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

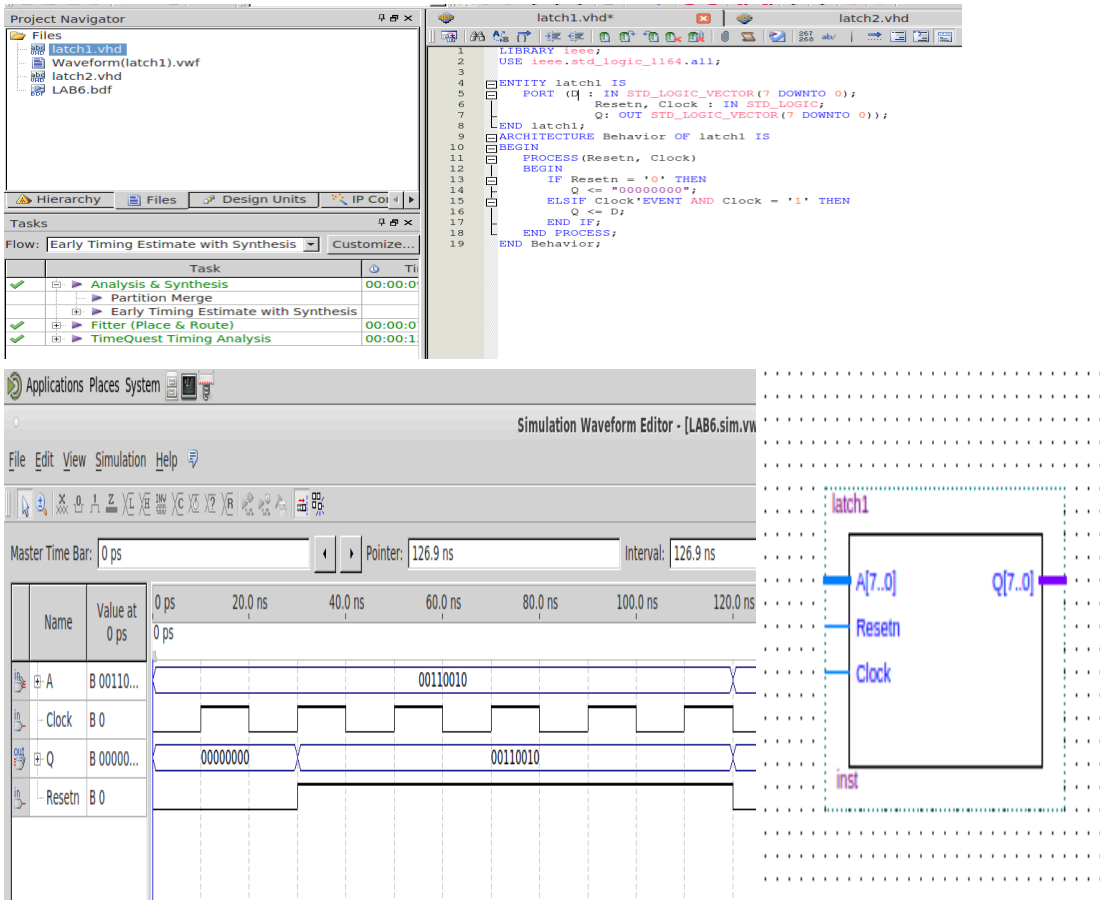


Figure 1-3: Latch 1 VHDL code, waveform, and block diagram

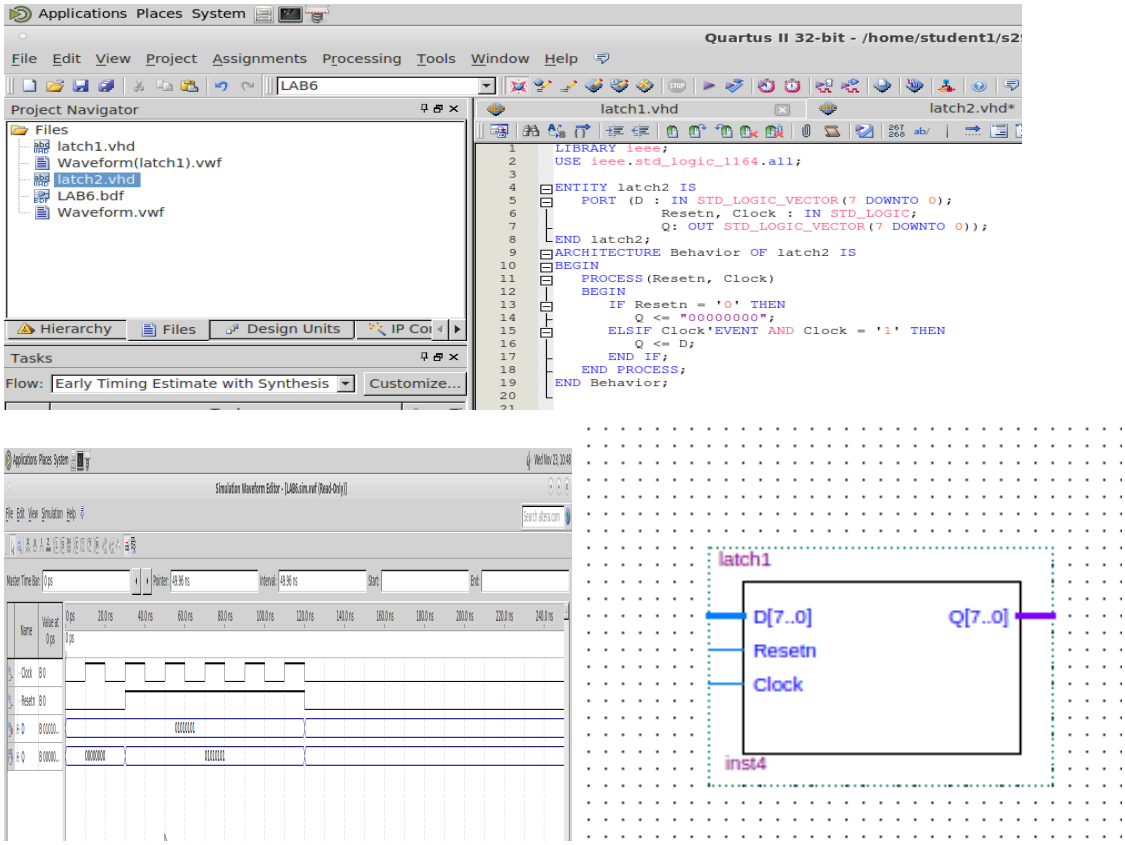


Figure 4-6: Latch 2 VHDL code, waveform, and block diagram

