DESIGN OF SINGLE CYCLE RISC-V PROCESSOR

Talluri Priya[1], Paruchuri Ushasri[2], Vetagiri Bhuvana Jyothirmai[3]

[1-3] *UG Students, Dept of Electronics and Communication Engineering, Vasireddy Venkatadri Institute of Technology, Nambur, Andhra Pradesh, India*

ABSTRACT

In the early the computer was stack architecture, later replaced by RISC architecture. The RISC-V Processor features a streamlined set of instructions, a consistent instruction length, an increased number of general-purpose registers, a load-store architecture, and simplified addressing modes. These characteristics contribute to faster execution of individual instructions, leading to improved overall performance and a simplified design. The choice of a RISC is easily understood. The processor is designed for targeting low-cost embedded devices. A Single-Cycle RISC-V Processor is a simplified, yet highly efficient microprocessor architecture that executes instructions in a single clock cycle. This design philosophy emphasizes minimalism, reduced complexity, and enhanced pipelining, which leads to faster execution times and a streamlined hardware structure. By implementing a RISC-V architecture, the processor benefits from an open standard, making it ideal for customization, expansion, and adaptation to various computing applications. The design of a Single-Cycle RISC-V Processor represents a significant leap in microprocessor architecture. Its minimalist design, open-source RISC-V instruction set, and single-cycle execution offer efficiency, scalability, and adaptability, making it an ideal choice for modern computing systems.

Keyword: - *RISC-V (Reduced instruction set computer 5th generation), Single cycle processor, Instruction Set Architecture (ISA), Data path, control unit*

---

1. INTRODUCTION

A Single-Cycle RISC-V processor is a type of microprocessor architecture designed based on the Reduced Instruction Set Computing (RISC) principles. RISC-V stands as an open-source instruction set architecture (ISA) recognized for its straightforwardness, adaptability, and scalability, which have contributed to its widespread adoption. The Single-Cycle design approach refers to the execution of each instruction in a single clock cycle, simplifying the pipeline stages and ensuring a uniform and predictable execution time for all instructions. RISC-V emerged as the solution to this challenge, offering a free and open Instruction Set Architecture (ISA) that is accessible for all to utilize. Single cycle processor refers to a computer processor architecture. In single cycle RISC V processor entire instruction executes in one clock cycle. It requires only state elements. Cycle time limited by longest instruction. It has key features Uniform execution time, Simplified control logic, Sequential Instruction Processing, High-Speed Execution for Simple Instructions and Efficiency Trade-offs.

2. EXISTING SYSTEM

In previous existing system we may observed that they use more number of LUTs, Adders, Multipliers and more number of components. This may cause longer compilation time, increasing of cost and Design complexity increases.

In our system we use less number of LUTs (2486) and less number of multipliers and Adders when compared to existing systems. This cause benefits of cost saving, easy to design system.

Table -1: Existing Model LUTs

Design Model	**LUTs**
Hyogeun et.al.,	6834
Don et.al.,	5578

3. PROPOSED SYSTEM

The suggested single cycle processor has many number of LUTs which produce complex design. In present system we use fewer LUT's than in other existing systems. Single cycle microarchitecture has two interactive components. Data path and control unit, Data path works on words of data control unit provides control signals to help data path to do right things at right time.

