RISC-V ISA CPU

Objectives

The goal of this project is to create a full 32 bit CPU which follows the RISC-V ISA. It is meant to created from the ground up using Verilog HDL in Quartus Prime and simulated using Model-Sim Altera. This processor should perform R, I, S, and B type instructions from the RISC-V ISA, including load and store operations by using a built in data memory (RAM).

At the conclusion of this project, the CPU should be supported by additional hardware allowing it to be loaded onto a DE-10 Lite FPGA board using a switch for reset and pushbutton to control the clock. To display instruction outputs, four seven-segment displays should be utilized to show the output in binary coded decimal (BCD) form.

Methodology

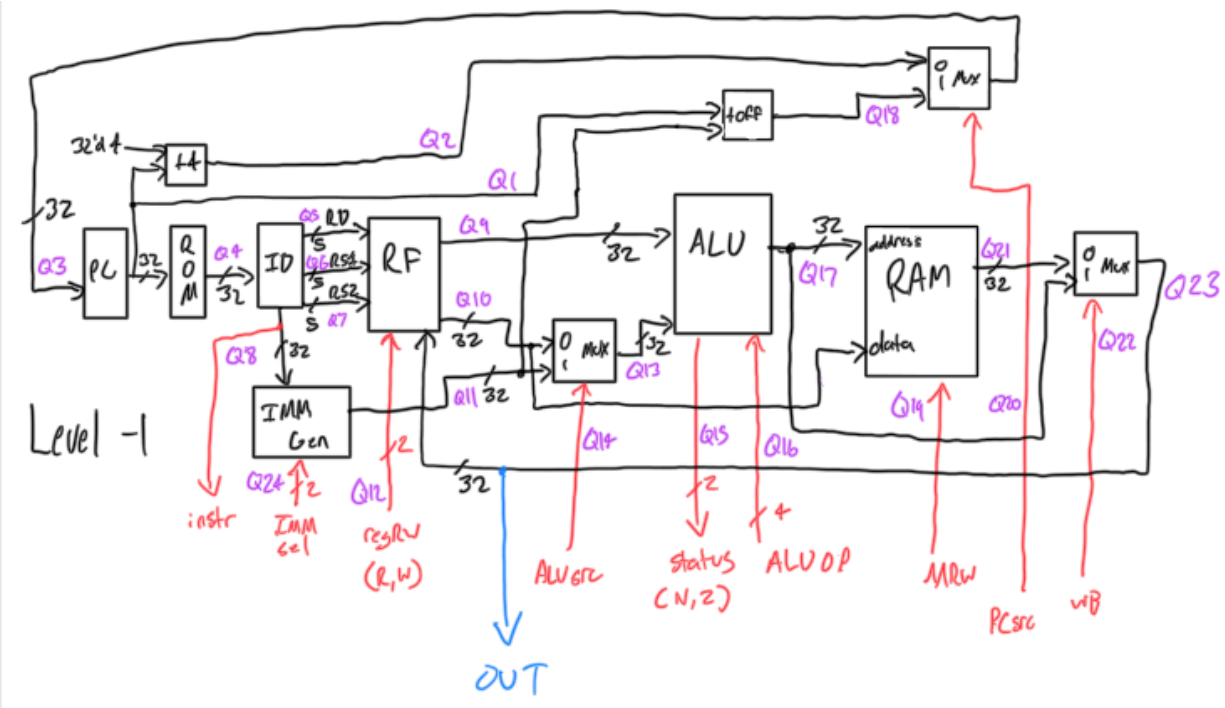
Top Level Ports

My RISC-V CPU has a clock and reset input. When the reset input goes high, all memory is cleared and the program counter is reset. Each time the clock signal goes high, the program counter is incremented and the next instruction in the ROM is decoded and executed.

