Final Project implementation of 16-bit and 32-bit processors in Logisim and Vivado

Hutton Amison-Addy Victor Quagraine Delali Nsiah-Asare

Ashesi University

CS331_A: Computer Organization and Architecture

Dr. Robert A. Sowah

05/08/2024

Part A: Logisim implementation of 16-bit Processor

Overview

The project required the development of a 16-bit CPU in Logisim. The circuit involved eight and 16-bit registers for Instruction, Output, A program and Step counter, Memory address and bus, as well as 2 data registers, an ALU circuit, a RAM to load in instructions for use in the CPU and a decoder for displaying onto a 7-segment display output.

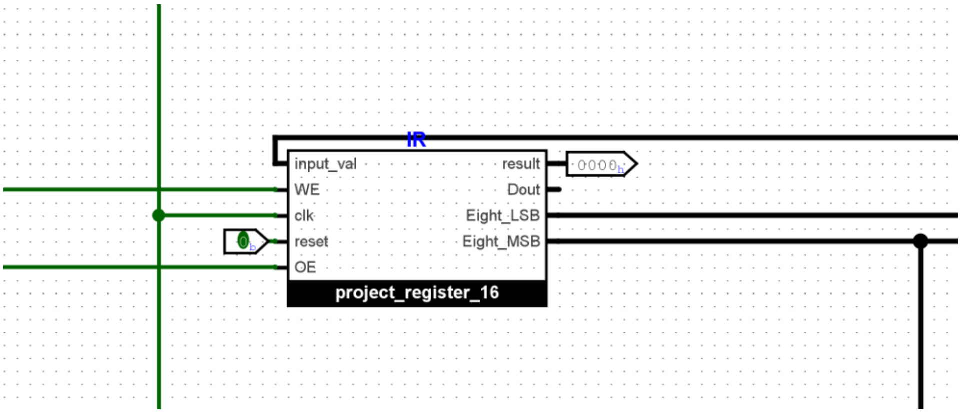
Challenges

The primary challenge faced involved debugging the circuit due to the number of components used.

