

---

## **REQUIREMENTS NOT MET**

---

N/A

---

## **VIDEO FILE LINK**

---

<https://drive.google.com/file/d/1hkCioxShRF72u8v41gJvQaLox0cByYvh/view?usp=drivesdk>

---

## **PROBLEMS ENCOUNTERED**

---

There were no problems encountered.

---

## **FUTURE WORK/APPLICATIONS**

---

This Lab 6 focuses on building an elementary CPU based on the RALU from Lab 4, with the addition of an Instruction Register (IR), Program Counter (PC), and a ROM to store the data and instructions. Using memory in this way (ROM) to perform simple arithmetic calculations represents the building blocks of simple computer logic, and can be further expanded to more complex functions and tasks.

---

## **PRE-LAB QUESTIONS OR EXERCISES**

---

### **PART 1**

1. Why did we require the new instruction register in this design?
  - a. We needed to add a new instruction register in this design to automate the op codes that go into the mux select outputs.
2. In this section of the lab, you are setting the INPUT bus by hand. If you wanted to read or fetch this value from memory, what could you add to do this automatically for you every CLK cycle?
  - a. If you wanted to read this value from memory, you could add a ROM to automatically store the value every CLK cycle.
3. How would you add more instructions (i.e., 8 instead of 4) to the controller?
  - a. To add more instructions to the controller I would increase the number of bits to the IR from two to three.

### **PART 2**

1. Why do we need the extra states in the LDAA and JMP instruction paths?
  - a. We need the extra states in the LDAA and JMP instruction paths in order to take in the address value and then set it to the PC.
2. What do you need to do to the address lines to get your program to start at address \$37D0 (instead of \$2B70)?
  - a. You need to directly connect the A[14..4] wires to the hexadecimal representation of \$2B70.

PRE-LAB REQUIREMENTS (Design, Schematic, ASM Chart, VHDL, etc.)

