

Part 1:

The student ID that will be used throughout the duration of this lab is 501103255 where $A = (32)_{16}$ and $B = (55)_{16}$

To convert these hexadecimal values into binary representation $A = (0011\ 0010)_2$ and $B = (0101\ 0101)_2$

Part 2:

Latch 1 & 2

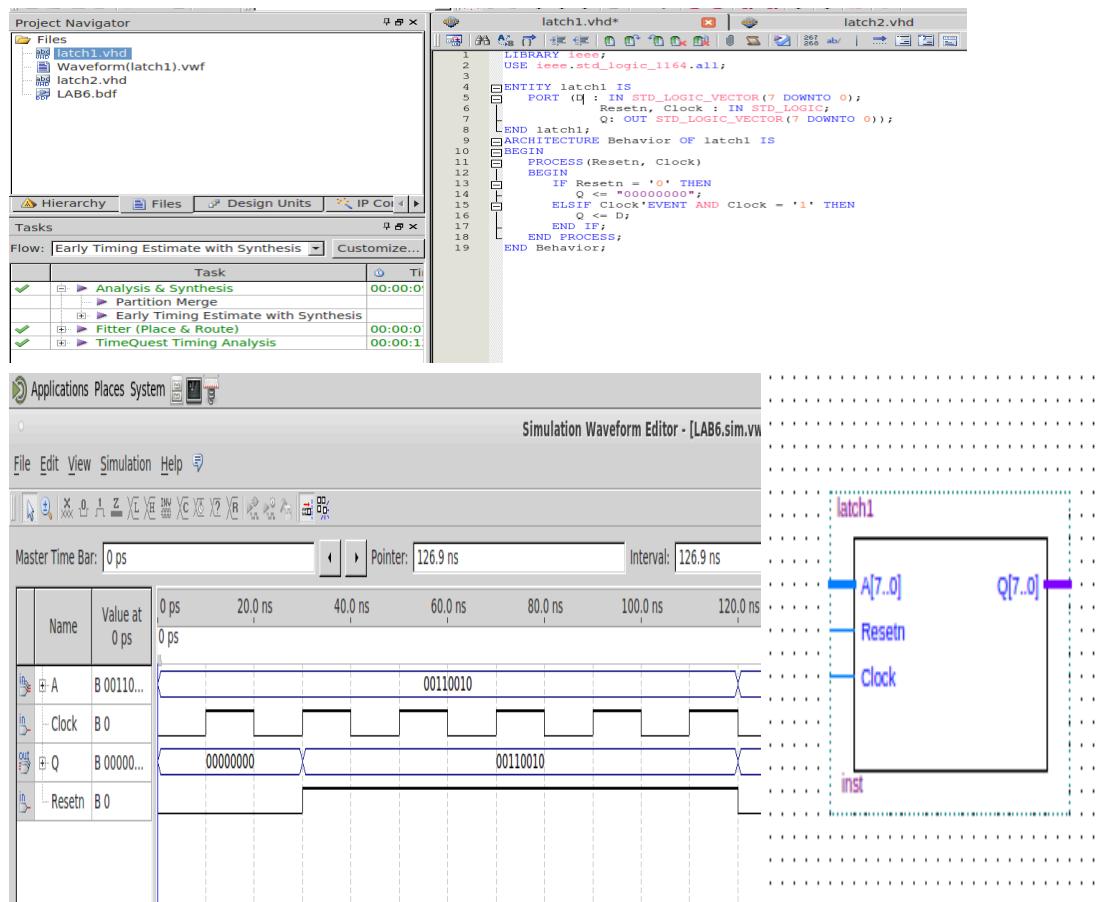


Figure 1-3: Latch 1 VHDL code, waveform, and block diagram