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

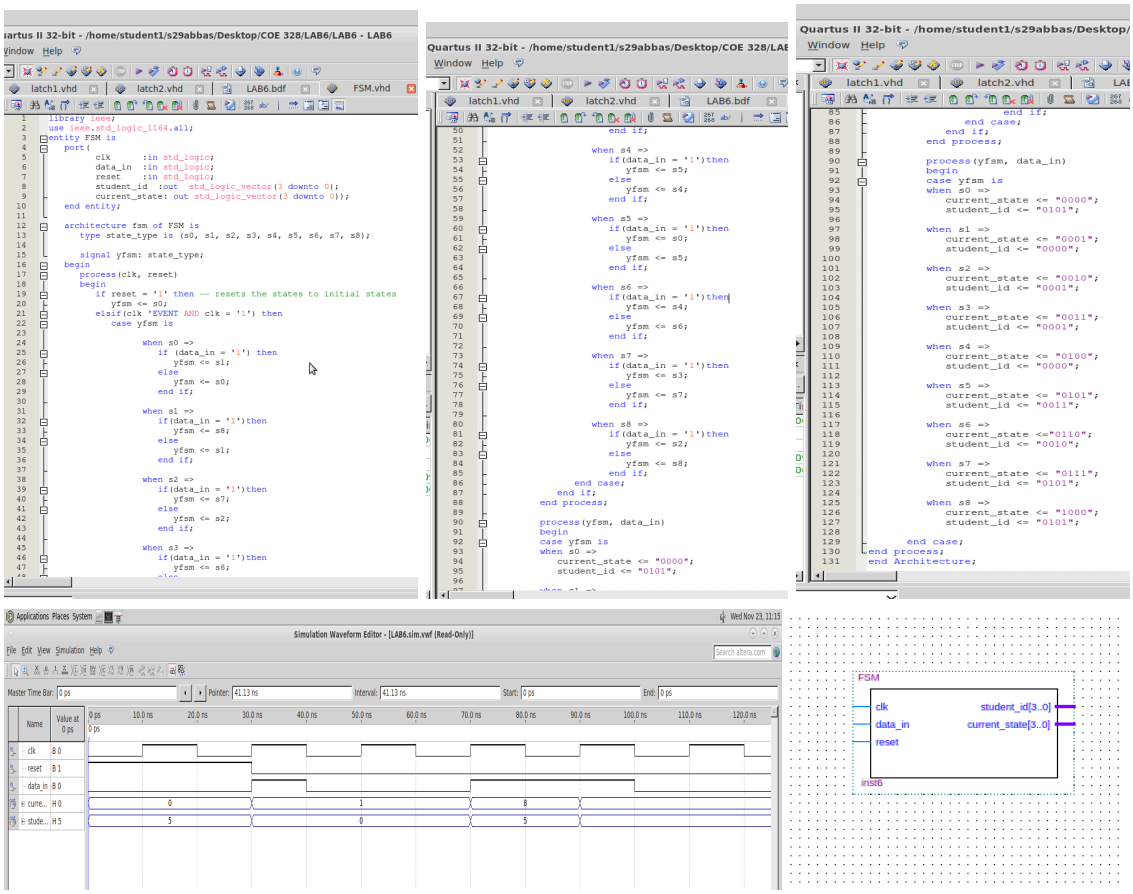


Figure 7-11: FSM VHDL code, waveform, and block diagram

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.