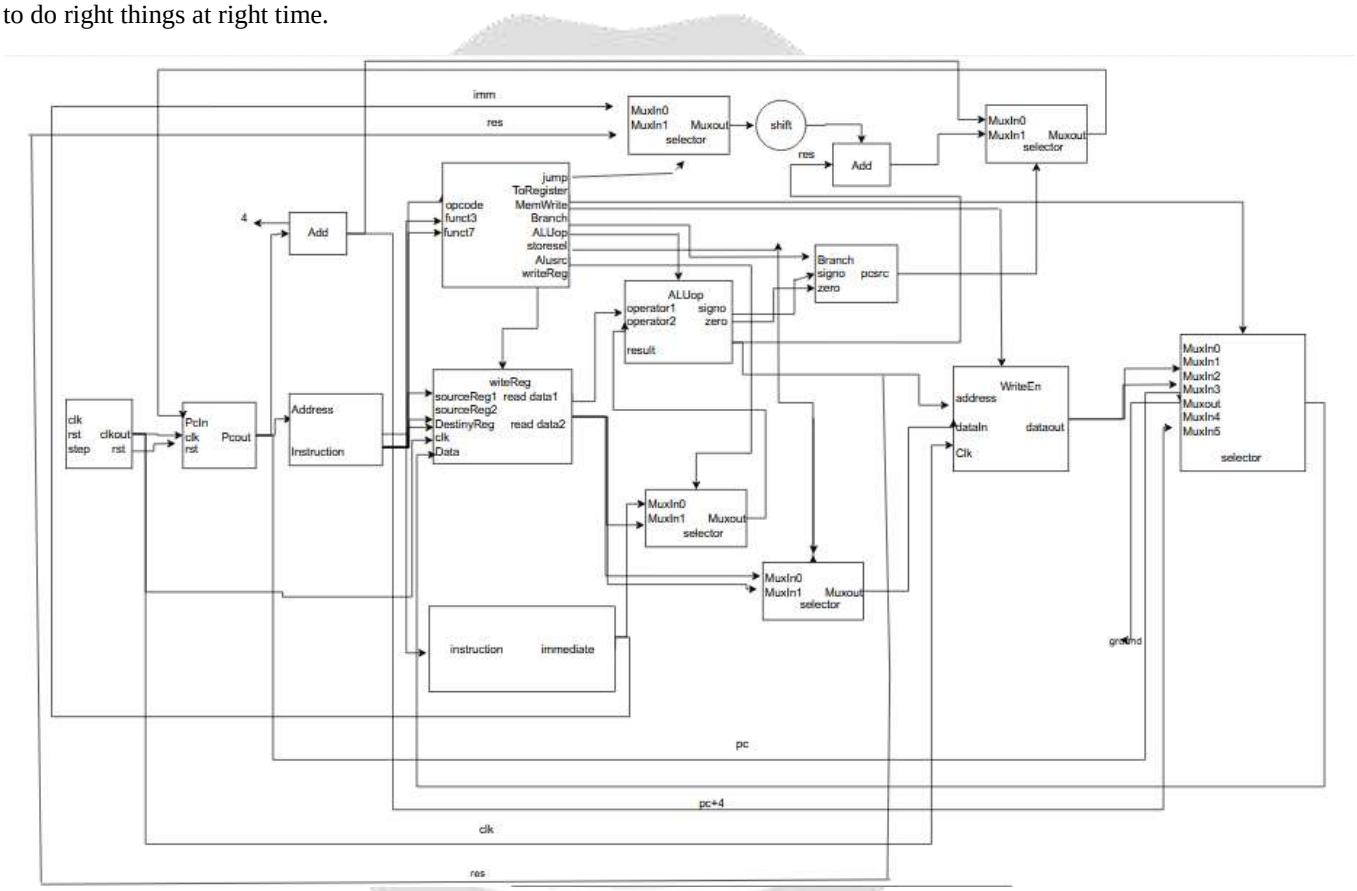


Fig-1: Block diagram of single cycle RISC-V processor

3.1 Functional blocks of Single cycle RISC-V Processor

The function of a rising edge detector is to identify the specific moment when the input signal undergoes a rising transition. The program counter (PC) points to the current instruction. Its input, PCIN, indicates the address of the next instruction. Instruction memory, often referred to as the instruction cache, is a component of the processor's memory hierarchy that stores the machine code instructions to be executed by the processor. The register file is a small, high-speed storage area within the processor that stores temporary data and operands. It consists of a set of registers, each capable of holding a fixed amount of data (typically 32). Data memory is only accessed by load and store instructions. ALU performs operations based on the instruction set we are given.

3.2 Control Unit

The control unit generates control signals based on opcode, Funct3, and Funct7 after reading instructions from instruction memory. The Control Unit have 8 output signals .to Register, it Selects the output for the Register File. Jump, it controls the multiplexer that allows the jump instruction. MemWrite, it allows to write in the Data Memory. Branch, it Determine which type Branch or Jump. ALUOp, it Determine the operation that must be performed by the ALU. StoreSel, it will Selects between the SB and SW data. ALUSrc, it will select the second operator for the ALU (register or immediate). WriteReg, it Allows writes in the Register File. Based on Funct 3, Funct 7, and ALUOP, ALU operates and generates the result.