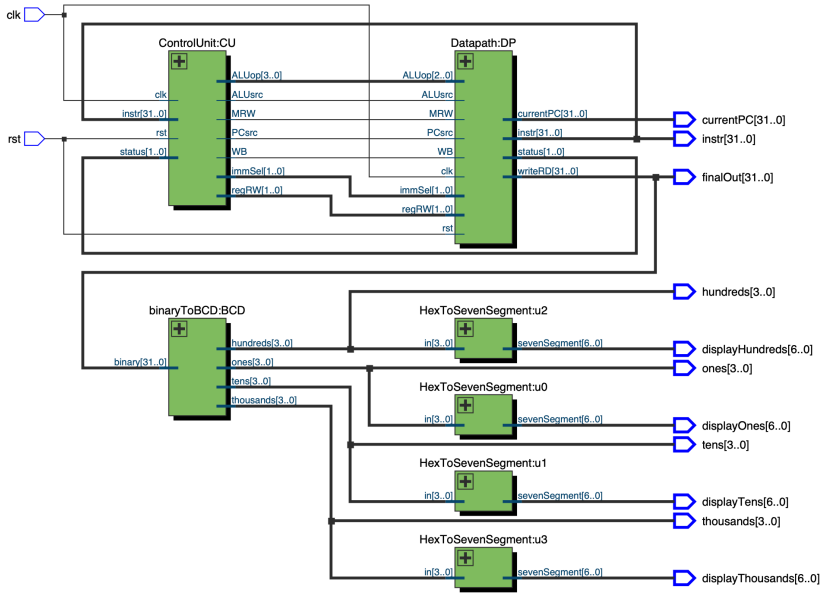
The CPU has a single 32 bit output which contains the results of each instructions execution.

Components

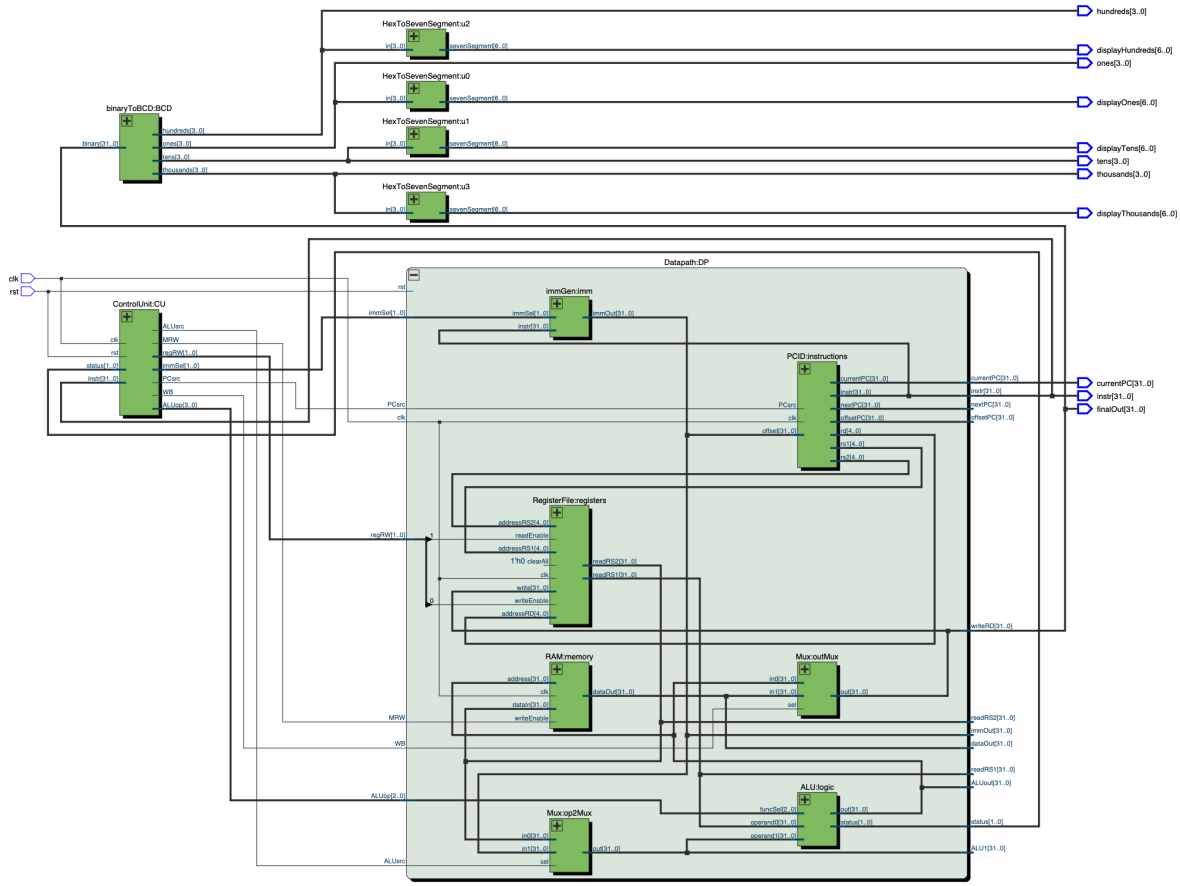
To make the CPU, the datapath I previously created in lab 8 is connected to the control unit I designed in lab 9. The datapath has the following level -1 block diagram:



CPU Netlist



CPU Netlist (expanded Datapath)



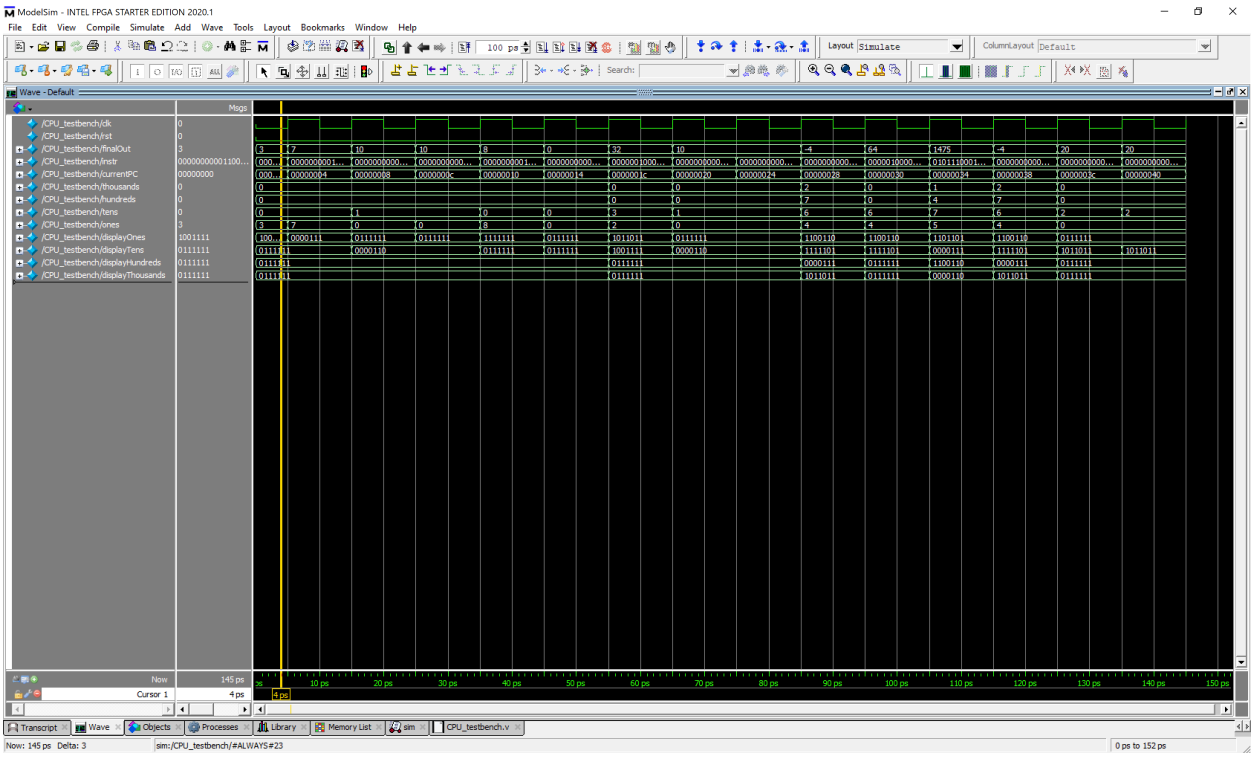
Results and Analysis

For testing, the following set of instructions were saved in the CPU instruction memory (ROM) and a simulation was run in Model-Sim Altera.