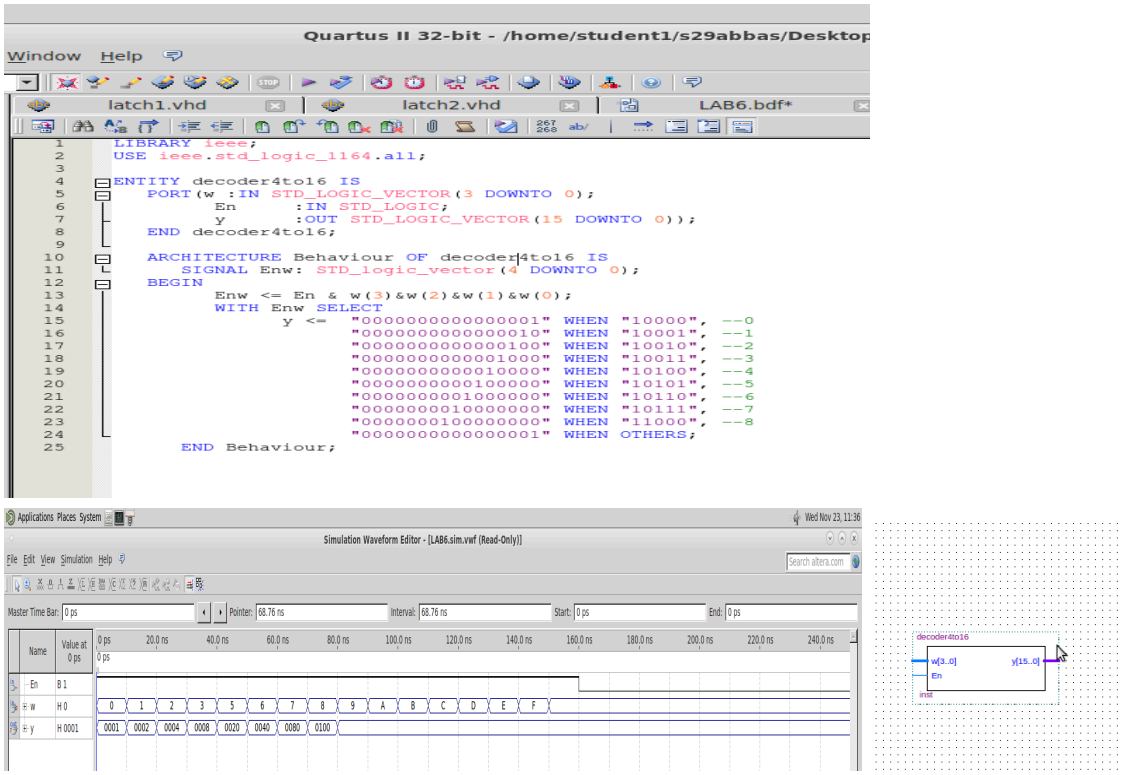


Figure 12-14: 4-16 decoder VHDL code, waveform, and block diagram

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

