for the control signals and select lines and ALU decoder responsible for ALU control signals.

The 5-Stages of Pipeline:

A) Instruction Fetch Stage: This stage consists of program counter, program counter adder, Instruction Memory and the logic for branching. The program counter holds the address to the next intruction in the instruction memory. The program counter is either updated with the address of the next instruction sequentially(pc+4) or if the branching instruction is detected in the instruction decode stage then the processor enters into stalling for 3 clock cycles and no instruction from the instruction memory is passed(NOP is passed). The instruction is fetched from instruction memory and passed to the next stage.

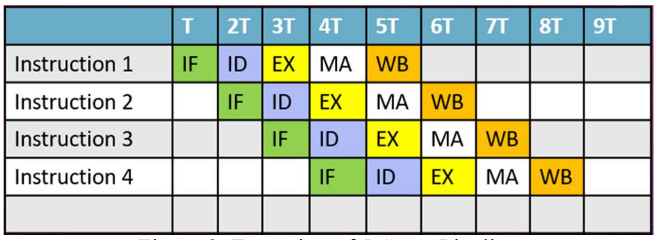
B) Instruction Decode Stage: After the instruction has been fetched, this stage is responsible for deciding the type of instruction and generating the read register values and control signals This stage mainly consists of register file, Immediate value block, Branch comaparator block and Instruction decoder which generates the control signal. This stage is where branch instruction is detected. Once the branch instruction is detected, the HCU will generate stall signals.

C) Execution Stage: Responsible for performing or executing the arithmatic and logical functions required with the register values, immidiate value or program counter. This stage mainly consists of ALU and extnesion block. The main purpose of this block is to perform the ALU operation and generate the data to be written in the data memory or back to the register file.

D) Memory Access: . This stage consists of Data memory block. This stage is mainly responsible for reading and writing in the data memory. If the instruction is related to data memory like store or load instruction then only the data memory. The dm_wr signal is by default set to 0 so read function is performed asynchronously whereas when dm_wr is 1 then write function is performed synchronously.

E) Write Back: This stage consists of muxes and extension block. The main purpose of this stage is that data is selected which will be written back into the register file. The data can be either the ALU result or next program counter value or the data memory read value. And depending on the type of instruction chosen may it be lw, lh or lb the size of the data memory read value will be selected and passed to the muxes.

Fig. 3 shows the normal execution of 4 instruction in a 5-Stage pipelined RISC-V processor.



	T	2T	3T	4T	5T	6T	7T	8T	9T
Instruction 1	IF	ID	EX	MA	WB				
Instruction 2		IF	ID	EX	MA	WB			
Instruction 3			IF	ID	EX	MA	WB		
Instruction 4				IF	ID	EX	MA	WB	