PART 1: FIRST RALU CONTROLLER

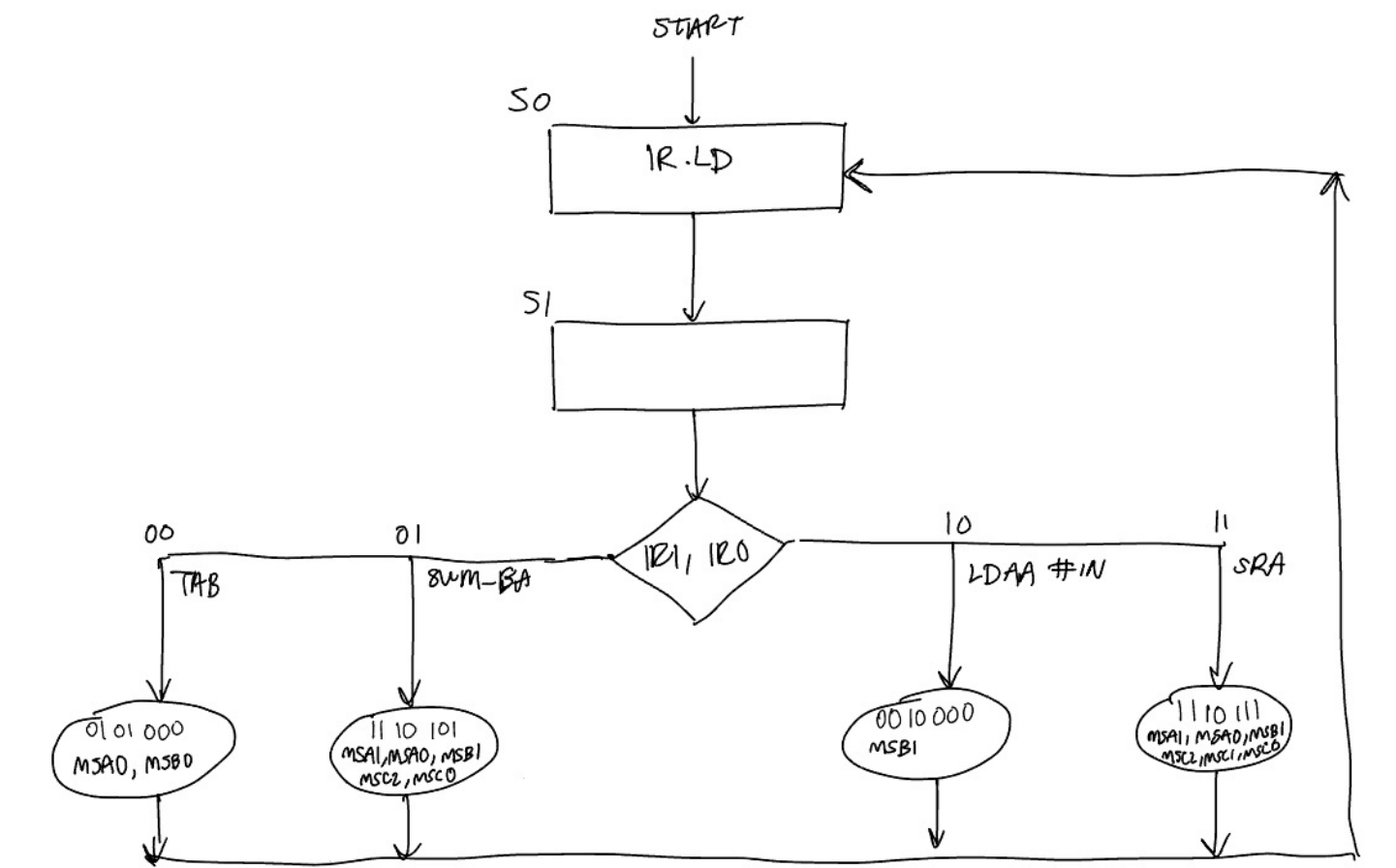


Figure 1: Controller ASM chart for Part 1

			D0								D0
		IR.LD	IR.LD+								
IR1	IR0	Q0	Q0+	MSA1	MSA0	MSB1	MSB0	MSC2	MSC1	MSC0	IR.LD
X	X	0	1	0	1	1	0	0	0	0	1
0	0	1	0	0	1	0	1	0	0	0	0
0	1	1	0	1	1	1	0	1	0	1	0
1	0	1	0	0	0	1	0	0	0	0	0
1	1	1	0	1	1	1	0	1	1	1	0

Figure 2: Next State Truth Table for Part 1

$$\begin{cases} IR.LD = \overline{Q0} \\ Q^*/D = \overline{Q0} \end{cases}$$

$$MSA1 = \overline{IR[1]} IR[0] Q0 + IR[1] \overline{IR[0]} Q0$$

$$MSA0 = \overline{IR[1]} \overline{IR[0]} Q0 + \overline{IR[1]} IR[0] Q0 + IR[1] \overline{IR[0]} Q0$$

$$MSB1 = \overline{IR[1]} IR[0] Q0 + IR[1] \overline{IR[0]} Q0 + IR[1] IR[0] Q0$$

$$MSB0 = \overline{IR[1]} \overline{IR[0]} Q0$$

$$MSC2 = \overline{IR[1]} IR[0] Q0 + IR[1] IR[0] Q0$$

$$MSC1 = IR[1] IR[0] Q0$$

$$MSC0 = \overline{IR[1]} IR[0] Q0 + IR[1] IR[0] Q0$$

Figure 3: Logic Equations for Part 1

```

1  library ieee; use ieee.std_logic_1164.all;
2
3  entity L6P1Controller is port(
4      --IRLD : out std_logic;
5      D0 : out std_logic;
6      Q0 : in std_logic;
7      IR : in std_logic_vector (1 downto 0);
8      MSA : out std_logic_vector (1 downto 0);
9      MSB : out std_logic_vector (1 downto 0);
10     MSC : out std_logic_vector (2 downto 0));
11
12  end L6P1Controller;
13
14  architecture behavior of L6P1Controller is
15      --signal CW, EV_L : std_logic;
16      begin
17
18          D0 <= not(Q0);
19
20          MSA(1) <= ( not(IR(1)) and (IR(0)) ) or
21                  ( (IR(1)) and (IR(0)) );
22
23          MSA(0) <= ( not(IR(1)) and not(IR(0)) ) or
24                  ( not(IR(1)) and (IR(0)) ) or
25                  ( (IR(1)) and (IR(0)) );
26
27          MSB(1) <= ( not(IR(1)) and (IR(0)) ) or
28                  ( (IR(1)) and not(IR(0)) ) or
29                  ( (IR(1)) and (IR(0)) );
30
31          MSB(0) <= ( not(IR(1)) and not(IR(0)) );
32
33          MSC(2) <= ( not(IR(1)) and (IR(0)) ) or
34                  ( (IR(1)) and (IR(0)) );
35
36          MSC(1) <= ( (IR(1)) and (IR(0)) );
37
38          MSC(0) <= ( not(IR(1)) and (IR(0)) ) or
39                  ( (IR(1)) and (IR(0)) );
40
41      end behavior;
42

```

Figure 4: VHDL Code for Part 1

Lab 6 Part 1  
Name: Emilee Zhou  
Class #: 11195  
PI's Name: Keith Khadar  
Description: RALU with IR Controller