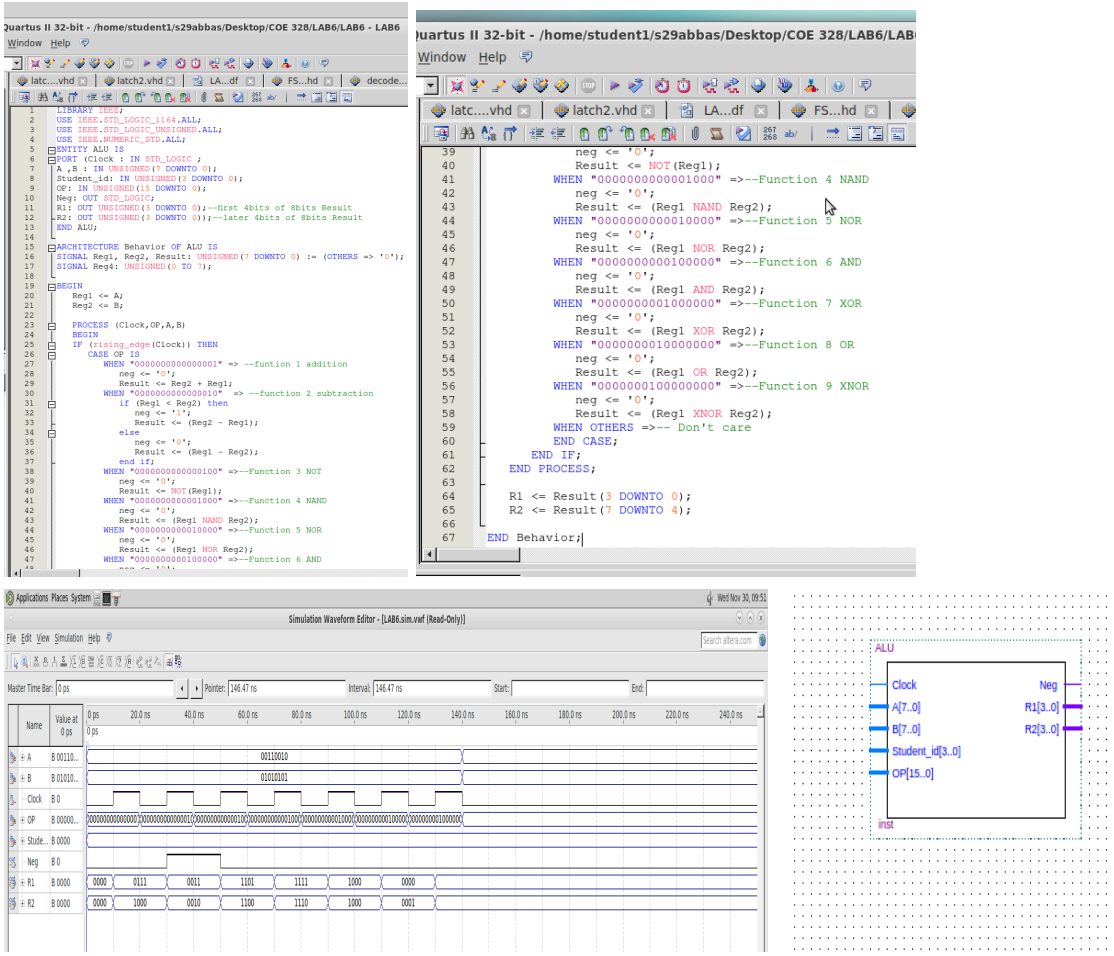


Figure 15-18: ALU VHDL code, waveform, and block diagram

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