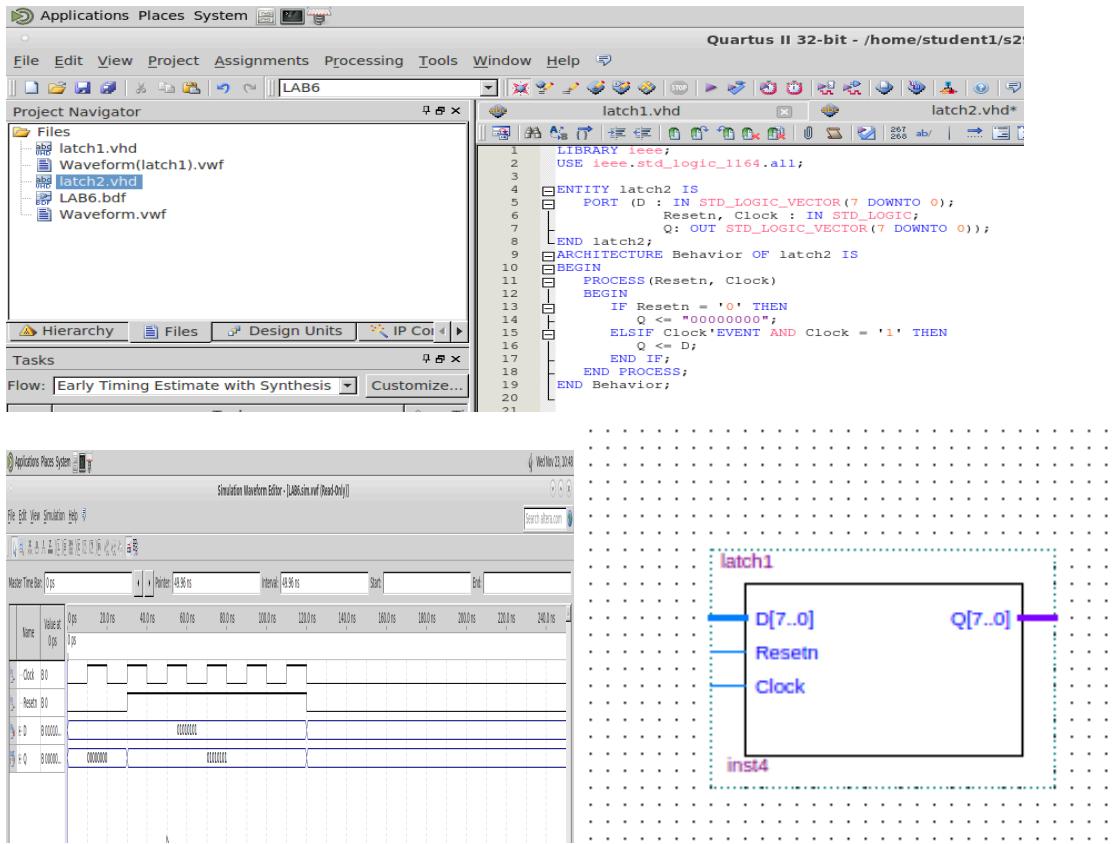


Figure 4-6: Latch 2 VHDL code, waveform, and block diagram

The waveform results from the VHDL code from latch 1 & latch 2 present the binary values of A and B as the outputs respectively. The waveform also includes a clock, reset, and the value of A as the input.

Part 3:

FSM & DECODER 4 to 16

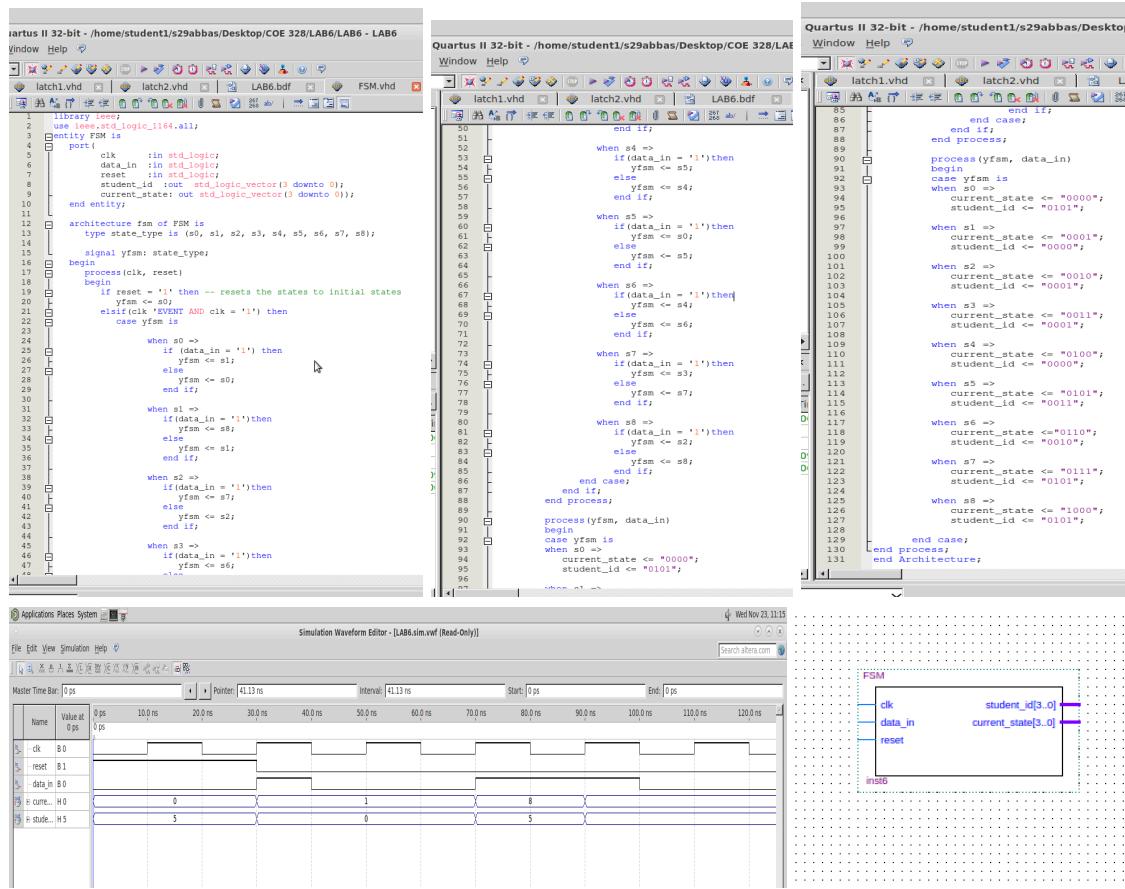


Figure 7-11: FSM VHDL code, waveform, and block diagram

The FSM waveform displays the student number and current state as the output given the selected data, clock, and reset. The output depends on the data that is chosen to be switched on (high bar) as it is processed through the FSM machine.