Figure 3. Execution of 5-Stage Pipeline

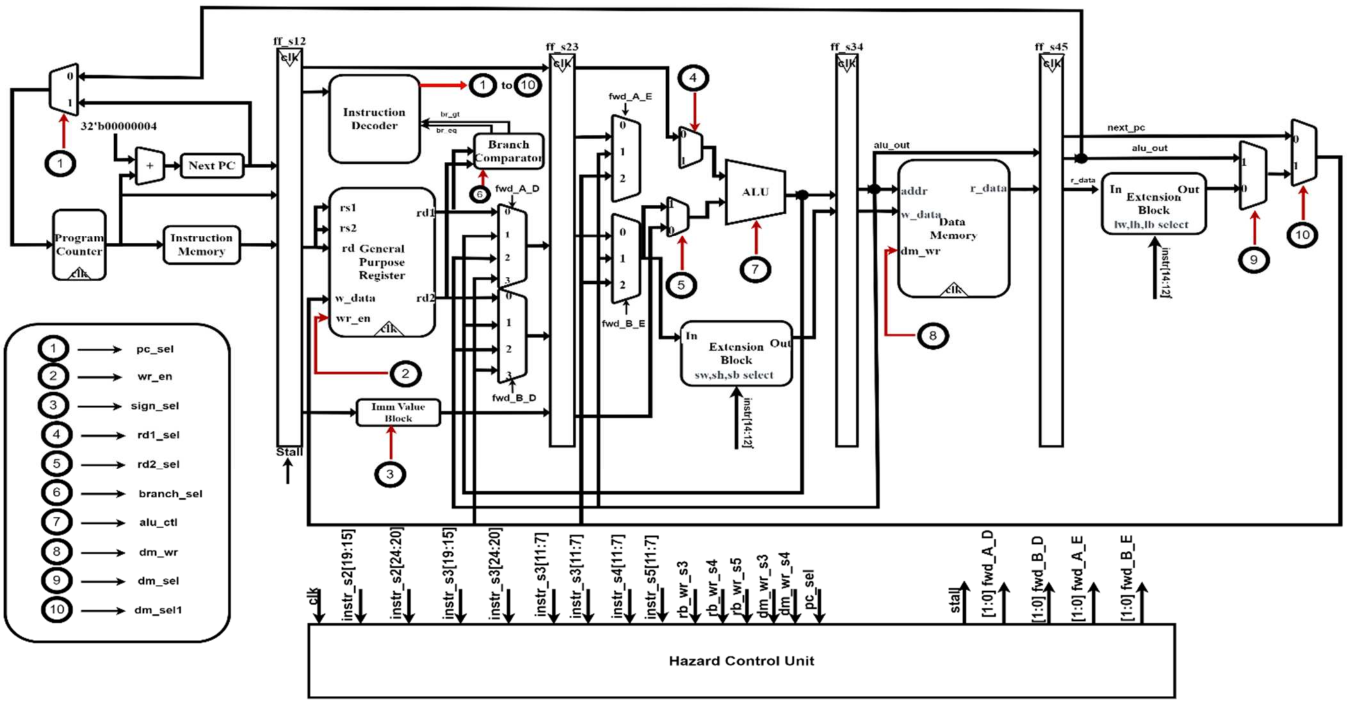


Figure 4. Architecture of 5-Stage Pipelined RISC-V Processor

The complete Architecture of the 5-Stage pipelined RISC-V core is shown in Fig. 4.

As discussed earlier there are 3 types of hazard that is to be taken care of

- Structural Hazard
- Data Hazard
- Control Hazard

The Structural Hazard is caused when the instruction to be executed does not get executed due to lack of resources. To eleminate this type of hazard, the design uses Harvard memory structure means there are separate memory structure for data and instruction so there will not be any conflict if both the instructions and data are to be accessed at the same point of time.

Data Hazard also known as pipeline hazard. This type of hazard occurs when the data required by the current instruction is not updated. So basically the data accessed is not present at that point of time. This kind of hazard is eliminated by using Data Forwarding technique.

Control Hazard also known as branch hazard. If a branch instruction is detected then the following instructions are not to be executed and should jump to the new instruction. If that does not happen then control hazard is said to have occurred. To eleminate this type of hazard, stalling is used.

Data Hazard and Control Hazard are eleminated by the following block Hazard Control Unit.

Hazard Control Unit(HCU): This block is responsible for checking if data forwarding is required. The forwarding units in the decode stage is controlled by fwd_A_D and fwd_B_D. And the forwarding units is the decode stage is controlled by fwd_A_E and fwd_B_E.

The forwarding control signals are determined by the register bank write signal and the data memory write signal in next stages. For example, if the register bank write signal is high in the preceding stages and the destination register is same as the source register of this stage, then the data is forwarded from that stage to the previous stage. Similar kind of mechanism is used for load and store instructions considering the data memory write signal.

The HCU contains another block called the stall controller which is responsible for generating the stall signals that will be low if the branching condition is false else if the branching condition is true or if it is an unconditional jump statement then the stall signal goes high and will stop the pipeline and pass NOP instruction which is addi x0,x0,0 and this will continue for the next 3 clock cycles.

After the NOP instructions the new instructions from the branch address are loaded into the PC which then continues to execute sequentially until another branch or jump instruction is encountered. Fig. 5 shows the working example of a sample code when a branching instruction is encountered at line 4, then as it passes to the instruction decode stage it is known that it is a branching instruction hence the stall cycle is activated for 3 clock cycles and once the instruction is completed it will either jump or continue sequentially.

For this case the program jumps to execute the instruction add x2, x3, x4.

INSTRUCTIONS\Clock Cycle	1T	2T	3T	4T	5T	6T	7T	8T	(N)T	(N+1)T	(N+2)T	(N+3)T	(N+4)T
addi x1,x0,5	IF	ID	EX	MA	WB								
or x9,x8,x4		IF	ID	EX	MA	WB							
and x5,x3,x2			IF	ID	EX	MA	WB						
beq x2,x6,12				IF	ID	EX	MA	WB					
and x11,x3,x7					IF	NOP	NOP	NOP					
add x2,x3,x4									IF	ID	EX	MA	WB

Figure 5. Stalling Example

The core has been designed in such a way that the user can interact with the core. For input the user can give an input of 8 bits and there will be an output port of size 10 bits. The input will be given with the help of switches and the output is displayed through LEDs. The input is continuously stored in the data memory at location "0" and the output is stored in the data memory location "1".

