COE328-Digital Systems

Name:
Dushel Perera

Part 1

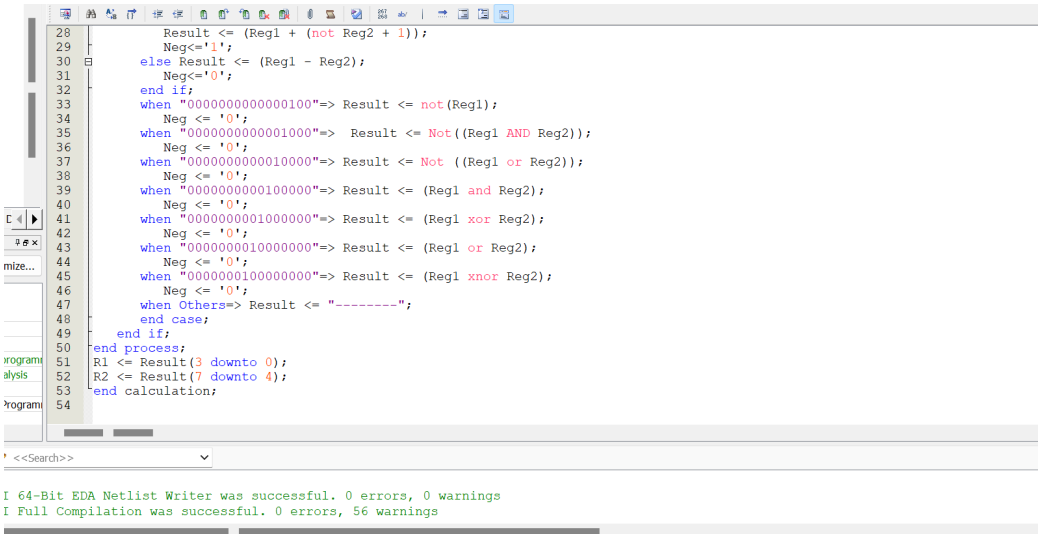
Latch

```
library ieee;
Use ieee.std_logic_1164.all;

Entity latch1 is
   port(a: in STD_logic_vector(7 downto 0);
   Resetn, clock: In Std_logic;
   Q: out std_logic_vector(7 downto 0));
End latch1;
Architecture Behavior OF latch1 is
begin
   process(Resetn, clock)
   begin
      if Resetn = '0' then
         Q <= "00000000";
      Elsif clock = '0' then
         Q <= A;
      End If;
   end process;
end behavior;
```

ALU

```
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
entity ALU is
   port(clock : in std_logic;
      A,B: in unsigned(7 downto 0);
      student_id : in unsigned(3 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 ALU;
architecture calculation of ALU 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"=> Result <= (Reg1 + Reg2);
      when "0000000000000010"=>
      if (Reg2>Reg1) then
         Result <= (Reg1 + (not Reg2 + 1));
         Neg<='1';
```

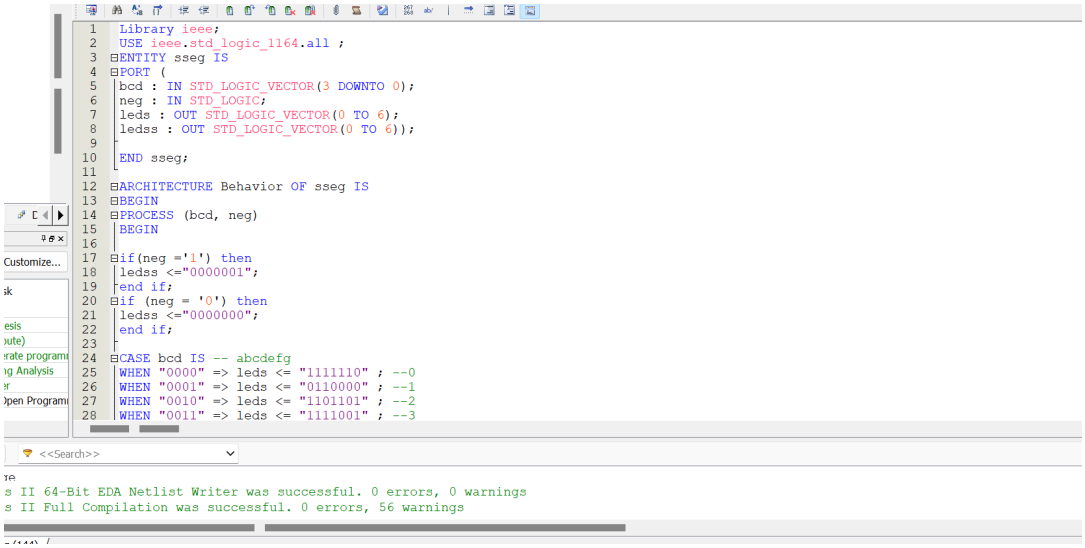
```
rtus II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
rtus II Full Compilation was successful. 0 errors, 56 warnings
```



```
28            Result <= (Reg1 + (not Reg2 + 1));
29            Neg<='1';
30         else Result <= (Reg1 - Reg2);
31            Neg<='0';
32         end if;
33         when "0000000000000100"=> Result <= not(Reg1);
34            Neg <= '0';
35         when "0000000000001000"=>  Result <= Not((Reg1 AND Reg2));
36            Neg <= '0';
37         when "0000000000010000"=> Result <= Not ((Reg1 or Reg2));
38            Neg <= '0';
39         when "0000000000100000"=> Result <= (Reg1 and Reg2);
40            Neg <= '0';
41         when "0000000001000000"=> Result <= (Reg1 xor Reg2);
42            Neg <= '0';
43         when "0000000010000000"=> Result <= (Reg1 or Reg2);
44            Neg <= '0';
45         when "0000000100000000"=> Result <= (Reg1 xnor Reg2);
46            Neg <= '0';
47         when Others=> Result <= "--------";
48         end case;
49      end if;
50   end process;
51   R1 <= Result(3 downto 0);
52   R2 <= Result(7 downto 4);
53   end calculation;
54
```