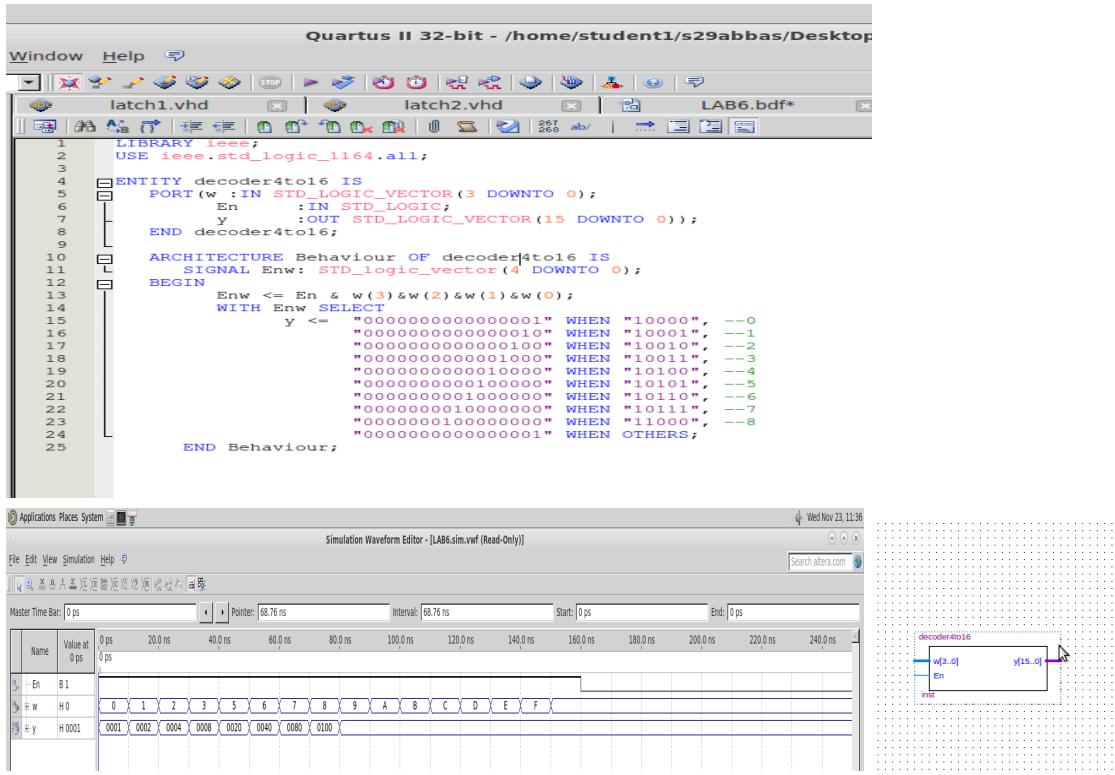


Figure 12-14: 4-16 decoder VHDL code, waveform, and block diagram

The 4-16 decoder waveform uses the hexadecimal number set as the input along with a switch that is turned on (a constant high bar). The output takes the input of 4-bits and modifies it into 16-bits, hence a 4-16 decoder.

Part 4:

ALU

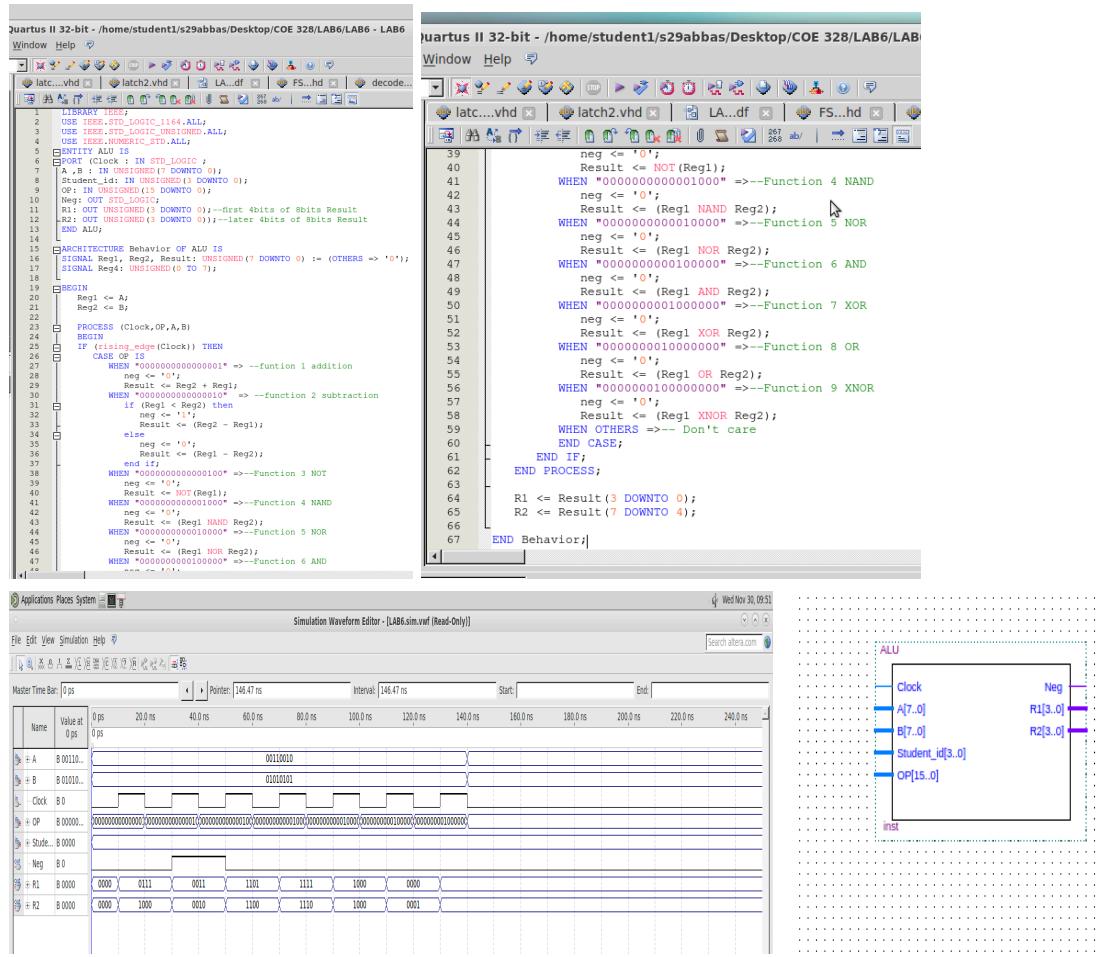


Figure 15-18: ALU VHDL code, waveform, and block diagram

The ALU has direct input and output access to the processor controller, main memory. The input consists of an instruction word that contains an operation code or "opcode," one or more operands.. The operation code tells the ALU what operation to perform and the operands are used in the operation.