For any program loaded into instruction memory may it be Sum of N numbers or Nth Fibonacci term. The value of N is given by the user and can be varied at any instant of time through the switches, and it will be stored at location "0". And the program is written in such a way that the final output is stored in the data memory location "1" and can be displayed with the help of 10 LEDs.

IV. RESULTS

The simulation has been performed on Xilinx Vivado tool. Fig. 6 shows the simulation results when the instruction memory is loaded with the basic instructions like:

- lw x3,4(x0) = 32'h00402183
- lw x5,5(x0) = 32'h00502283
- addi x2, x0,2 = 32'h00200113

Initially register x7 is set to 8 and the rest of registers are 0. The instruction at the last stage (instr_s5) contains 32'h00402183 (lw x3,4(x0)) so accordingly at the next clock edge the register x3 is loaded with the data present at the data memory location '4'. Similarly at the next clock edge the register x5 is loaded with data at location '5' and then there is an addi instruction in which the register x2 stores the result of 2 added to x0.
The instructions are passed through the 5 stages as shown in Fig. 6 and the flow of data from the instruction fetch stage to the write back stage.

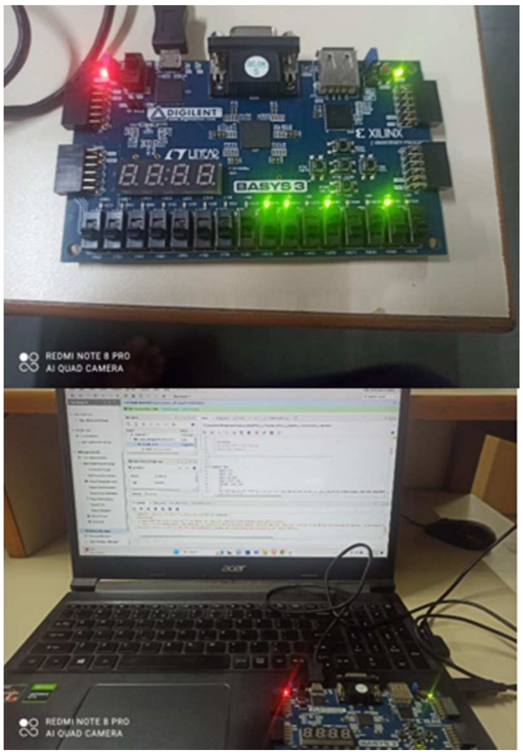


Figure 7. FPGA Output for Sum (=210) of N numbers (N=20)

FPGA Implementation of 5-Stage Pipelined RISC-V processor is shown in Fig. 7. The FPGA Board used was Basys-3 and the input (N=20) was given which generated the output sum equal to 210.

The single cycle architecture is tested for individual instructions in simulations after which it is tested for simple programs like finding the sum of first N natural numbers or finding the Nth Fibonacci term. The value of N is given as an input by the user. The design is dumped on the Basys-3 FPGA board. Fig. 8 shows the power and timing implementation results. The obtained power consumption is 0.244W for 25MHz clock the design produced the worst negative slack of 8.341ns means that the maximum operating frequency can be 31.6MHz. Fig. 9 shows the design utilization of the single cycle architecture.