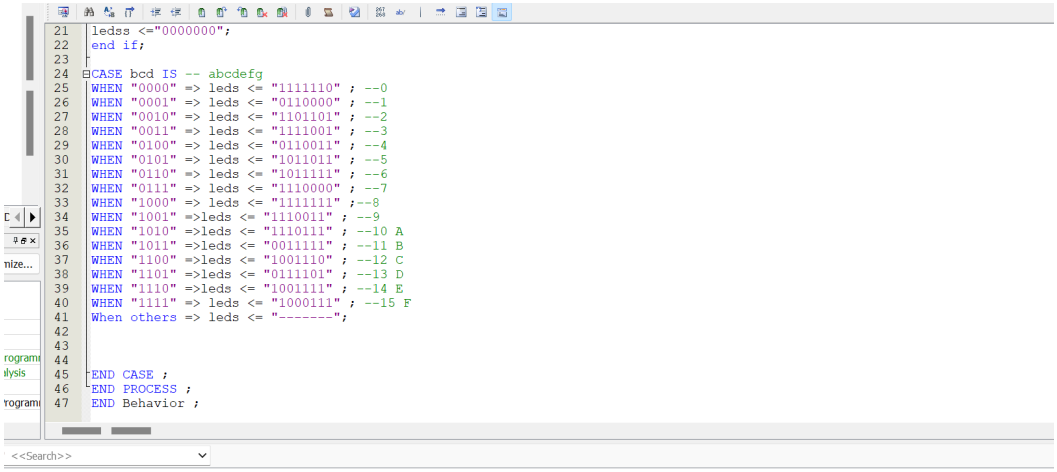
```
Info: 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Info: Full Compilation was successful. 0 errors, 56 warnings
```

SSEG



```
Library ieee;
USE ieee.std_logic_1164.all ;
ENTITY sseg IS
PORT (
 bcd : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
 neg : IN STD_LOGIC;
 leds : OUT STD_LOGIC_VECTOR(0 TO 6);
 ledss : OUT STD_LOGIC_VECTOR(0 TO 6));

END sseg;

ARCHITECTURE Behavior OF sseg IS
BEGIN
PROCESS (bcd, neg)
 BEGIN

if(neg ='1') then
 ledss <="0000001";
end if;
if (neg = '0') then
 ledss <="0000000";
 end if;

CASE bcd IS -- abcdefg
 WHEN "0000" => leds <= "1111110" ; --0
 WHEN "0001" => leds <= "0110000" ; --1
 WHEN "0010" => leds <= "1101101" ; --2
 WHEN "0011" => leds <= "1111001" ; --3
```

```
<<Search>>
s II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
s II Full Compilation was successful. 0 errors, 56 warnings
```



```
ledss <="0000000";
end if;

CASE bcd IS -- abcdefg
 WHEN "0000" => leds <= "1111110" ; --0
 WHEN "0001" => leds <= "0110000" ; --1
 WHEN "0010" => leds <= "1101101" ; --2
 WHEN "0011" => leds <= "1111001" ; --3
 WHEN "0100" => leds <= "0110011" ; --4
 WHEN "0101" => leds <= "1011011" ; --5
 WHEN "0110" => leds <= "1011111" ; --6
 WHEN "0111" => leds <= "1110000" ; --7
 WHEN "1000" => leds <= "1111111" ;--8
 WHEN "1001" =>leds <= "1110011" ; --9
 WHEN "1010" =>leds <= "1110111" ; --10 A
 WHEN "1011" =>leds <= "0011111" ; --11 B
 WHEN "1100" =>leds <= "1001110" ; --12 C
 WHEN "1101" =>leds <= "0111101" ; --13 D
 WHEN "1110" =>leds <= "1001111" ; --14 E
 WHEN "1111" => leds <= "1000111" ; --15 F
 When others => leds <= "-------";


END CASE ;
END PROCESS ;
END Behavior ;
```

```
<<Search>>

64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Full Compilation was successful. 0 errors, 56 warnings
```