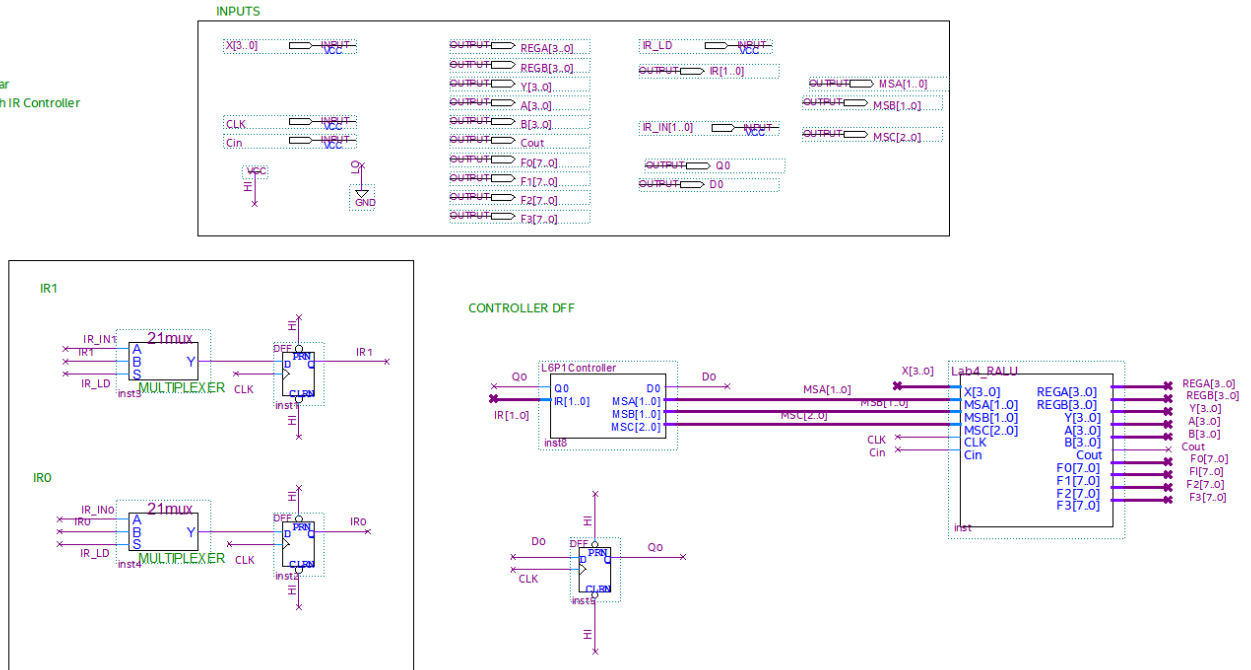


Figure 5: BDF for Part 1

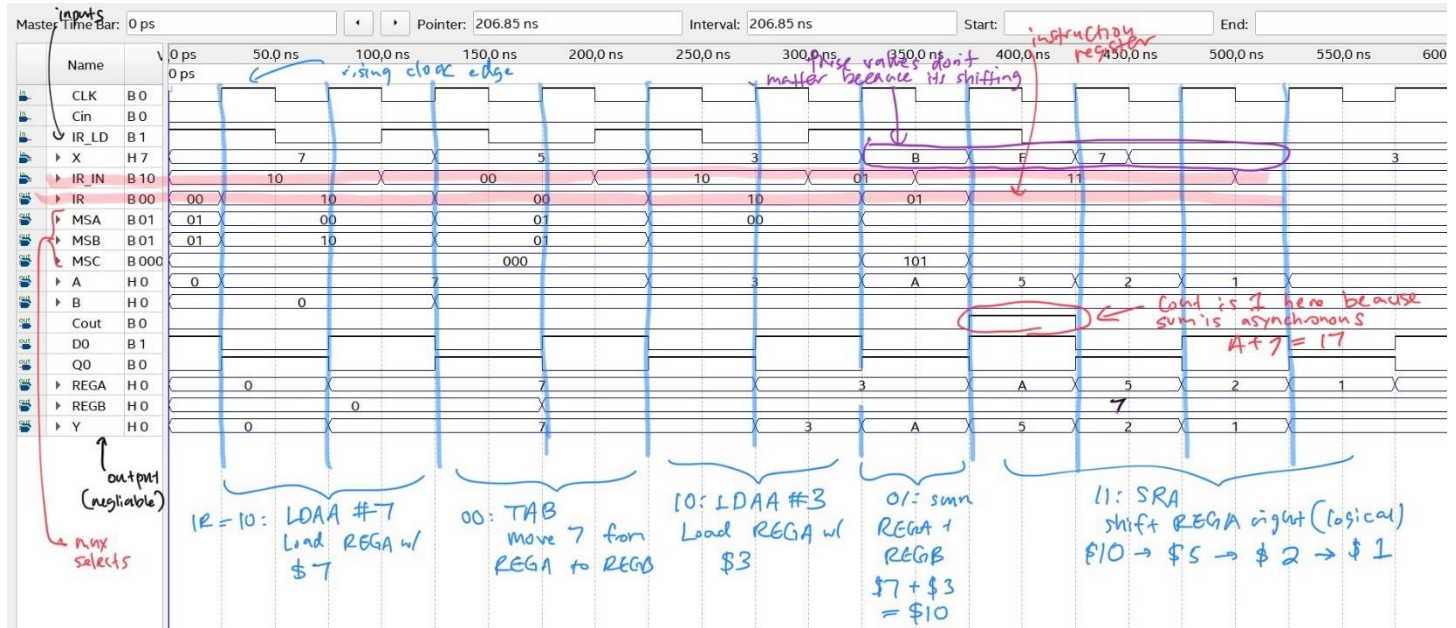


Figure 6: Functional Simulation for Part 1

PART 2: SECOND RALU CONTROLLER WITH ROM

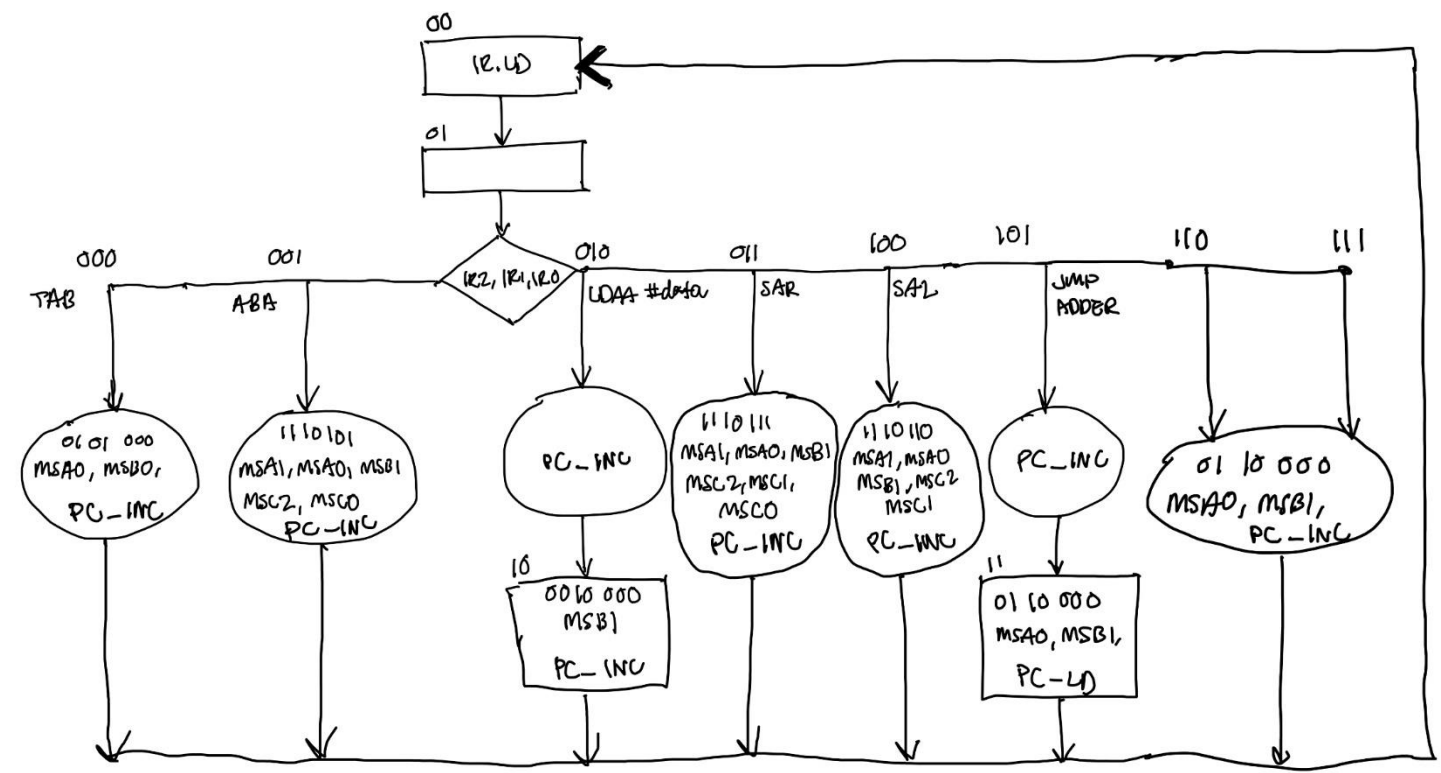


Figure 7: Controller ASM for Part 2

Table 1: Next State Truth Table for Part 2

IR2	IR1	IR0	Q1	Q0	D1	D0	MSA1	MSA0	MSB1	MSB0	MSC2	MSC1	MSC0	IR.LD	PC_INC	PC_LD
-	-	-	0	0	0	1	0	1	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0
0	0	1	0	1	0	0	1	1	1	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0	1	1	0	0	0	0	0	1	0
0*	1*	0*	1	0	0	0	0	0	1	0	0	0	0	0	1	0
0	1	1	0	1	0	0	1	1	1	0	1	1	1	0	1	0
1	0	0	0	1	0	0	1	1	1	0	1	1	0	0	1	0
1	0	1	0	1	1	1	0	1	1	0	0	0	0	0	1	0
1*	0*	1*	1	1	0	0	0	1	1	0	0	0	0	0	0	1