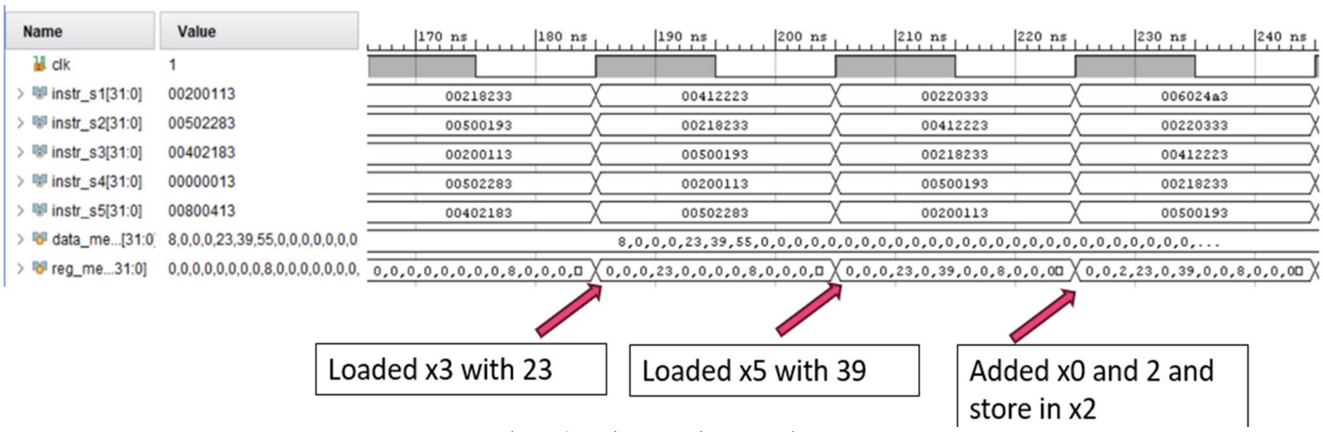
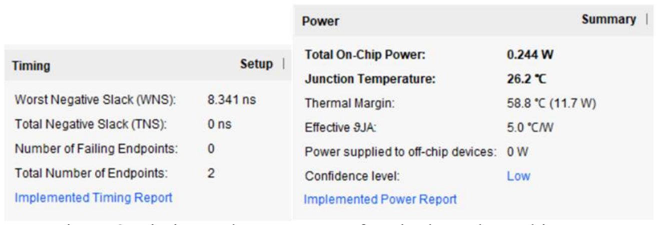


Figure 6. Basic Instruction Execution



Timing | Setup

Worst Negative Slack (WNS):	8.341 ns
Total Negative Slack (TNS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	2

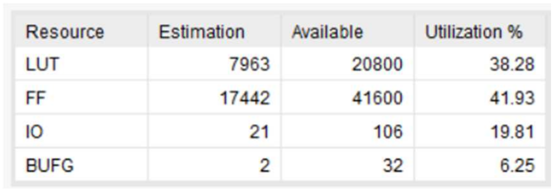
Implemented Timing Report

Power | Summary

Total On-Chip Power:	**0.244 W**
Junction Temperature:	**26.2 ℃**
Thermal Margin:	58.8 ℃ (11.7 W)
Effective ϑJA:	5.0 ℃/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

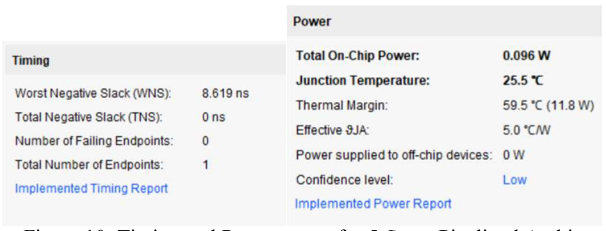
Figure 8. Timing and Power report for Single cycle Architecture



Resource	Estimation	Available	Utilization %
LUT	7963	20800	38.28
FF	17442	41600	41.93
IO	21	106	19.81
BUFG	2	32	6.25

Figure 9. Design Utilization for Single Cycle Architecture

The 5-Stage Pipelined architecture is also tested for the same instructions and programs like finding the sum of first N natural numbers or finding the Nth Fibonacci term. Fig. 10 shows the power and timing implementation results. After implementation on the Basys-3 board the power consumption is obtained as 0.096W and with a 50MHz clock frequency and achieved worst negative slack is 8.619ns indicating that the maximum operating frequency can be 87.86MHz. Fig. 11 shows the design utilization of the pipelined architecture.
It can be inferred from the utilization of both the design that number of LUT, LUTRAM and Flip flops have increased, and this was expected.



Timing

Worst Negative Slack (WNS):	8.619 ns
Total Negative Slack (TNS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	1

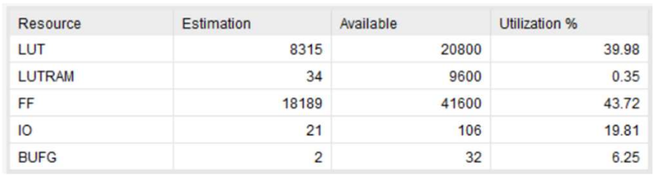
Implemented Timing Report

Power

Total On-Chip Power:	**0.096 W**
Junction Temperature:	**25.5 ℃**
Thermal Margin:	59.5 ℃ (11.8 W)
Effective ϑJA:	5.0 ℃/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

Figure 10. Timing and Power report for 5-Stage Pipelined Architecture



Resource	Estimation	Available	Utilization %
LUT	8315	20800	39.98
LUTRAM	34	9600	0.35
FF	18189	41600	43.72
IO	21	106	19.81
BUFG	2	32	6.25

Figure 11. Design Utilization for Pipelined Architecture

The 5-Stage Pipelined design shows a decent decrease in power dissipation and maximum clock frequency for 5-Stage processor coming out as 2.78 times the single cycle processor.