Decoder

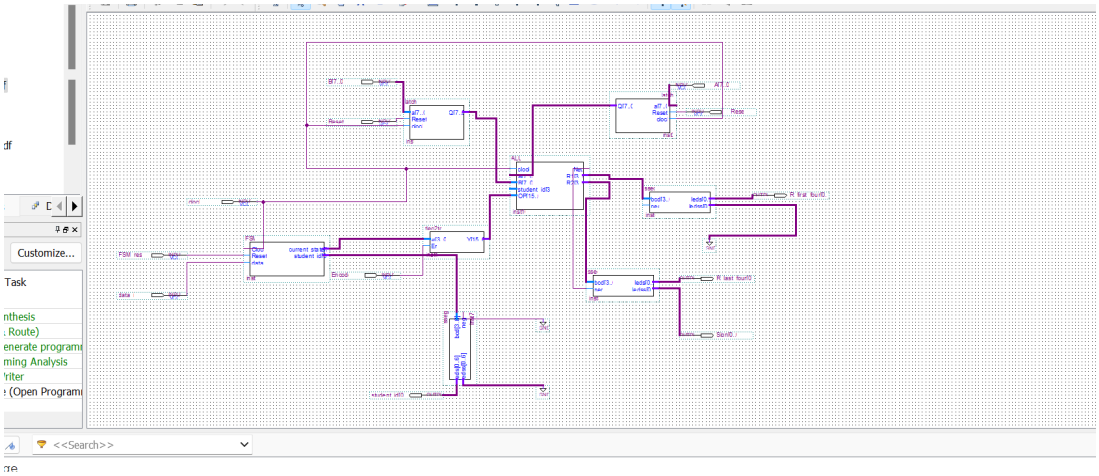
```
library ieee;
use ieee.std_logic_1164.all;

entity dec2to4 is
port(w: in std_logic_vector(3 downto 0);
    En: in std_logic;
    Y: out std_logic_vector(15 downto 0));
end dec2to4;

architecture Behavior of dec2to4 is
    signal Enw: std_logic_vector(4 downto 0);
begin
    Enw <= En & w;
    with Enw select
    y<=

 "0000000000000001" when "10000",
 "0000000000000010" when "10001",
 "0000000000000100" when "10010",
 "0000000000001000" when "10011",
 "0000000000010000" when "10100",
 "0000000000100000" when "10101",
 "0000000001000000" when "10110",
 "0000000010000000" when "10111",
 "0000001000000000" when "11000",
 "0000010000000000" when "11001",
 "0000100000000000" when "11010",
 "0001000000000000" when "11011",
```

```
II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
```

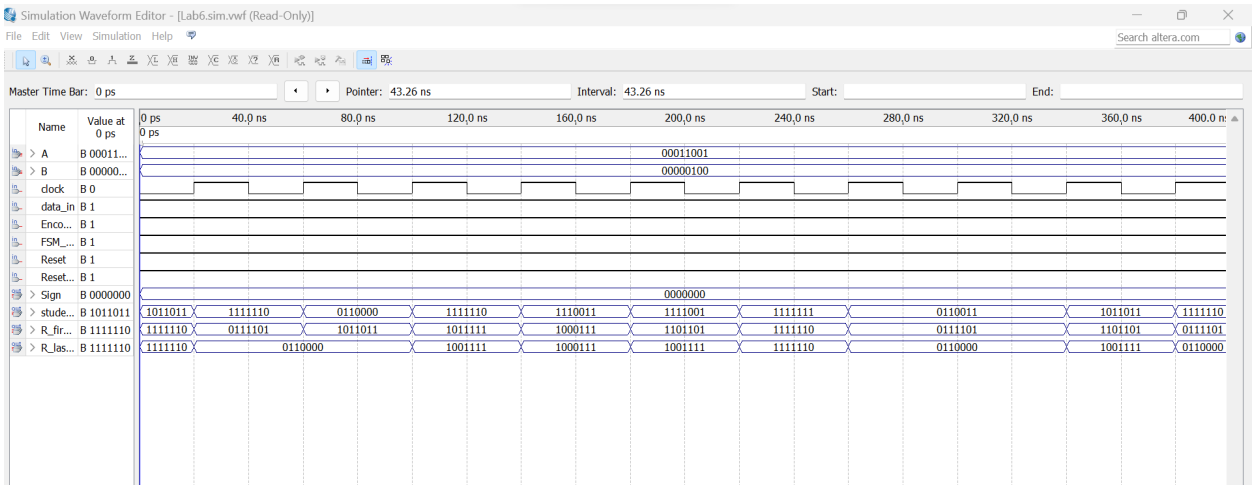
```
end dec2to4;

architecture Behavior of dec2to4 is
    signal Enw: std_logic_vector(4 downto 0);
begin
    Enw <= En & w;
    with Enw select
    y<=

 "0000000000000001" when "10000",
 "0000000000000010" when "10001",
 "0000000000000100" when "10010",
 "0000000000001000" when "10011",
 "0000000000010000" when "10100",
 "0000000000100000" when "10101",
 "0000000001000000" when "10110",
 "0000000010000000" when "10111",
 "0000001000000000" when "11000",
 "0000010000000000" when "11001",
 "0000100000000000" when "11010",
 "0001000000000000" when "11011",
 "0010000000000000" when "11100",
 "0100000000000000" when "11101",
 "1000000000000000" when "11110",
 "0000000000000000" when "11111",
 "0000000000000000" when others;
end Behavior;
```

```
64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Full Compilation was successful. 0 errors, 56 warnings
```

Block Diagram



Waveform



For the first part we will first start by assigning A and B. We will get the last 4 digits of my student number which is 0425, 04 is A and 25 is B. Then by getting out latch code which includes the reset and the clock, and then run it and see if it works. The next code we will be getting is the same sseg code that we used in the past labs. As we know sseg will give us the number in binary in a seven segment. We will also use a decoder code that we used in the previous labs as well. We will then create an ALU code that we can copy from the lab manual. Lastly the last code that we will use is the FSM which we used in the previous lab. We will have to change the code up a bit by changing the states and change to starting by 0. Once we run all the codes we will then create our block diagram that is seen in the lab manual. Once we run that we will get our waveform.