Part 6:

Combined block diagram

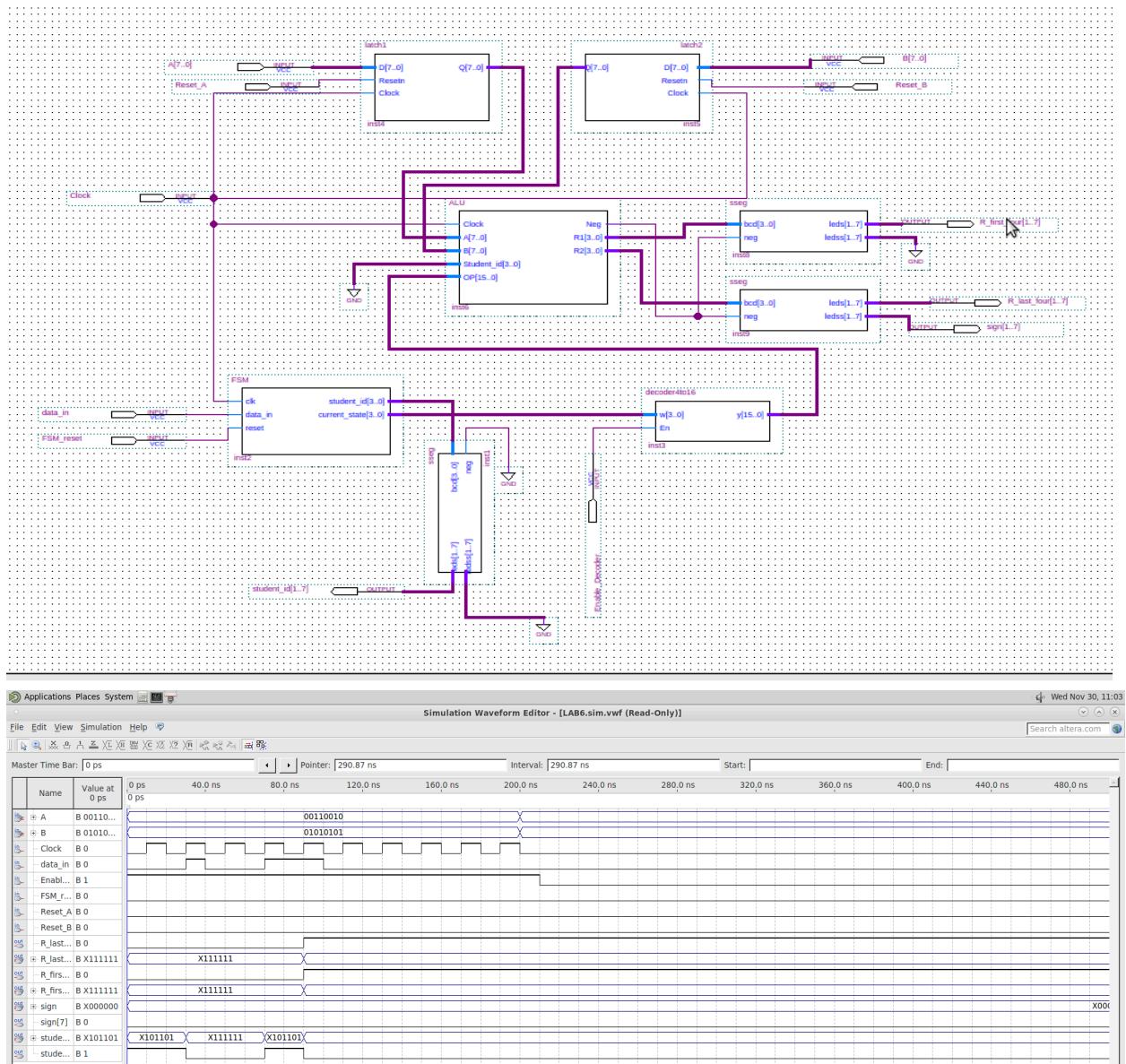


Figure 19-20: Combined Block VHDL code, waveform, and block diagram

The combined block consists of inputs A and B which one of our student IDs was used for, clock, data_in which we put as random inputs to understand what the outputs will be, Enable which is always on, and the reset is turned off.