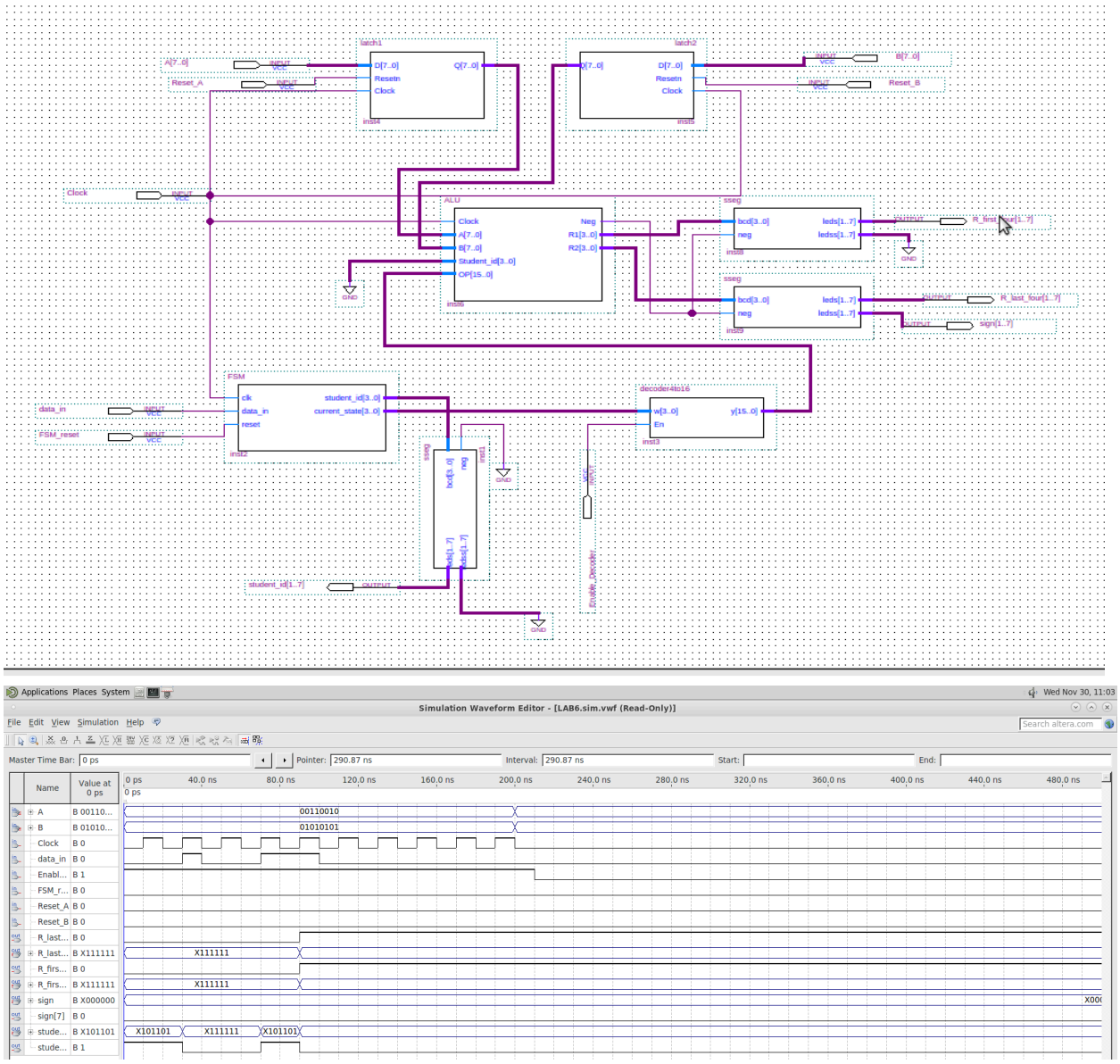


Figure 19-20: Combined Block VHDL code, waveform, and block diagram

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:

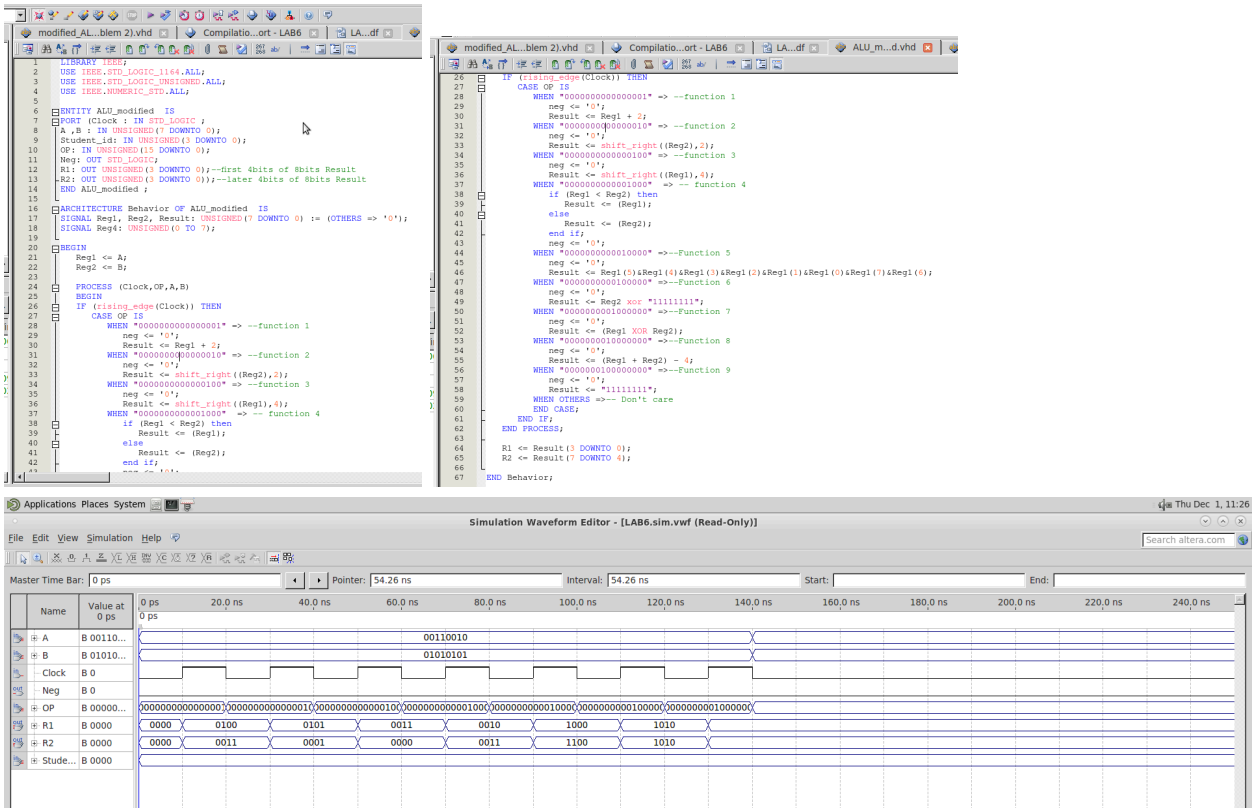


Figure 21-23: Problem 2, part a, VHDL code, waveform

Problem 3:

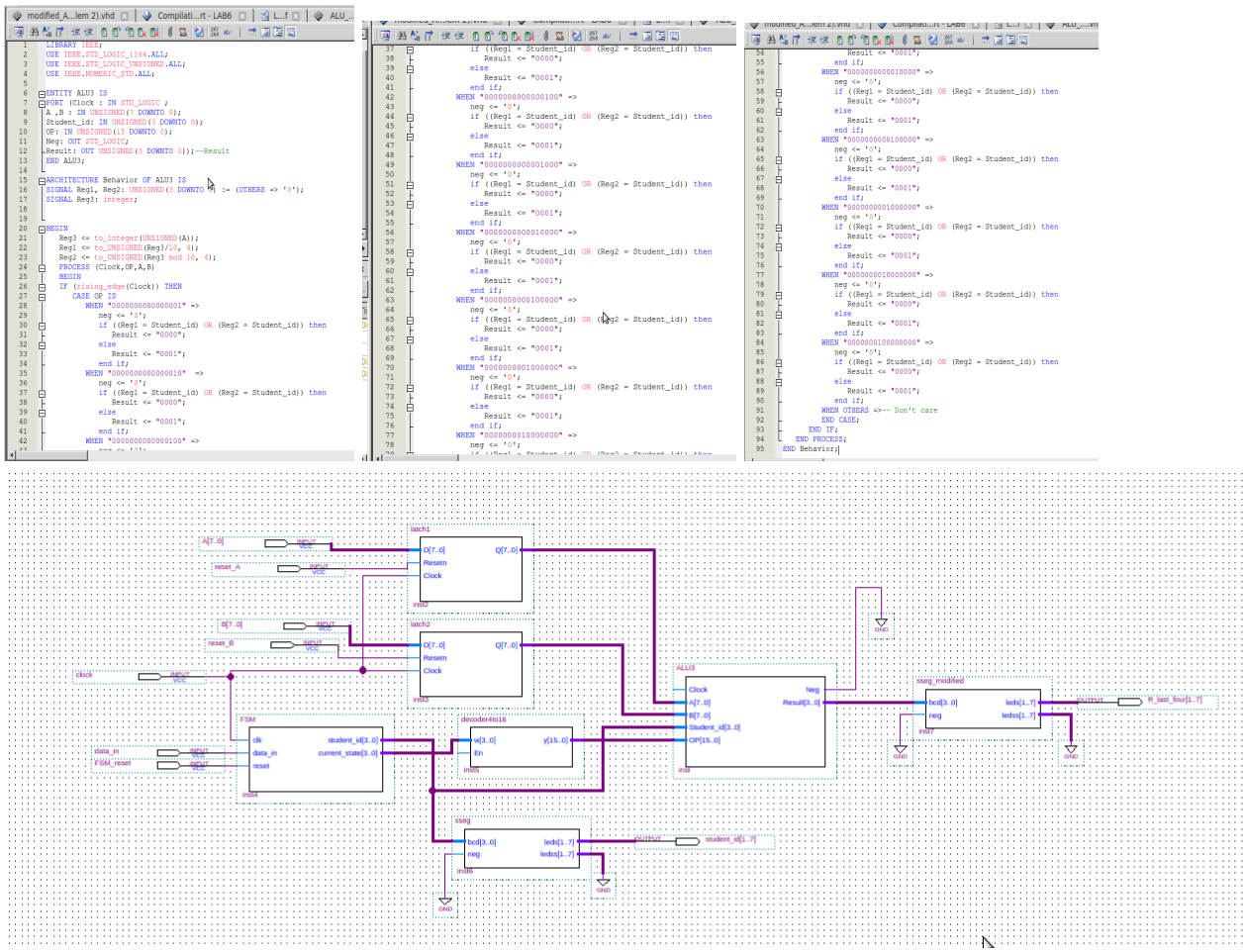


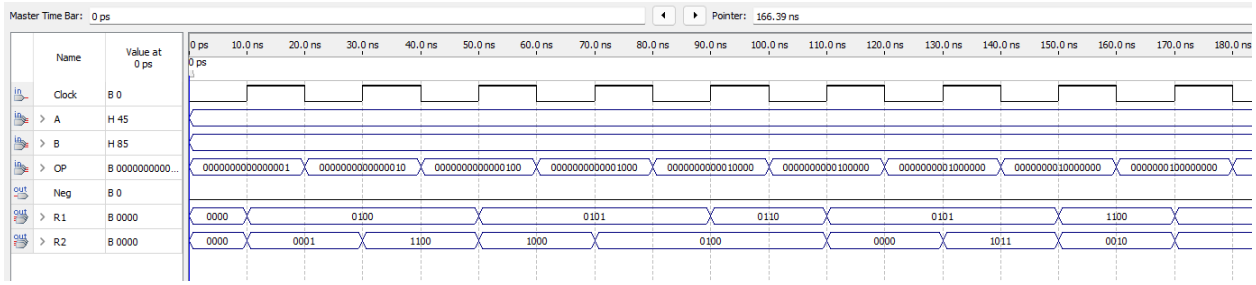
Figure 24-27: Problem 3 VHDL code, modified circuit diagram, and waveform

Problem 2:

```
--501124585
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.NUMERIC_STD.ALL;
ENTITY ALU2 IS
PORT(
    Clock : IN STD_LOGIC;
    A,B : IN UNSIGNED(7 DOWNTO 0);
    OP : IN UNSIGNED(15 DOWNTO 0);
    Neg : OUT STD_LOGIC;
    R1 : OUT UNSIGNED (3 DOWNTO 0);
    R2 : OUT UNSIGNED (3 DOWNTO 0));
END ALU2;

ARCHITECTURE calculation OF ALU2 IS
SIGNAL reg1, reg2, result : UNSIGNED(7 DOWNTO 0):= (OTHERS =>'0');
SIGNAL reg4 : UNSIGNED(0 TO 7);
BEGIN
reg1 <= A;
reg2 <= B;
PROCESS(Clock, OP)
BEGIN
   IF(rising_edge(Clock)) THEN
      CASE OP IS
                    WHEN "0000000000000001" =>-- Shift A to right by two bits, input bit = 1 (SHR)
                     result <= shift_left(reg1,2);
                     neg <= '0';
                    WHEN "0000000000000010" =>-- Produce the difference of A and B, then increment by 4
                      result <= ((reg1 - reg2) + 4);
                      IF (result >= 0) THEN
                            neg <= '0';
                      ELSE
                            neg <= '1';
                      END IF;
                    WHEN "0000000000000100" =>-- Find the greater value of A and B and produce the results ( Max(A,B) )
                     neg <= '0';
                     if (Reg1 < Reg2) then
                         Result <= Reg2;
                     else
