RISC-V Single Cycle Processor Implementation in Verilog

Ghulam Ishaq Khan Institute

Muhammad Sufyan

Instructors: Dr Fahad Bin Muslim and Sir Abbas Khan

Abstract:

This project implements a 32-bit single-cycle RISC-V processor in Verilog HDL. The design executes fundamental R-type, I-type, and memory access instructions in a single clock cycle. The processor was verified through simulation in Xilinx Vivado, demonstrating correct execution of arithmetic, load/store, and branch instructions. The design showcases the integration of key datapath components including ALU, control unit, instruction memory, and register file.

Objectives:

- To design and implement a single-cycle RISC-V processor.
- To understand the internal datapath and control signal flow.
- To verify functional correctness using simulation waveforms and console output.
- To demonstrate execution of a basic instruction sequence including arithmetic and memory operations.

Processor Architecture:

This processor contains the following modules that are necessary to implement the instruction formats we plan to execute:

I. Program Counter: the program counter is responsible for sequentially executing the instructions of our program or (if need) jump to the specified address of the instruction memory that is in the case of branching.

II. Instruction Memory: it is a vital part of our processor that contain the set of instructions of our program which are accessed by their respective memory addresses.

III. Register File: the register file contains all the registers for or processor that when needed can be accessed by our program and used to store data.

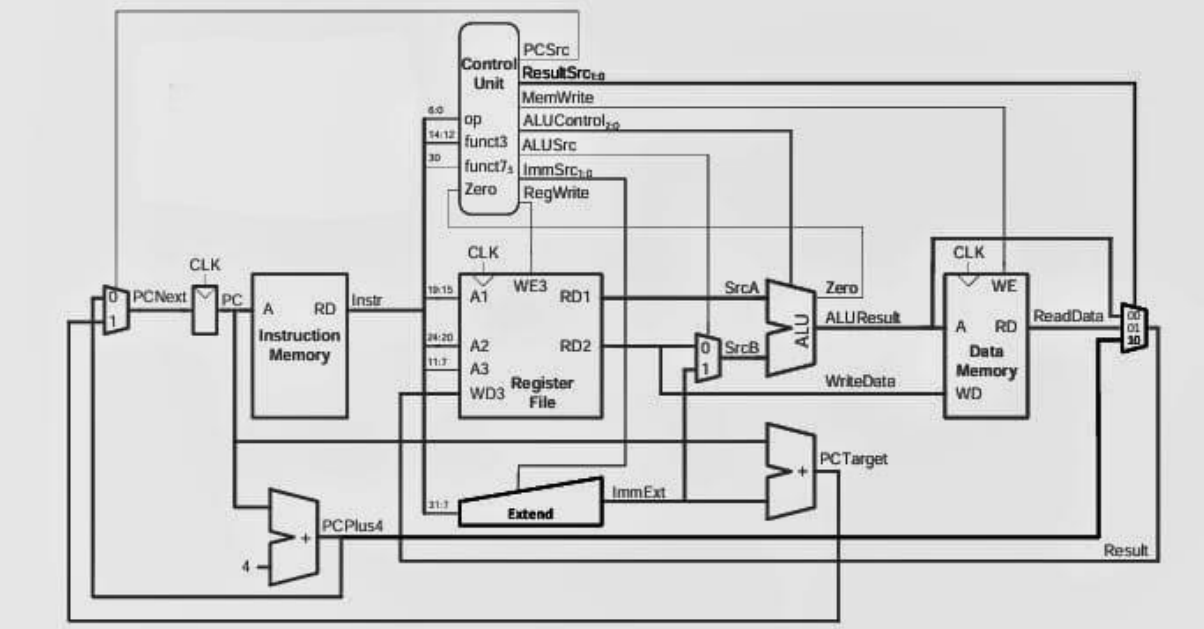
IV. Immediate generator: the immediate generator is used for extending the 12-bit immediate from our instruction to 32-bit format so it can be used to perform arithmetic/logical operations.

V. ALU: the ALU is an essential component of our CPU that performs the required arithmetic or logical operations based on our program's instructions.

VI. Data Memory: the data memory is used to store or retrieve our data when required.

VII. Control Unit: the control unit is responsible for sending the appropriate signals to the each component such as ALU, ALU-src ,PC-src, Result-src etc so the correct datapath is followed with respect to the instruction. The Control Unit sends these signals based on the Opcode of each instruction.

Other components such as the PC-Adder, PC-Target, or the Multiplexers are used to aid integrating the functionality of multiple datapaths.