1	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1	0
1	1	1	0	1	0	0	0	1	1	0	0	0	0	0	1	0

```

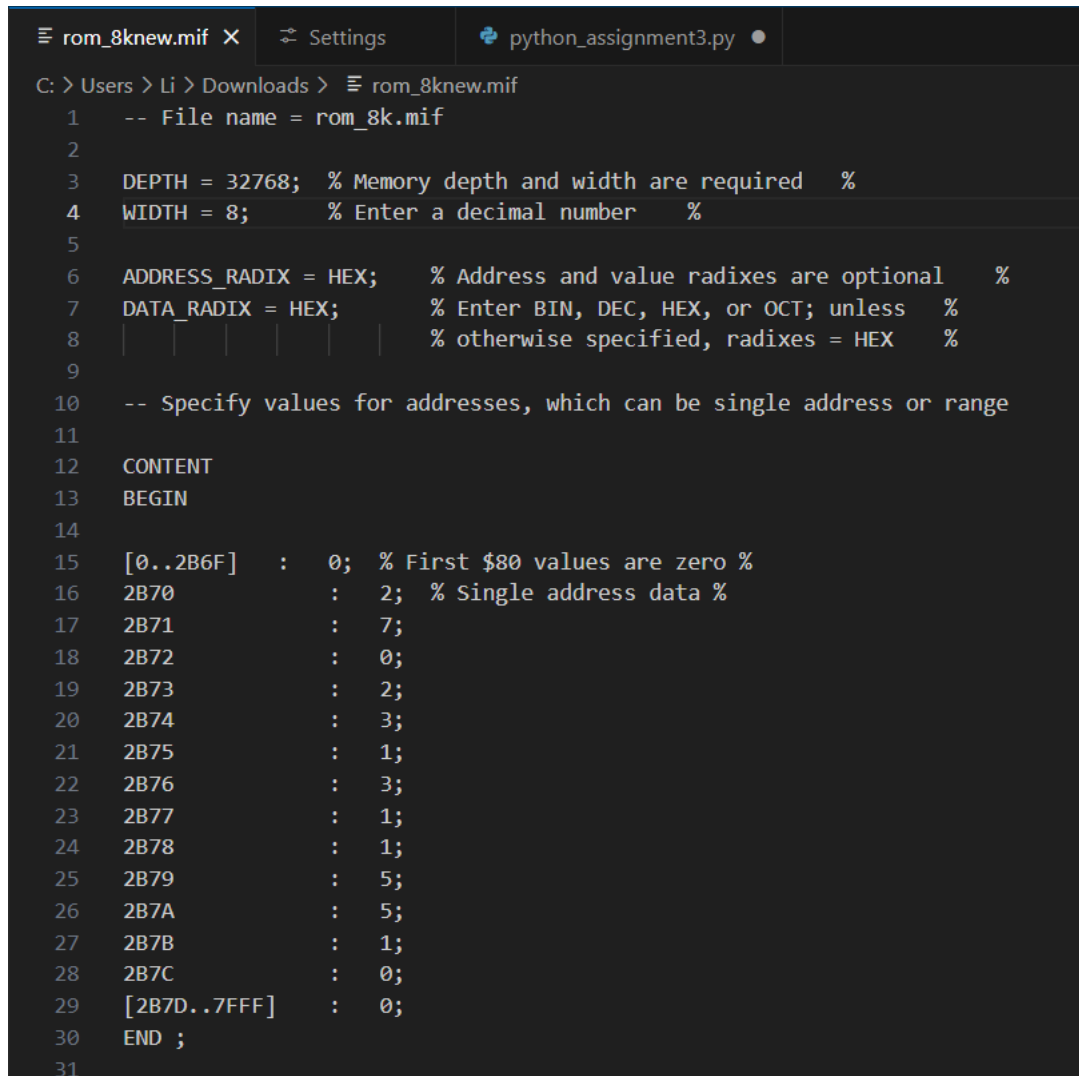
1  library ieee; use ieee.std_logic_1164.all;
2
3  entity L6P2Cont is port(
4      D1, D0, IRLD, PC_INC, PC_LD : out std_logic;
5      Q1, Q0 : in std_logic;
6      IR : in std_logic_vector (2 downto 0);
7      MSA : out std_logic_vector (1 downto 0);
8      MSB : out std_logic_vector (1 downto 0);
9      MSC : out std_logic_vector (2 downto 0));
10
11  end L6P2Cont;
12
13  architecture behavior of L6P2Cont is
14      --signal CW, EV_L : std_logic;
15  begin
16
17      D1 <= ( not(IR(2)) and (IR(1)) and not(IR(0)) and not(Q1) and (Q0) ) or
18            ( (IR(2)) and not(IR(1)) and (IR(0)) and not(Q1) and (Q0) );