Function #	Microcode	Boolean Operation / Function
1	0000000000000001	sum(A, B)
2	0000000000000010	diff(A, B)
3	0000000000000100	$\overline{A}$
4	0000000000001000	$\overline{A \cdot B}$
5	0000000000010000	$\overline{A + B}$
6	0000000000100000	$A \cdot B$
7	0000000001000000	$A \oplus B$
8	0000000010000000	$A + B$
9	0000000100000000	$\overline{A \oplus B}$

Table 1. ALU Core Operations for Problem 1

$A = (04)_{16}$ $B = (25)_{16}$

$A = (00000100)$ $B = 00100101$

29 → 1D

Addition 29 ↓ $(1D)_{16}$

Subtraction −21 ↓ $(15)_{16}$

3) Invert A

$255 - 25 = 230$ decimal ↓ E6

4) NAnd

$04 = 25 \Rightarrow$ False = 0

$255 - 0 = 255$ decimal ↓ FF Hex

5) $\overline{A+B}$

$04 + 25 = 29$ decimal

$255 - 29 = 226$ decimal

$= (E2)_{16}$

6) And

$04 = 25 \Rightarrow$ false

$(00)_{16}$

(8) W

True = 1 = 29

$= (1D)_{16}$

7) $A \oplus B$ True = 1

$04 \oplus 25 \Rightarrow (1D)_{16}$

9) Xnor ⇒ fake = 0

$255 - (04 + 25)$

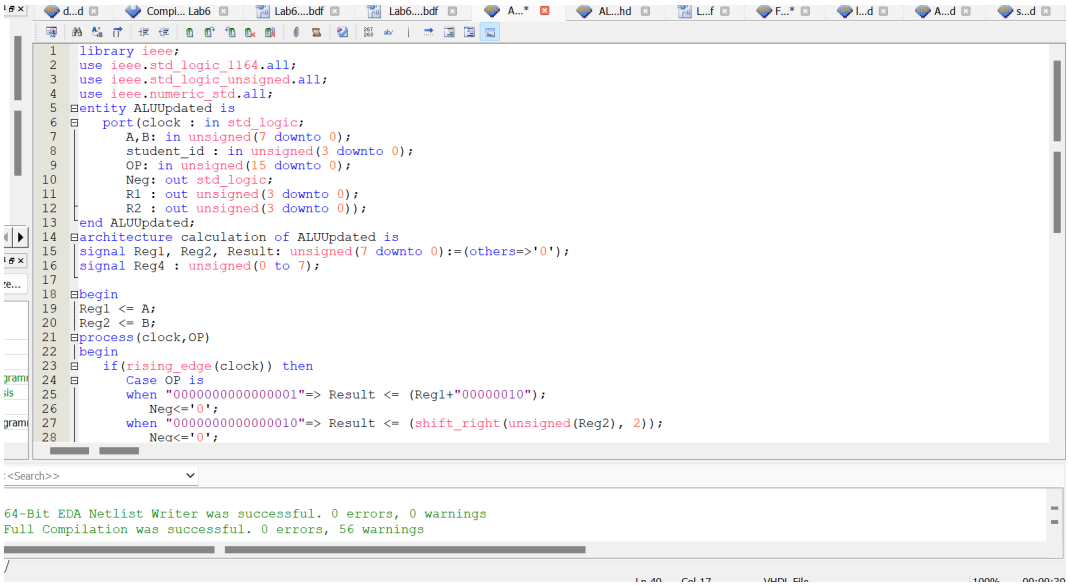
$= 226$

$= E2$

The next step of part 1 is to see if our waveform is right. In the lab manual of table 1 it says what the product of our A and B will look like. Such as the first one will be the sum between the two and the second will be the subtraction. In my work above we can see that the addition of the two is 29, and the hexadecimal of that is 1D. As we can see in our waveform the first cycle the bottom shows the 1 as a seven segment and the top displays the D. Same for the next step, when you subtract each other you will get 21, in hexadecimal you get 15. We can see in the waveform it is still 1 on the bottom and the top shows 5 in a seven segment. If we carry on with my work the rest will work out.