Table-2: Control signals are generated based on Instruction format

Opcode	Jump	Branch	ToRegister	MemWrite	StoreSel	ALUSrc	WriteReg
R-type (0110011)	0	000	000	0	0	1	1
I-type (0010011)	0	000	000	0	0	0	1
I-store (0100011)	0	000	000	1	0-SW 1-SB	0	0
I-load (0000011)	0	000	001-LB 010-LW	0	0	0	1
Branch (1100011)	0	001-BEQ 010-BNQ 100-BLT 101-BGT	000	0	0	1	0
JALR (1100111)	1	101	011	0	0	1	0
JAL (1101111)	0	110	101	0	0	1	0

Table-3: Based on instructions the operations are followed in ALU

Instruction	Funct 3	Funct 7	ALUOP	Operation
R-Type	000	0000000	100	ADD
	000	0100000	101	SUB
	001		110	SLL
	010		011	SLT
	100		010	XOR

	101		111	SRL
	110		010	OR
	111		000	AND
I – Type Arith	000		100	ADDI
	111		000	ANDI
	100		010	XORI
	110		100	ORI
I – Type Load	000		100	LB
	010		100	LW
I – Type Store	000		100	SB
	010		100	SW
Branch	000		101	BEQ
	001		101	BNQ
	100		101	BLT
	101		101	BGT

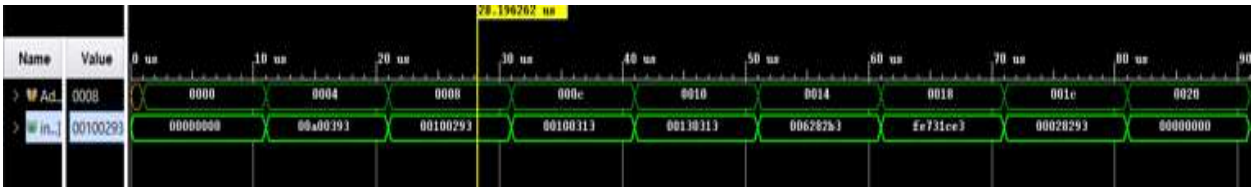
4. RESULTS

The proposed design is verified for a sample assembly code for a sum of first ten integer numbers. Instructions for which design has made, is used in the code. In this code registers x5 and x6 are used to carry the integers and intermediate results while x7 is used to hold the counter value in order to repeat the loop. The assembly language program and the binary format of the instructions are shown in Table 4.

Table-4: Sample code and it's binary format

code	Binary format
addi x7,x0,a	00000000101000000000000111001001 1
addi x5,x0,1	00000000000100000000001010010011
addi x6,x0,1	00000000000100000000001100010011
next:addi x6,x6,1	00000000000100110000001100010011
add x5,x5,x6	00000000011000101000001010110011
bne x6,x7,next	11111110011100110001110011100011
addi x5,x5,0	00000000000000101000001010010011

The proposed design with example on the sum of ten integers is simulated using Xilinx Vivado 2019.1 and observed the results. After adding first ten decimal numbers, the result should be 37H (55D). At the 10th iteration the obtained



result is produced and stored in x6 register. The simulation waveforms of instruction memory, and first and last iterations are shown in the following figures. The number of LUT's are also 50 percent lesser than the existing

system this result in system easy to design and less complex when compared to existing system.

4.1 Code in Instruction Memory (Binary format):

```
constant ROM: ROM_ARRAY := (
"00000000","00000000","00000000","00000000",
 "00000000","10100000","00000011","10010011",   ---addi x7,x0,a
 "00000000","00010000","00000010","10010011",   ---addi x5,x0,1    04H,07H
 "00000000","00010000","00000011","00010011",  ---addi x6,x0,1     08H,0BH
 "00000000","00010011","00000011","00010011",  ---next:addi x6,x6,1
 "00000000","01100010","10000010","10110011",  ---add x5,x5,x6
 "11111110","01110011","00011100","11100011",  ---bne x6,x7,next
 "00000000","00000010","10000010","10010011",  ---addi x5,x5,0
others => X"00"
  );
```

Fig-2: Simulation of instruction memory

Fig-3: Simulation of Data path for 1st iteration

Fig-4: Simulation of Data path for 10th iteration

Table-5: Number of LUT's used

Elements	LUT'S Required
Rising edge detector	10
Program Counter	100
Instruction Memory	94
Register File	275
MUX 0	129
ALU	468
MUX 1	129
Data Memory	563
Branch control	7
Mux to register	299
Control	61
MUX 2	129

MUX 3	129
Immediate Generator	93

5. CONCLUSIONS

The project focuses on the design of single cycle RISC-V processor. A compact and high-speed system can be designed by using RISC V processor which can be used to develop a low-cost real-time systems with fewer number of LUT's.

6. REFERENCES

[1]. Digital design and computer Architecture RISC V Edition by Sarah L Harris and David money Harris
[2]. Single Cycle 32-bit RISC-V ISA Implementation and Verification by Hyogeun Et.al
[3]. Don KD, Ayushi P, Virk SS, Sajal A, Tanuj S, Arit M, Kailash CR, Single cycle RISC-V micro architecture processor and its FPGA prototype, Proceedings of 7th International Symposium on Embedded Computing and System Design. 2017 Dec. India.