RISC-V ISA ROM Instructions for CPU Testing

Program Counter	Assembly Code	Instruction Type	Machine Code (b)	Equivalent Statement	Expected Output	Simulated Output	Executed
0x00	addi x1, x0, 3	I	000000000011_00000_000_00001_0010011	x1 = 0 + 3	3	3	y
0x04	addi x2, x0, 7	I	000000000111_00000_000_00010_0010011	x2 = 0 + 7	7	7	y
0x08	add x3, x2, x1	R	0000000_00001_00010_000_00011_0110011	x3 = 7 + 3	10	10	y
0x0C	add x4, x3, x0	R	0000000_00000_00011_000_00100_0110011	x4 = 10 + 0	10	10	y
0x10	sw x4, 1(x2)	S	0000000_00100_00010_010_00001_0100011	RAM[1+7] = 10	8	8	y
0x14	beq x4, x3, 8	B	0000000_00011_00100_000_01000_1100011	if(10 == 10) PC += 8	0	0	y
0x18	addi x7, x0, 16	A	0000000010000_00000_000_00111_0010011	x7 = 0 + 16	16	16	n
0x1C	addi x8, x0, 32	A	0000001000000_00000_000_01000_0010011	x8 = 0 + 32	32	32	y
0x20	lw x9, 1(x2)	I (lw)	000000000001_00010_010_01001_0000011	x9 = RAM[1+7]	10	10	y
0x24	add x10, x9, x0	R	0000000_00000_01001_000_01010_0110011	x10 = 10 + 0	10	10	y
0x28	blt x1, x2, 8	B	0000000_00010_00001_100_01000_1100011	if(3 < 7) PC += 8	-4	-4	y
0x2C	addi x7, x0, 16	I	0000000010000_00000_000_00111_0010011	x7 = 0 + 16	16	16	n
0x30	addi x11, x0, 64	I	0000010000000_00000_000_01011_0010011	x11 = 0 + 64	64	64	y
0x34	xori x8, x2, 1476	R (logical)	010111000100_00010_100_01000_0010011	x8 = 7 ^(xor) 1476	1475	1475	y
0x38	bge x1, x2, 8	B	0000000_00010_00001_101_01000_1100011	if(3 >= 7) PC += 8	-4	-4	y
0x3C	add x3, x3, x3	R	0000000_00011_00011_000_00011_0110011	x3 = 10 + 10	20	20	y
0x40	add x16, x3, x0	R	0000000_00000_00011_000_01111_0110011	x16 = 20 + 0	20	20	Y

In the following simulation, all outputs function exactly as expected. Each finalOut value from the CPU matches the expected value from the instructions, and the program counter increments correctly including for both true and false branch instructions. Additionally, it can be seen that the BCD conversion is accurate and the seven-segment decoding of these BCD values is correct and would be displayed properly on the DE-10 Lite FPGA board.



Discussion and Conclusion

After evaluating test results for a comprehensive set of testing instructions, it is clear that this CPU is functional for R, I, S, and B type instructions from the RISC-V ISA, including load and store operations by using a built in data memory (RAM).

The CPU is also set up in a way which can be loaded onto a DE-10 Lite board, using a switch for reset and pushbutton for the clock, to display the instruction outputs as binary coded decimal (BCD) on the board's four seven-segment displays. Unfortunately, because my CPU was created in a virtual machine on a Mac, I was never able to get the USB Blaster II software installed and able to properly interface with my DE-10 Lite board through the virtual machine to hardware. However, my simulations prove that this project would work as expected if loaded onto a board from another sdevice.