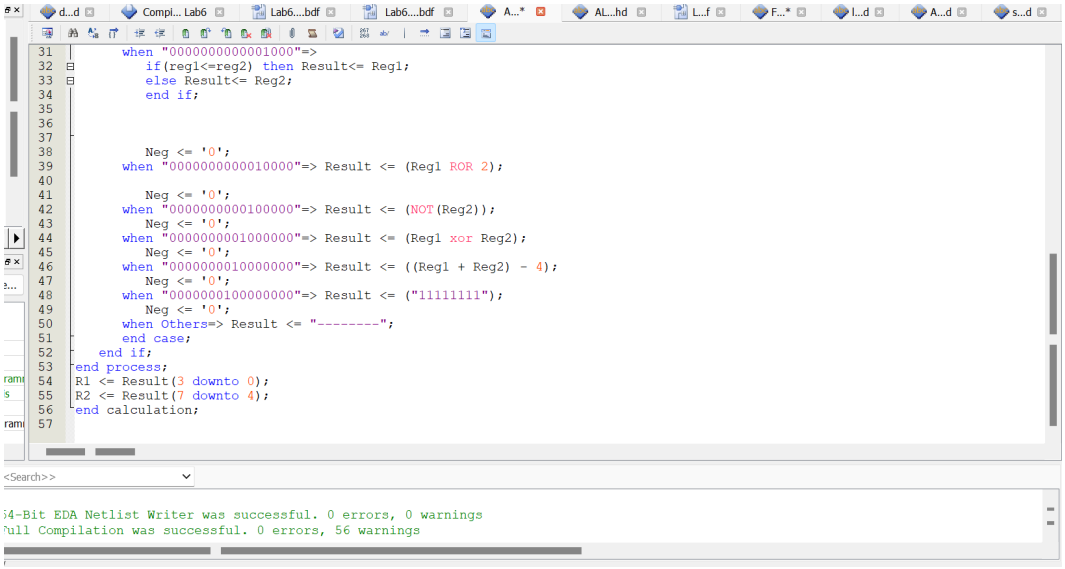
PART 2

ALU



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
entity ALUUpdated is
   port(clock : in std_logic;
        A,B: in unsigned(7 downto 0);
        student_id : in unsigned(3 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 ALUUpdated;
architecture calculation of ALUUpdated 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"=> Result <= (Reg1+"00000010");
         Neg<='0';
      when "0000000000000010"=> Result <= (shift_right(unsigned(Reg2), 2));
         Neg<='0';
```

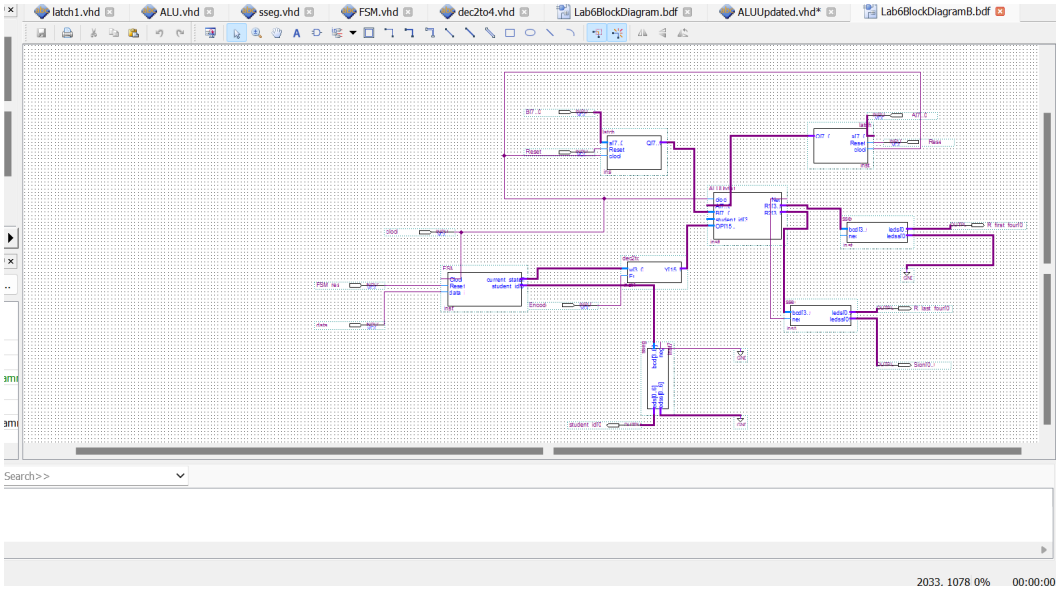
```
64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Full Compilation was successful. 0 errors, 56 warnings
```



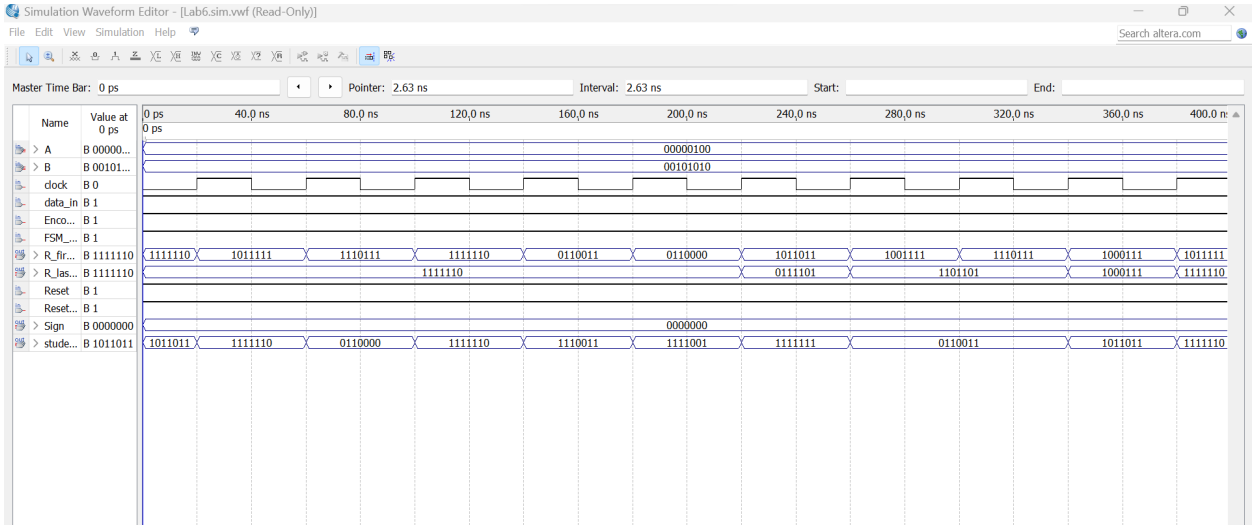
```
      when "0000000000001000"=>
         if(reg1<=reg2) then Result<= Reg1;
         else Result<= Reg2;
         end if;