Part 8:

Problem 2:

The figure displays three windows from the Quartus II software interface:

- modified_ALU.vhd**: The original VHDL code for the ALU. It defines a package with STD_LOGIC_UNSIGNED types, an entity ALU_modified with two ports: Clock and R1, R2, and a single output Result. The architecture Behavior contains a process that handles various operations based on the inputs and clock edge.
- modified_ALU_0000000000000000.vhd**: A copy of the modified_ALU.vhd code, likely for simulation purposes. It includes the same package and entity definitions but lacks the specific clock edge detection logic found in the original.
- Simulation Waveform Editor - [LAB6.sim.vwf (Read-Only)]**: This window shows the simulation results for the modified ALU. The timeline ranges from 0 ps to 240.0 ns. Traces are provided for all signals: A, B, Clock, Neg, OP, R1, R2, and Strobe. The waveforms show the inputs A and B being switched between values, the clock signal, the negation of the result, the operation selection (OP), and the outputs R1 and R2. The result signal shows the calculated 16-bit result of the ALU operation.

Figure 21-23: Problem 2, part a, VHDL code, waveform

Problem 3:

```
37      If (Reg1 = Student_Id) OR (Reg2 = Student_Id) then
38      else
39          Result <= "0001";
40      end if;
41      WHEN "00000000000000001000" =>
42          Reg <= "0";
43          If (Reg1 = Student_Id) OR (Reg2 = Student_Id) then
44          else
45              Result <= "0000";
46          end if;
47          Result <= "0001";
48      end if;
49      WHEN "0000000000000000100000" =>
50          Reg <= "0";
51          If (Reg1 = Student_Id) OR (Reg2 = Student_Id) then
52          else
53              Result <= "00000";
54          end if;
55          Result <= "0001";
56      end if;
57      WHEN "000000000000000010000000" =>
58          Reg <= "0";
59          If (Reg1 = Student_Id) OR (Reg2 = Student_Id) then
60          else
61              Result <= "0001";
62      end if;
63      WHEN "00000000000000001000000000" =>
64          Reg <= "0";
65          If (Reg1 = Student_Id) OR (Reg2 = Student_Id) then
66          else
67              Result <= "0000";
68      end if;
69      WHEN "0000000000000000100000000000" =>
70          Reg <= "0";
71          If (Reg1 = Student_Id) OR (Reg2 = Student_Id) then
72          else
73              Result <= "0001";
74      end if;
75      WHEN "000000000000000010000000000000" =>
76          Reg <= "0";
77          If (Reg1 = Student_Id) OR (Reg2 = Student_Id) then
78          else
79              Result <= "0001";
80      end if;
```

```
34      Result <= "0001";
35      end_if;
36      WHEN "0000000000000000" =>
37          neg <= "0";
38          if ((Reg1 = Student_id) OR (Reg2 = Student_id)) then
39              Result <= "0000";
40          else
41              Result <= "0001";
42          end_if;
43      WHEN "0000000000000000" =>
44          neg <= "0";
45          if ((Reg1 = Student_id) OR (Reg2 = Student_id)) then
46              Result <= "0000";
47          else
48              Result <= "0001";
49          end_if;
50      WHEN "0000000000000000" =>
51          neg <= "0";
52          if ((Reg1 = Student_id) OR (Reg2 = Student_id)) then
53              Result <= "0000";
54          else
55              Result <= "0001";
56          end_if;
57      WHEN OTHERS => Don't care
58  END CASE;
59  END IF;
60 END PROCESS;
61 END Behavior;
```