19
20      D0 <= ( not(Q1) and not(Q0) ) or
21            ( (IR(2)) and not(IR(1)) and (IR(0)) and not(Q1) and (Q0) );
22
23      IRLD <= ( not(Q1) and not(Q0) );
24
25      PC_INC <= ( Q1 or Q0 ) and (not(IR(2)) or (IR(1)) or not(IR(0)) or not(Q1) or not(Q0));
26
27      PC_LD <= ( (IR(2)) and not(IR(1)) and (IR(0)) and (Q1) and (Q0) );
28
29      MSA(1) <= ( not(IR(2)) and not(IR(1)) and (IR(0)) and not(Q1) and (Q0) ) or
30                ( not(IR(2)) and (IR(1)) and (IR(0)) and not(Q1) and (Q0) ) or
31                ( (IR(2)) and not(IR(1)) and not(IR(0)) and not(Q1) and (Q0) );
32
33      MSA(0) <= ( (IR(2)) or not(IR(1)) or (IR(0)) or not(Q1) or (Q0) );
34
35      MSB(1) <= ( (IR(2)) or (IR(1)) or (IR(0)) or (Q1) or not(Q0) );
36
37      MSB(0) <= ( not(IR(2)) and not(IR(1)) and not(IR(0)) and not(Q1) and (Q0) );
38
39      MSC(2) <= ( not(IR(2)) and not(IR(1)) and (IR(0)) and not(Q1) and (Q0) ) or
40                ( not(IR(2)) and (IR(1)) and (IR(0)) and not(Q1) and (Q0) ) or
41                ( (IR(2)) and not(IR(1)) and not(IR(0)) and not(Q1) and (Q0) );
42
43      MSC(1) <= ( not(IR(2)) and (IR(1)) and (IR(0)) and not(Q1) and (Q0) ) or
44                ( (IR(2)) and not(IR(1)) and not(IR(0)) and not(Q1) and (Q0) );
45
46      MSC(0) <= ( not(IR(2)) and not(IR(1)) and (IR(0)) and not(Q1) and (Q0) ) or
47                ( not(IR(2)) and (IR(1)) and (IR(0)) and not(Q1) and (Q0) );
48
49  end behavior;
50
51
52