         Neg <= '0';
      when "0000000000010000"=> Result <= (Reg1 ROR 2);

         Neg <= '0';
      when "0000000000100000"=> Result <= (NOT(Reg2));
         Neg <= '0';
      when "0000000001000000"=> Result <= (Reg1 xor Reg2);
         Neg <= '0';
      when "0000000010000000"=> Result <= ((Reg1 + Reg2) - 4);
         Neg <= '0';
      when "0000000100000000"=> Result <= ("11111111");
         Neg <= '0';
      when Others=> Result <= "--------";
      end case;
   end if;
end process;
R1 <= Result(3 downto 0);
R2 <= Result(7 downto 4);
end calculation;
```

```
64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Full Compilation was successful. 0 errors, 56 warnings
```

Block Diagram



Waveform



For this next part we will just have to change the ALU code to match the table that was given in the lab manual. I chose option table A so I changed the code to match that. Once we get that to run we will update our block diagram and get a new display for the waveform.

a)

Function #	Operation / Function
1	Increment **A** by 2
2	Shift **B** to right by two bits, input bit = 0 (SHR)
3	Shift **A** to right by four bits, input bit = 1 (SHR)
4	Find the smaller value of **A** and **B** and produce the results (Min(**A**,**B**))
5	Rotate **A** to right by two bits (ROR)
6	Invert the bit-significance order of **B**
7	Produce the result of XORing **A** and **B**
8	Produce the summation of **A** and **B**, then decrease it by 4
9	Produce all high bits on the output

Par 2 I chose A) A=04 B=25

A=00000100 B=00100101

1) 04+2 = 06 dec

↓

6 hex

2) shift B right 2 bits

00001001 = 09

3) shift A right by 4 bits

00000000 = 00

4) min(A,B) = 4 dec

= $(04)_{16}$

5) Rotate A right by 2 bits

00000001 $(1)_{16}$

6) Invert B

11011010

D5

7) A XOR B

29

$(10)_{16}$

8 A+B-4

= 4+25-4

= 25

= $(19)_{16}$

9) "11111111"

FF

Now we will see if our waveform matches the table. Above we can see the calculations I did for the table. First thing it asks is to increase the increment A by 2 which I did, that gives us 06 in hexadecimal. As we can see in the waveform the first circle displays a 6 in a seven segment. The next part is shift B to the right by two bits, doing that we get 00001001 which is 9 hexadecimal. In the waveform the next cycle is 9 in a seven segment so it is correct, the next are correct as well.

PART 3

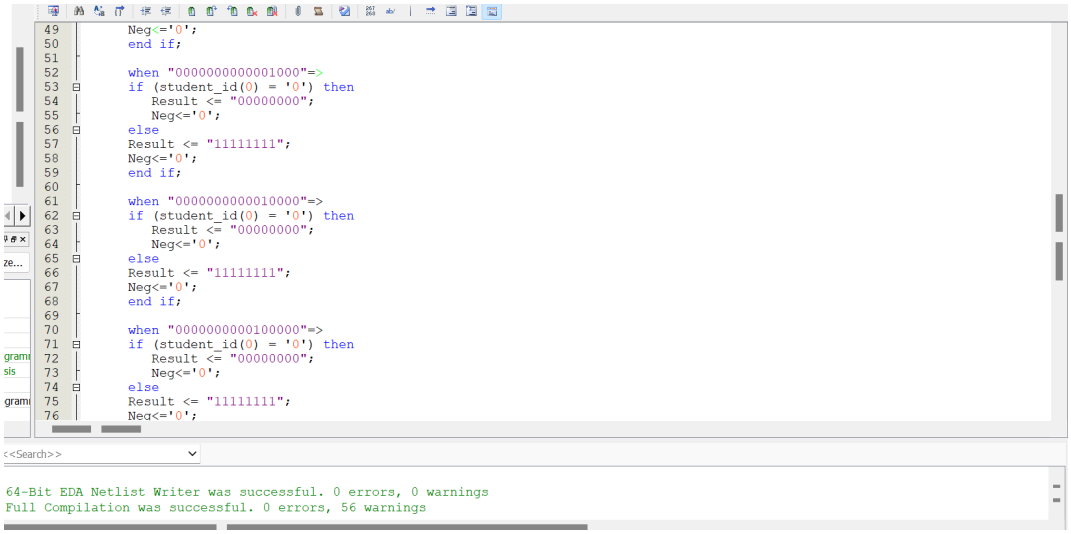
ALU

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
entity ALUUPDATEDC is
   port(clock : in std_logic;
        A,B: in unsigned(7 downto 0);
        student_id : in unsigned(3 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 ALUUPDATEDC;
architecture calculation of ALUUPDATEDC 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"=>
      if (student_id(0) = '0') then
         Result <= ("00000000");
         Neg<=('0');
```

```
I 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
I Full Compilation was successful. 0 errors, 56 warnings
```