Implemented ISA:

	31	30 25	24 21	20	19 15	14 12	11 8	7	6 0
R	funct7		rs2		rs1	funct3	rd		opcode
I	imm[11:0]				rs1	funct3	rd		opcode
S	imm[11:5]		rs2		rs1	funct3	imm[4:0]		opcode
B	imm[12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm[11]	opcode

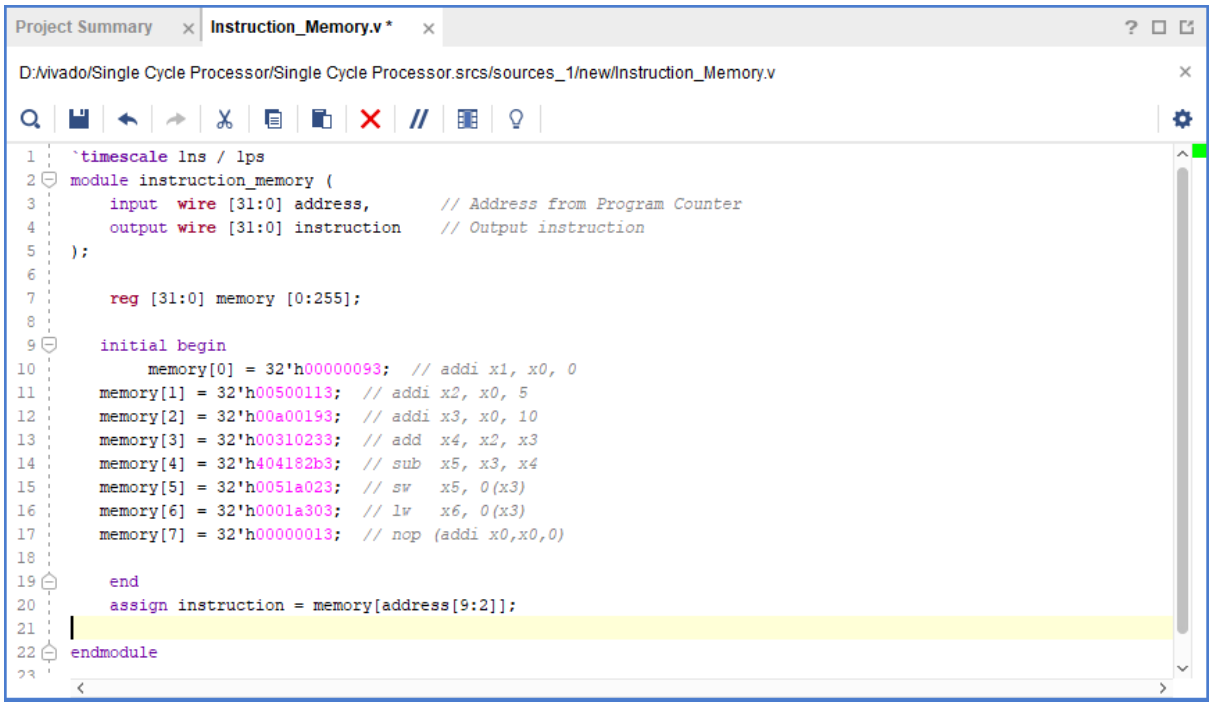
1. I-type: Performs arithmetic/logic or load operations using one register and an immediate value
2. S-type: Stores data from a register into memory.
3. R-type: Performs arithmetic/logic operations using **two registers**, result stored in another register.
4. B-type: Compares two registers and branches to a new address if the condition is true.

Simulation and Verification:

To correctly verify the complete correctness and working of our single cycle processor, we write the code for all the instruction format of our interest. Using Venus-Simulator online we write and generate the machine code to be hard-coded into our instruction memory.

PC	Machine Code	Basic Code	Original Code
0x0	0x00000093	addi x1 x0 0	addi x1, x0, 0
0x4	0x00500113	addi x2 x0 5	addi x2, x0, 5
0x8	0x00A00193	addi x3 x0 10	addi x3, x0, 10
0xc	0x00310233	add x4 x2 x3	add x4, x2, x3
0x10	0x404182B3	sub x5 x3 x4	sub x5, x3, x4
0x14	0x0051A023	sw x5 0(x3)	sw x5, 0(x3)
0x18	0x0001A303	lw x6 0(x3)	lw x6, 0(x3)

The highlighted portion in the above figure shows the 32-bit instructions of our RISC-V program in hexadecimal. These machine codes are hard-coded into the instruction memory module of our RISC-V processor as shown in the Verilog code below:



When the following instructions are executed on our processor, we expect the following output at our modules:

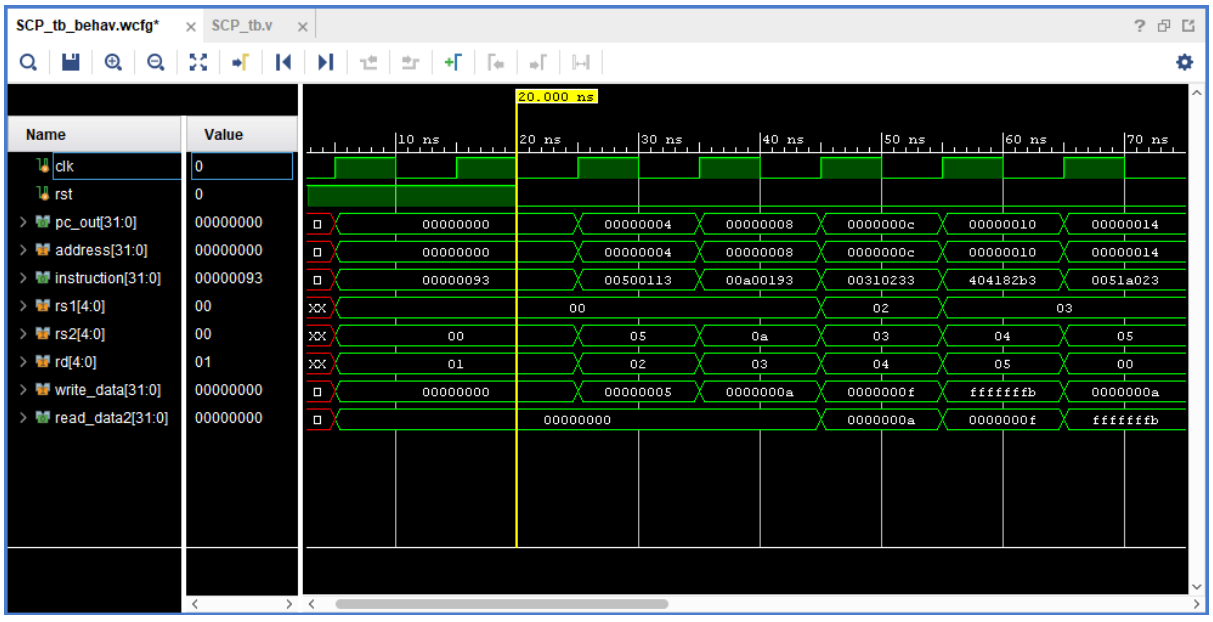
PC	Instruction	Action / Meaning	Register Result	ALU Result (Hex)
0x00000000	addi x1, x0, 0	x1 = 0	x1 = 0	0x00000000
0x00000004	addi x2, x0, 5	x2 = 5	x2 = 5	0x00000005
0x00000008	addi x3, x0, 10	x3 = 10	x3 = 10	0x0000000A
0x0000000C	add x4, x2, x3	x4 = 15	x4 = 15	0x0000000F
0x00000010	sub x5, x3, x4	x5 = -5	x5 = -5	0xFFFFFFFB
0x00000014	sw x5, 0(x3)	MEM[x3] = x5	(no register change)	0x0000000A
0x00000018	lw x6, 0(x3)	x6 = -5	x6 = -5	0xFFFFFFFB
0x0000001C	nop	No change	No register change	0x00000000

Using the $monitor command in our testbench we can generate the values at each module's input/output for all the instructions that are executed. Using these values, we verify the correctness of execution by comparing it with the above truth table.

Tcl Console | Messages | Log

```
# run 1000ns
Time      PC   RS1 RS2 RD   ALU_Result
0    xxxxxxxx     x   x    x    xxxxxxxx
5000      00000000     0   0    1    00000000
25000     00000004     0   5    2    00000005
35000     00000008     0   10   3    0000000a
45000     0000000c     2   3    4    0000000f
55000     00000010     3   4    5    fffffffb
65000     00000014     3   5    0    0000000a
75000     00000018     3   0    6    0000000a
85000     0000001c     0   0    0    00000000
95000     00000020     x   x    x    xxxxxxxx
105000    00000024     x   x    x    xxxxxxxx
115000    00000028     x   x    x    xxxxxxxx
125000    0000002c     x   x    x    xxxxxxxx
```

From the above-mentioned output, we have successfully verified the correctness of the execution of our Single Cycle Processor. This output can also be verified using the simulation waveforms as shown below:



Conclusion

The designed RISC-V single-cycle processor successfully demonstrates the fundamental operation of a modern CPU architecture. It executes instructions in one clock cycle, correctly updating registers and memory. The project strengthened understanding of instruction flow, control logic, and hardware design using Verilog HDL.