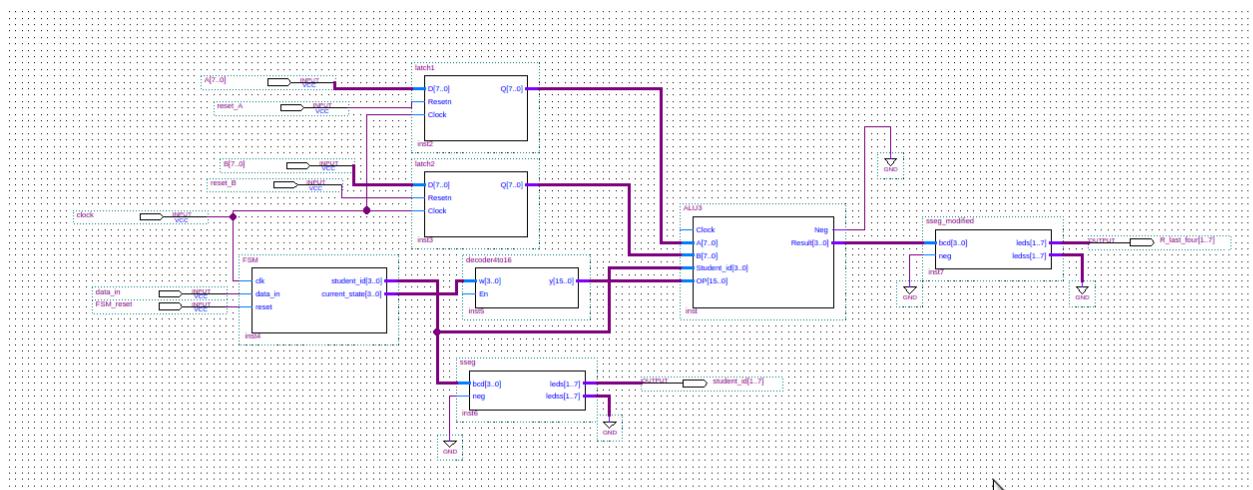


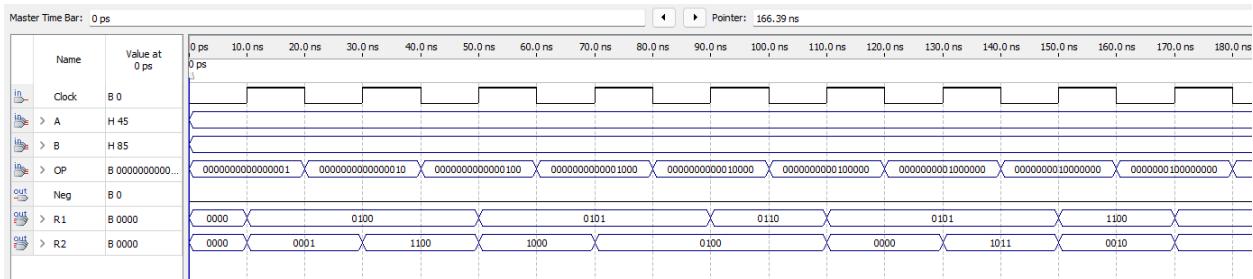
Figure 24-27: Problem 3 VHDL code, modified circuit diagram, and waveform