```

Figure 8: VHDL Code for Part 2

Table 2: Program for Part 2

Addr		Mach Codes	A	B	A	B	A	B	A	B
\$2B70	LDAA #7	\$2, #7	0111	0000						
\$2B72	TAB	\$0	0111	0111						
\$2B73	LDAA #3	\$2, #3	0011	0111						
\$2B75	ABA	\$1	1010	0111	1010	0111	1010	0111	1010	0111
\$2B76	SAR	\$3	0101	0111	0101	0111	0101	0111	0101	0111
\$2B77	ABA	\$1	1100	0111	1100	0111	1100	0111	1100	0111
\$2B78	ABA	\$1	0011	0111	0011	0111	0011	0111	0011	0111
\$2B79	JMP 5	\$5, #5	0011	0111	0011	0111	0011	0111	0011	0111
\$2B7B	ABA	\$1								
\$2B7C	TAB	\$0								



```
1  -- File name = rom_8k.mif
2
3  DEPTH = 32768; % Memory depth and width are required %
4  WIDTH = 8; % Enter a decimal number %
5
6  ADDRESS_RADIX = HEX; % Address and value radices are optional %
7  DATA_RADIX = HEX; % Enter BIN, DEC, HEX, or OCT; unless %
8  | | | | | % otherwise specified, radices = HEX %
9
10 -- Specify values for addresses, which can be single address or range
11
12 CONTENT
13 BEGIN
14
15 [0..2B6F] : 0; % First $80 values are zero %
16 2B70 : 2; % Single address data %
17 2B71 : 7;
18 2B72 : 0;
19 2B73 : 2;
20 2B74 : 3;
21 2B75 : 1;
22 2B76 : 3;
23 2B77 : 1;
24 2B78 : 1;
25 2B79 : 5;
26 2B7A : 5;
27 2B7B : 1;
28 2B7C : 0;
29 [2B7D..7FFF] : 0;
30 END ;
31
```

Figure 9: MIF file for Part 2



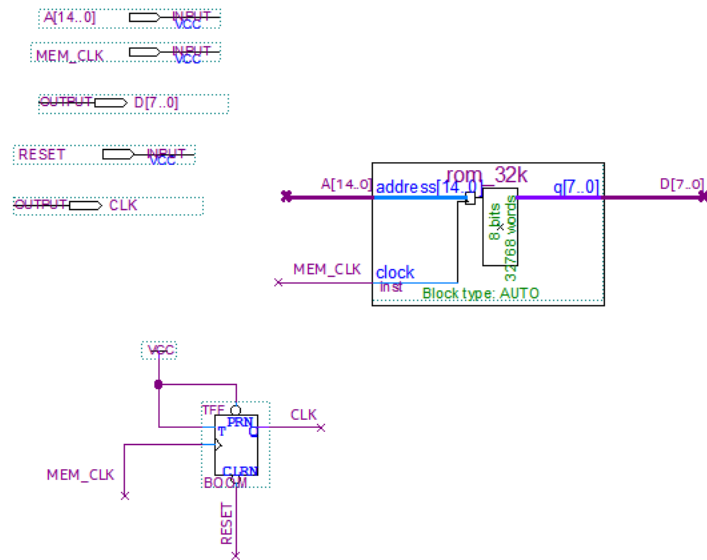


Figure 10: BDF of ROM 32K x 8

Lab 6 Part 2  
 Name: Emilee Zhou  
 Class #: 11195  
 Pf's Name: Keith Khadar  
 Description: RALU with IR Controller and ROM