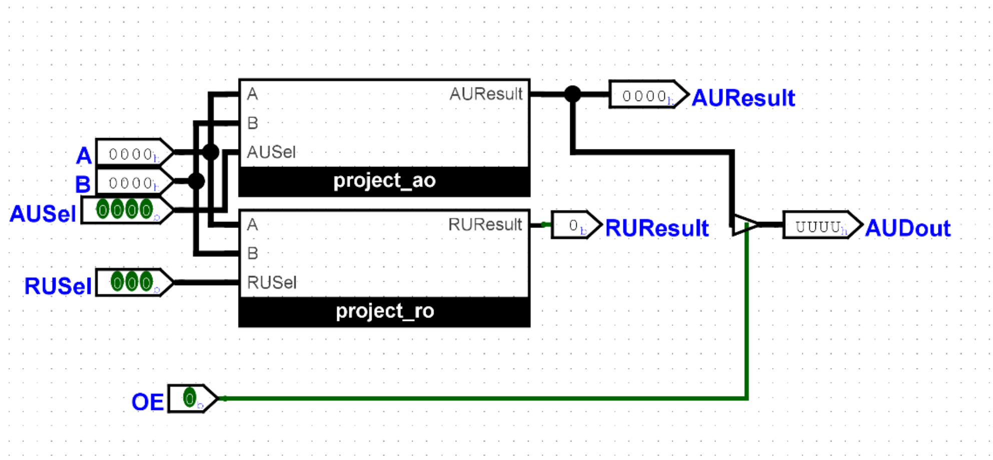
Implementation

1. CPU Design Architecture

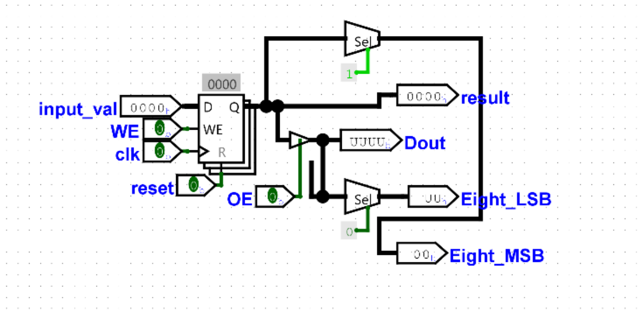
Instruction Register (IR) 16-bit



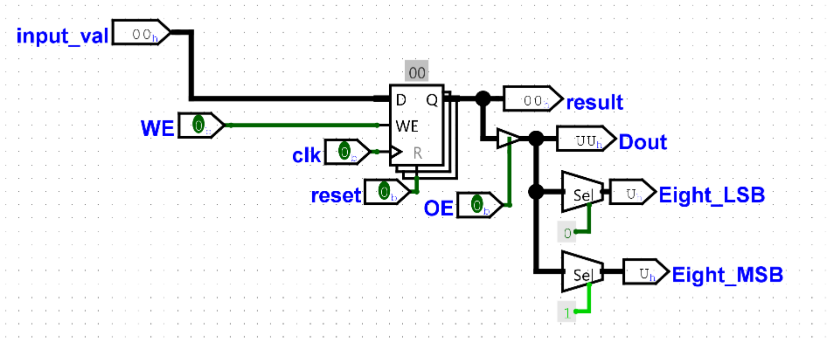
Arithmetic Logic Unit (ALU) 16 bits



16-bit-Register (Output Register, RegA, RegB, MBR)



8-bit-Register (MAR)



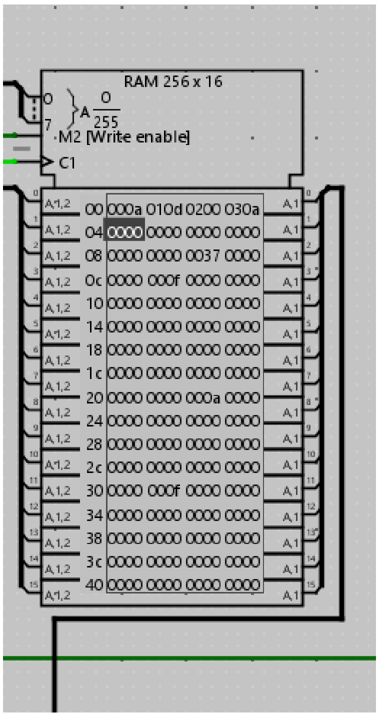
2. Datapath Design

a. Instruction implementation loaded into the RAM

Memory Address	**Content**	**Hex**	RAM Instruction form
0000	**00000000 00001010**	0x00 0x0A	000a
0001	00000001 00001101	0x01 0x0D	010d
0010	00000010 00000000	0x02 0x00	0200
0011	00000011 00001010	0x03 0x0A	030a
0100	00000000 00000000	0x00 0x00	
0101	00000000 00000000	0x00 0x00	
0110	00000000 00000000	0x00 0x00	
0111	00000000 00000000	0x00 0x00	
1000	00000000	0x00	
1001	00000000	0x00	

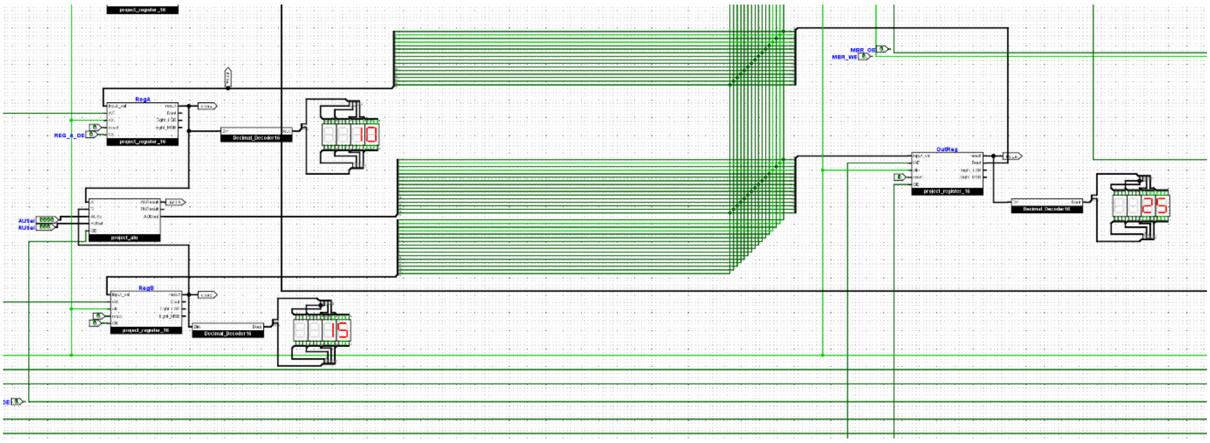
1010	00001010	0x0A	000a
1011	00000000	0x00	
1100	00000000	0x00	
1101	00001111	0x0F	000f
1110	00000000	0x00	
1111	00000000	0x00	