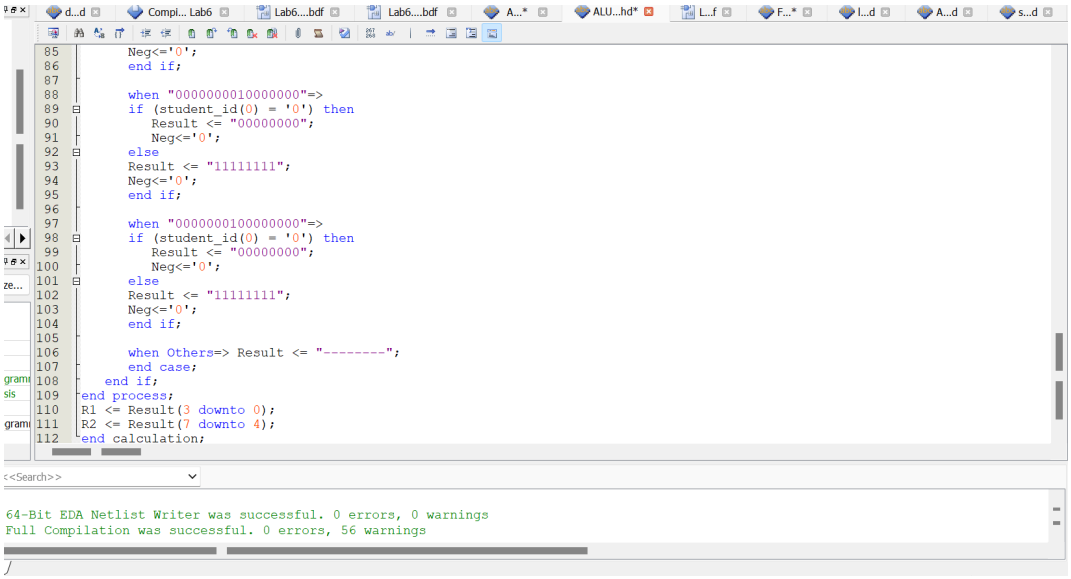
```
         Result <= ("00000000");
         Neg<=('0');
      else
      Result <= "11111111";
      Neg<='0';
      end if;

      when "0000000000000010"=>
      if (student_id(0) = '0') then
         Result <= "00000000";
         Neg<='0';
      else
      Result <= "11111111";
      Neg<='0';
      end if;

      when "0000000000000100"=>
      if (student_id(0) = '0') then
         Result <= "00000000";
         Neg<='0';
      else
      Result <= "11111111";
      Neg<='0';
      end if;

      when "0000000000001000"=>--complete 4 if one is greater than the other
      if (student_id(0) = '0') then
         Result <= "00000000";
```

```
64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
```



```
        Neg<='0';
        end if;

        when "0000000000001000"=>
        if (student_id(0) = '0') then
           Result <= "00000000";
           Neg<='0';
        else
        Result <= "11111111";
        Neg<='0';
        end if;

        when "0000000000010000"=>
        if (student_id(0) = '0') then
           Result <= "00000000";
           Neg<='0';
        else
        Result <= "11111111";
        Neg<='0';
        end if;

        when "0000000000100000"=>
        if (student_id(0) = '0') then
           Result <= "00000000";
           Neg<='0';
        else
        Result <= "11111111";
        Neg<='0';
```

```
64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Full Compilation was successful. 0 errors, 56 warnings
```



```
        Neg<='0';
        end if;

        when "0000000010000000"=>
        if (student_id(0) = '0') then
           Result <= "00000000";
           Neg<='0';
        else
        Result <= "11111111";
        Neg<='0';
        end if;

        when "0000000100000000"=>
        if (student_id(0) = '0') then
           Result <= "00000000";
           Neg<='0';
        else
        Result <= "11111111";
        Neg<='0';
        end if;

        when Others=> Result <= "--------";
        end case;
    end if;
end process;
R1 <= Result(3 downto 0);
R2 <= Result(7 downto 4);
end calculation;
```