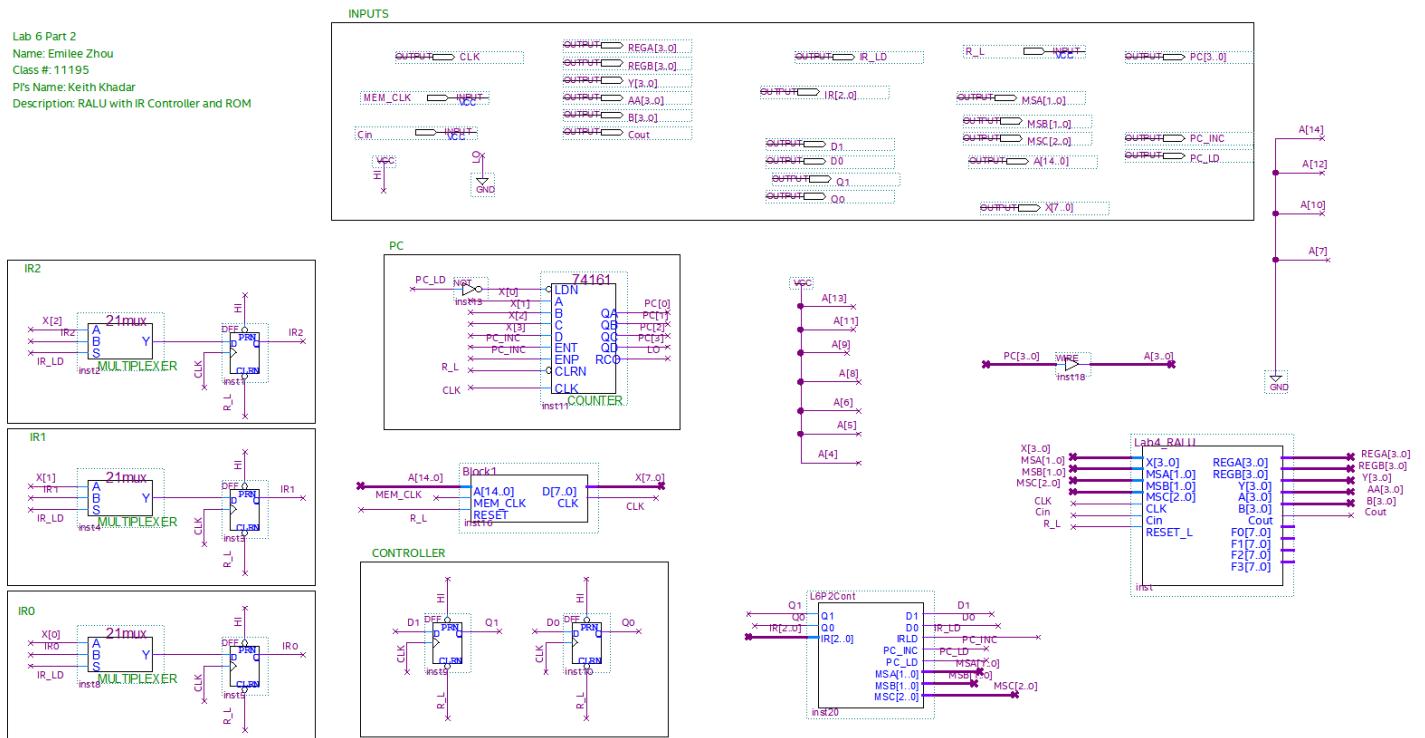


Figure 11: BDF of RALU with Controller and ROM for Part 2

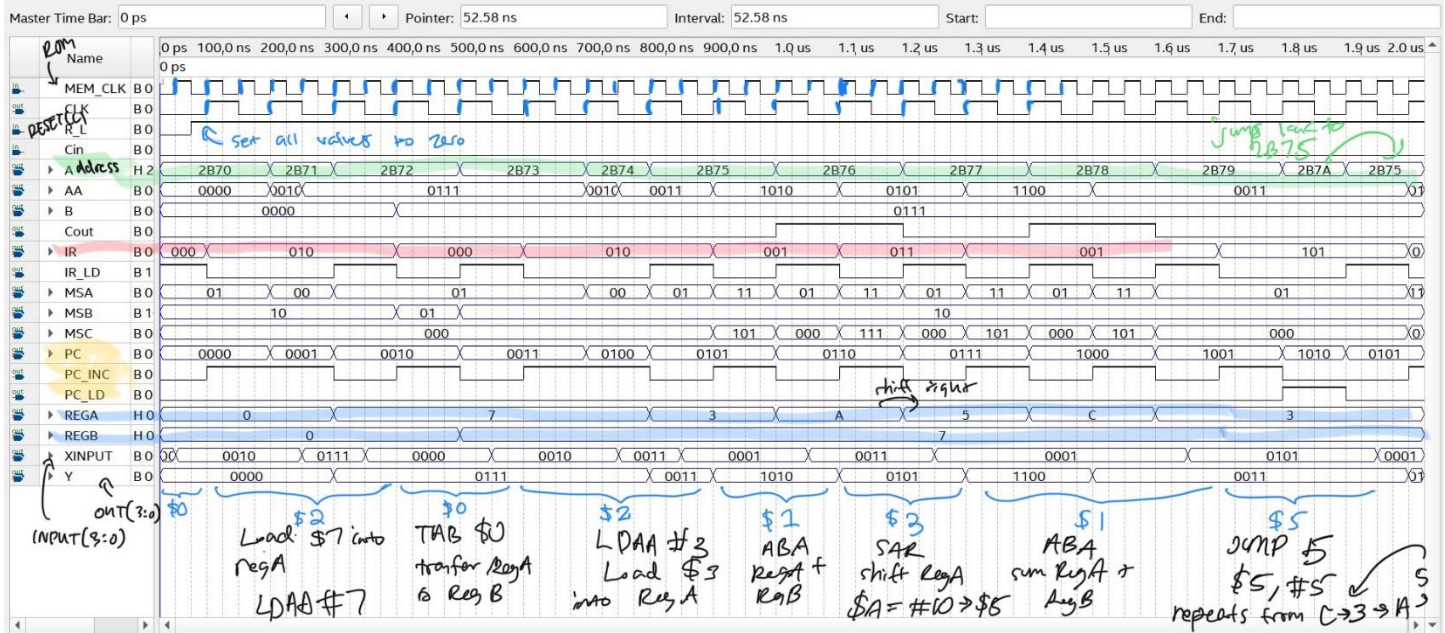


Figure 12: Functional Simulation for Part 2