V. CONCLUSION

The biggest advantage of RISC-V processor is its simple design and open-source features. Compared to the modern microprocessors which have complex ISA and are IP-protected, which makes it very difficult to study. RISC-V processors are easily available for design and prototyping. The proposed design contains RAM of size 16KB (512*32) which is data memory and ROM of size 64KB (2048*32) which is the Instruction memory. The register file contains the 32 register of width 32 bits. For single cycle architecture a WNS of 8.3ns at a clock frequency of 25MHz is obtained by dividing the clock frequency of the Basys-3 board by 4. Hence a maximum operating frequency of 31.6MHz is achieved for the single cycle processor. The total power consumption of the single cycle design on the targeted board is 0.244W. For pipelined structure the worst negative slack is 8.619ns at 50MHz obtained by dividing the clock frequency of the targeted board by 2. Hence the maximum operating frequency obtained is 87.86MHz with a power consumption of 0.096W. The FPGA prototype is verified by loading the instruction memory with programs like Sum of N numbers and finding the Nth Fibonacci term in hexadecimal format.

VI. FUTURE WORKS

The next step is to optimize the HDL code further with the aim of reducing resource utilization. After that the aim should be to add Cache memory to increase the memory speed. Bus controllers along with buses connecting other peripherals could be a great add-on in the near future. Designing custom processors, high performing computers along with research and academic purposes are the biggest uses cases.

VII. REFERENCES

[1] Andrew Waterman, Krste Asanovi, and Five Inc. “The RISC-V Instruction Set Manual Volume I: User Level ISA Document Version 2.” CS Division, EECS Department, University of California, Berkeley 2017.

[2] David A. Patterson and John L. Hennessy “Computer Organization and Design RISC-V Edition: The Hardware Software Interface.” Morgan Kaufmann Publishers Inc., 2017, San Francisco, CA, USA.

[3] Ludovico Poli, Sangeet Saha, Xiaojun Zhai and Klaus D. Mcdonald-Maier, "Design and Implementation of a RISC V Processor on FPGA" 17th International Conference on Mobility, Sensing and Networking (MSN)

[4] Don Kurian Dennis, Ayushi Priyam, Sukhpreet Singh Virk, Sajal Agrawal, Tanuj Sharma, Arijit Mondal and Kailash Chandra Ray, "Single Cycle RISC-V Micro Architecture Processor and its FPGA Prototype", Seventh International Symposium on Embedded Computing and System Design (ISED), Durgapur, India, 2017, pp. 1-5, doi: 10.1109/ISED.2017.8303926.

[5] Wendi Zhang, Yonghui Zhang and Kun Zhao, “Design and Verification of Three-stage Pipeline CPU Based on RISC-V Architecture” 5th Asian Conference on Artificial Intelligence Technology (ACAIT) 2021.

[6] P. Singh, A. Rai, A. Rajput, P. C. Joshi and A. Prakash, "Design and Analysis of High Speed RISC Processor Using Pipelining Technique," 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 1642-1645, doi: 10.1109/ICAC3N56670.2022.10074423.

[7] J. Jeemon, "Pipelined 8-bit RISC processor design using Verilog HDL on FPGA," 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2016, pp. 2023-2027, doi: 10.1109/RTEICT.2016.7808194.

[8] S. Prabhakaran, M. N and V. Vedanarayanan, "Design and Analysis of a Multi Clocked Pipelined Processor Based on RISC-V," 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT), Chennai, India, 2022, pp. 1-5, doi: 10.1109/IC3IOT53935.2022.9767960.

[9] A. Tiwari, P. Guha, G. Trivedi, N. Gupta, N. Jayaraj and J. Pidanic, "IndiRA: Design and Implementation of a Pipelined RISC-V Processor," 2023 33rd International Conference Radioelektronika (RADIOELEKTRONIKA), Pardubice, Czech Republic, 2023, pp. 1-6, doi: 10.1109/RADIOELEKTRONIKA57919.2023.10109058.

[10] I. Thanga Dharsni, K. S. Pande and M. K. Panda, "Optimized Hazard Free Pipelined Architecture Block for RV32I RISC-V Processor," 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2022, pp. 739-746, doi: 10.1109/ICOSEC54921.2022.9952122.