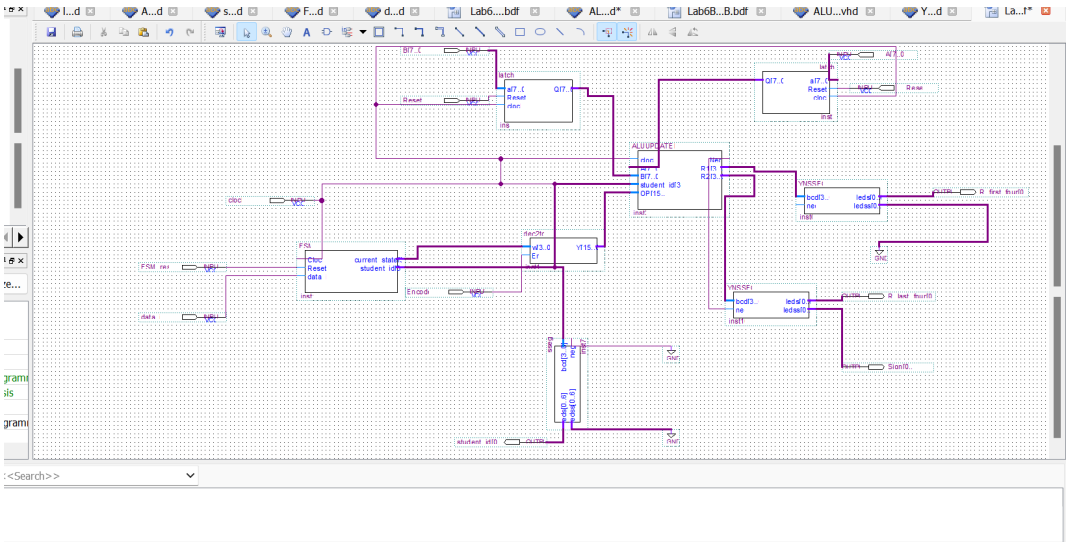
```
64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Full Compilation was successful. 0 errors, 56 warnings
```

SSEG

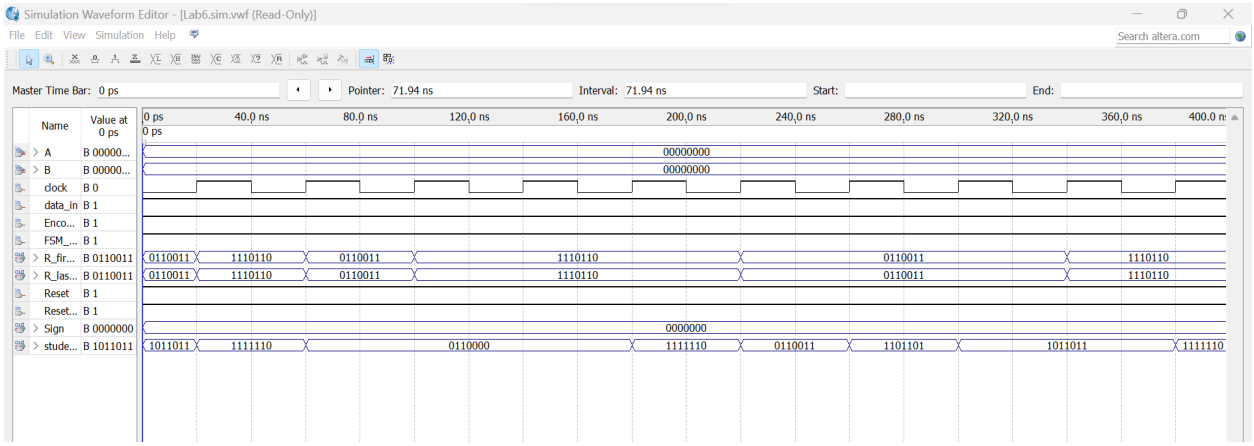
```
1  Library ieee;
2  USE ieee.std_logic_1164.all ;
3  ENTITY YNSSEG IS
4  PORT (
5  bcd : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
6  neg : IN STD_LOGIC;
7  leds : OUT STD_LOGIC_VECTOR(0 TO 6);
8  ledss : OUT STD_LOGIC_VECTOR(0 TO 6));
9
10 END YNSSEG;
11
12 ARCHITECTURE Behavior OF YNSSEG IS
13 BEGIN
14 PROCESS (bcd, neg)
15 BEGIN
16
17 if(neg ='1') then
18 ledss <="0000001";
19 end if;
20 if (neg = '0') then
21 ledss <="0000000";
22 end if;
23
24 CASE bcd IS -- abcdefg
25 WHEN "0000" => leds <= "0110011" ; -- Y
26 WHEN "1111" => leds <= "1110110" ; -- N
27 When others => leds <= "-------";
28
```

```
:Search>>

4-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
ull Compilation was successful. 0 errors, 56 warnings
```

Block Diagram



Waveform



The next part I will have to change the ALU code to match the next step which is a)(seen below). I will also have to change the sseg code to match it as well. Then I will run it and update the block diagram, and get the waveform.

your TA. Some modifications to the 7 segment may also be needed to display “y” or “n”. The TA shall assign one of the following problems for each student.

a) For each microcode instruction, display 'y' if the FSM output (**student_id**) is odd and 'n' otherwise

Now it is time to see if our final waveform is correct. What should display is your student number and the option I chose should change the cycle from an odd and even number. My student number is 501110425. We can see the 5 is there by itself, the next number is 0 which is registered as even that's why it separates. The next three numbers are 1 which are all odd, that's why we can see three segments together. Then the next three numbers are 042 which are all even, hence we see three segments by themselves. Lastly is it number 5 and we see that by itself, this proves the waveform to be correct.