3. Simulation and Testing



```
00 000a 010d 0200 030a
04 0000 0000 0000 0000
08 0000 0000 0037 0000
0c 0000 000f 0000 0000
10 0000 0000 0000 0000
14 0000 0000 0000 0000
18 0000 0000 0000 0000
1c 0000 0000 0000 0000
20 0000 0000 000a 0000
24 0000 0000 0000 0000
28 0000 0000 0000 0000
2c 0000 0000 0000 0000
30 0000 000f 0000 0000
34 0000 0000 0000 0000
38 0000 0000 0000 0000
3c 0000 0000 0000 0000
40 0000 0000 0000 0000
```

Figure 1: RAM after 3 iterations of the instructions provided


project_register_16
RegA
Input_val
WE
clk
reset
OE
result
Dout
Eight_LSB
Eight_MSB
project_register_16
REG_A_OE
Din
Dout
Decimal_Decoder16
10
A
B
AUSel
RUSel
OE
AUResult
RUResult
AUDout
project_alu
AUSel
RUSel
RegB
Input_val
WE
clk
reset
OE
result
Dout
Eight_LSB
Eight_MSB
project_register_16
Din
Dout
Decimal_Decoder16
15
OE
MBR_OE
MBR_WE
OutReg
Input_val
WE
clk
reset
OE
result
Dout
Eight_LSB
Eight_MSB
project_register_16
Din
Dout
Decimal_Decoder16
25

Part 2: VHDL Implementation of a 32-bit Processors

Overview

The Project expands on the previous part, building a 32-bit processor with implementation in VHDL. We used Vivado to simulate the output through a testbench.

Challenges

Everything. The implementation of branch instructions caused an error with ALU outputs. We were unsure about the assignment of opcodes for the ALUop, ALU operations, in the control unit file.