Problem 2:

```

1  --501124585
2  LIBRARY IEEE;
3  USE IEEE.STD_LOGIC_1164.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5  USE IEEE.NUMERIC_STD.ALL;
6  ENTITY ALU2 IS
7  PORT(
8   |   Clock : IN STD_LOGIC;
9   |   A,B : IN UNSIGNED(7 DOWNTO 0);
10  |  OP : IN UNSIGNED(15 DOWNTO 0);
11  |  Neg : OUT STD_LOGIC;
12  |  R1 : OUT UNSIGNED (3 DOWNTO 0);
13  |  R2 : OUT UNSIGNED (3 DOWNTO 0));
14  END ALU2;
15
16 ARCHITECTURE calculation OF ALU2 IS
17  SIGNAL reg1, reg2, result : UNSIGNED(7 DOWNTO 0):= (OTHERS =>'0');
18  SIGNAL reg4 : UNSIGNED(0 TO 7);
19 BEGIN
20  reg1 <= A;
21  reg2 <= B;
22  PROCESS(Clock, OP)
23  BEGIN
24  IF(rising_edge(Clock)) THEN
25  CASE OP IS
26    WHEN "0000000000000001" =>-- Shift A to right by two bits, input bit = 1 (SHR)
27      result <= shift_left(reg1,2);
28      neg <= '0';
29    WHEN "0000000000000010" =>-- Produce the difference of A and B, then increment by 4
30      result <= ((reg1 - reg2) + 4);
31      IF (result >= 0) THEN
32        neg <= '0';
33      ELSE
34        neg <= '1';
35      END IF;
36    WHEN "0000000000001000" =>-- Find the greater value of A and B and produce the results ( Max(A,B) )
37      neg <= '0';
38      if (Reg1 < Reg2) then
39        Result <= Reg2;
40      else
41        Result <= Reg1;
42      end if;
43    WHEN "0000000000001000" =>-- Swap the upper 4 bits of A by the lower 4 bits of B
44      Result <= Reg1(7 DOWNTO 4) & Reg2(3 DOWNTO 0);
45    WHEN "0000000000010000" =>-- Increment A by 1
46      Result <= Reg1 + "00000001";
47      neg <= '0';
48    WHEN "00000000000100000" =>-- Produce the result of ANDing A and B
49      Result <= Reg1 AND Reg2;
50      neg <= '0';
51    WHEN "0000000001000000" =>-- Invert the upper four bits of A
52      neg <= '0';
53      Result <= NOT(Reg1(7 DOWNTO 4));
54    WHEN "0000000010000000" =>-- Rotate B to left by 3 bits (ROL)
55      neg <= '0';
56      Result <= Reg2(4)&Reg2(3)&Reg2(2)&Reg2(1)&Reg2(0)&Reg2(7)&Reg2(6)&Reg2(5);
57    WHEN "0000000100000000" =>-- Show null on the output
58      neg <= '0';
59      result <= "-----";
60    WHEN OTHERS =>
61
62          END CASE;
63
64      END IF;
65
66      END PROCESS;
67
68      R1 <= Result(3 DOWNTO 0);
69      R2 <= Result(7 DOWNTO 4);
70
71      END calculation;

```



Problem 3:

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3 USE ieee.std_logic_unsigned.all;
4 USE ieee.numeric_std.all;
5
6 ENTITY ALU3 IS
7 PORT( Clock      : IN STD_LOGIC;
8        A,B       : IN unsigned(7 DOWNTO 0);
9        student_id : IN unsigned(3 DOWNTO 0);
10       OP        : IN unsigned(0 TO 15);
11       Neg       : OUT STD_LOGIC;
12       R         : OUT UNSIGNED (7 DOWNTO 0);
13       R1        : OUT unsigned(3 DOWNTO 0);
14       R2        : OUT unsigned(3 DOWNTO 0));
15 END ALU3;
16
17 ARCHITECTURE calculation OF ALU3 IS
18 SIGNAL Reg1, Reg2, Result : unsigned(7 downto 0):=
19 SIGNAL Reg4: UNSIGNED (0 to 7);
20 BEGIN
21     Reg1 <= A;
22     Reg2 <= B;
23     Process (Clock, OP)
24     BEGIN
25         if(rising_edge(Clock)) THEN
26             CASE OP IS
27                 WHEN "0000000000000001" => --5
28                     Result <= "00000000";
29                     neg <= '0';
30
31                 WHEN "0000000000000010" => --0
32                     Result <= "00000001";
33                     neg <= '0';
34
35                 WHEN "0000000000000100" => --1
36                     Result <= "00000000";
37                     neg <= '0';
38
39                 WHEN "0000000000001000" => --1
40                     Result <= "00000000";
41                     neg <= '0';
42
43                 WHEN "00000000000010000" => --2
44                     Result <= "00000001";
45                     neg <= '0';
46
47                 WHEN "00000000000100000" => --4
48                     Result <= "00000001";
49                     neg <= '0';
50
51                 WHEN "0000000001000000" => --5
52                     Result <= "00000000";
53                     neg <= '0';
```

```

54
55      WHEN "0000000010000000" => --8
56      Result <= "00000001";
57      neg <= '0';
58
59      WHEN "0000000100000000" => --5
60      Result <= "00000000";
61      neg <= '0';
62
63      WHEN OTHERS =>
64      END CASE;
65      END IF;
66      END PROCESS;
67      R <= Result (7 DOWNTO 0);
68      END calculation;
69

```