                         Result <= Reg1;
                     end if;
                    WHEN "0000000000001000" => -- Swap the upper 4 bits of A by the lower 4 bits of B
                     Result <= Reg1(7 DOWNTO 4) & Reg2(3 DOWNTO 0);
                    WHEN "0000000000010000" => -- Increment A by 1
                     Result <= Reg1 + "00000001";
                     neg <= '0';
                    WHEN "0000000000100000" => -- Produce the result of ANDing A and B
                     Result <= Reg1 AND Reg2;
                     neg <= '0';
                    WHEN "0000000001000000" => -- Invert the upper four bits of A
                     neg <= '0';
                     Result <= NOT(Reg1(7 DOWNTO 4));
                    WHEN "0000000010000000" => -- Rotate B to left by 3 bits (ROL)
                     neg <= '0';
                     Result <= Reg2(4)&Reg2(3)&Reg2(2)&Reg2(1)&Reg2(0)&Reg2(7)&Reg2(6)&Reg2(5);
                    WHEN "0000000100000000" => -- Show null on the output
                     neg <= '0';
                     result <= "--------";
                  WHEN OTHERS =>
```

```
                END CASE;
            END IF;
        END PROCESS;
    R1 <= Result(3 DOWNTO 0);
    R2 <= Result(7 DOWNTO 4);
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 "0000000000010000" => --2
44              Result <= "00000001";
45              neg <= '0';
46
47              WHEN "0000000000100000" => --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
```