Result of Simulation

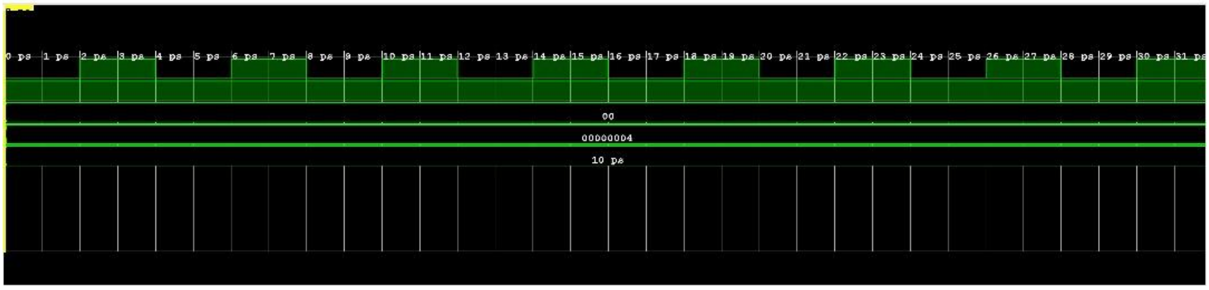


Figure 2: Instruction 0

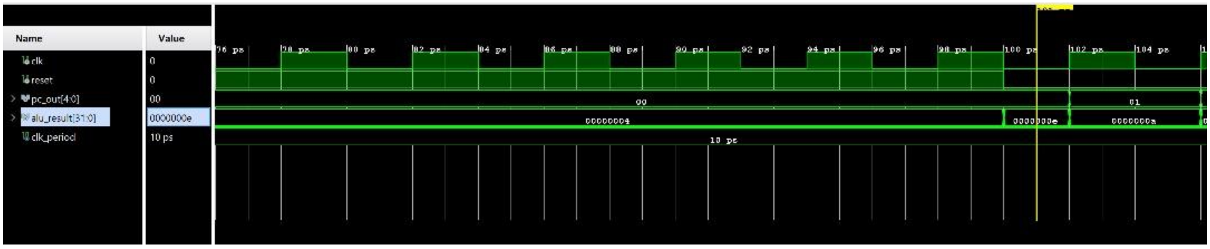


Figure 3: Instruction 1

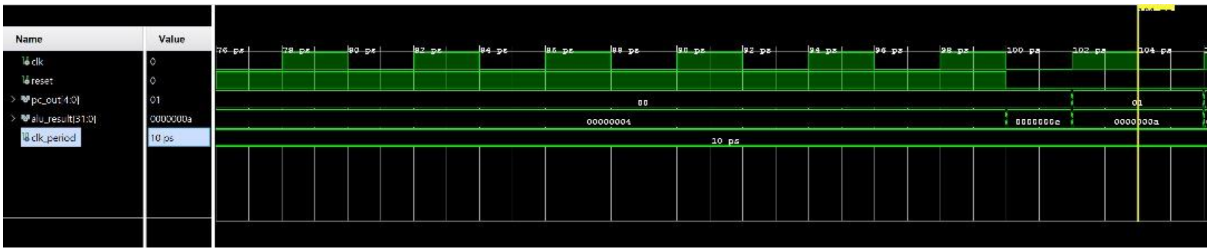
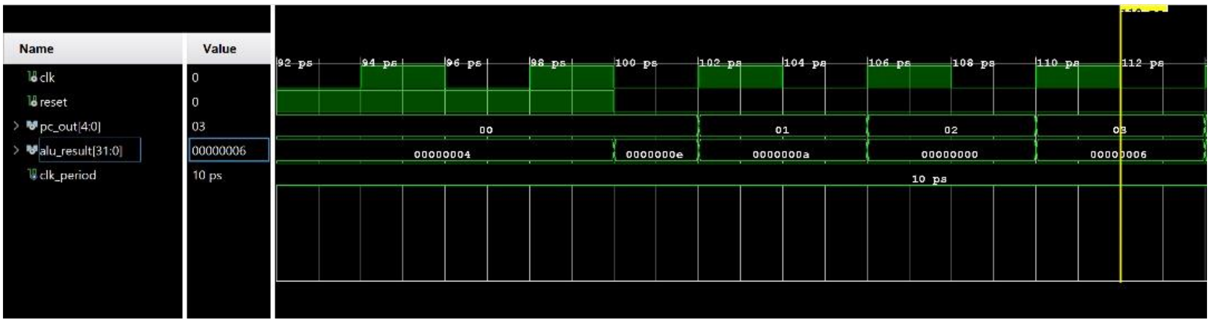
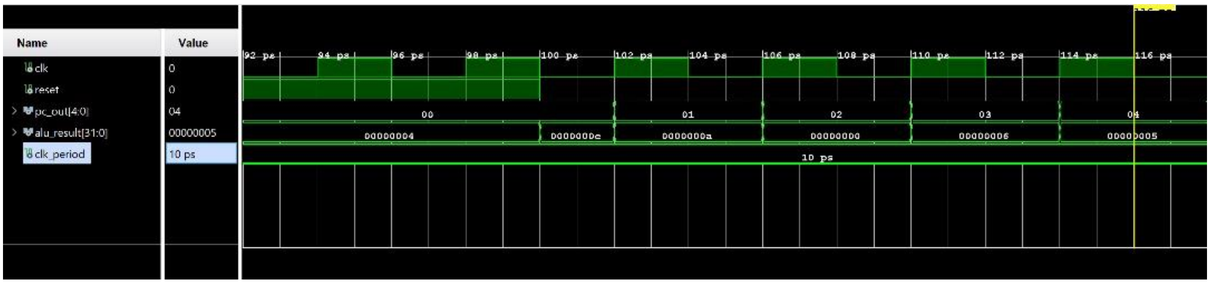


Figure 4: Instruction 2



Name	Value
clk	0
reset	0
pc_out[4:0]	03
alu_result[31:0]	00000006
clk_period	10 ps

Figure 5: Instruction 3



Name	Value
clk	0
reset	0
pc_out[4:0]	04
alu_result[31:0]	00000005
clk_period	10 ps

Figure 6: Instruction 4