Ryerson University Faculty of Science

Department of Physics

Course Number	*COE 328*
Course Title	*Digital Circuits*
Semester/Year	*Fall 2021*
Instructor	*Dr. Reza Sedaghat*
TA Name	*Sajjad Rostami Sani*

Lab/Tutorial Report No.	*3*

Report Title	*Lab 3*

Section No.	*03*
Submission Date	*31-10-2021*
Due Date	*01-11-2021*

Student Name	Student ID	Signature*
Ahmad El-Gohary	501011852	

**By signing above, you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at:*

http://www.ryerson.ca/content/dam/senate/policies/pol60.pdf

Objective

The objective of this lab was to build a simple adder and subtractor circuit with a 7-segment display that takes 2 binary numbers in binary form and outputs their sum in the hexadecimal numerical system for part 1. For part 2 the circuit was modified to have an encoder which will display the respective digit of my student number, for example adding 2+3 is equal 5 so the circuit would output the 5^{th} digit of my student number.

[This part of page is left blank intentionally]

Procedure

Part 1:

VHDL code for adder and subtractor unit:

```
1   LIBRARY ieee ;
2   USE ieee.std_logic_1164.all ;
3   USE ieee.std_logic_unsigned.all ;
4
5   ENTITY ASU IS
6       PORT ( Cin : IN STD_LOGIC ;
7           X, Y : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
8           S : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) ;
9           Cout, neg : OUT STD_LOGIC ) ;
10  END ASU ;
11
12  ARCHITECTURE Behavior OF ASU IS
13      SIGNAL Sum : STD_LOGIC_VECTOR(4 DOWNTO 0) ;
14  BEGIN
15      PROCESS (Cin)
16      BEGIN
17      IF (Cin = '1') THEN
18          Sum <= ('0' & X) - ('0' & Y);
19          S <= Sum(3 DOWNTO 0) ;
20          Cout <= Sum(4) ;
21          neg <= '1' ;
22
23      ELSE
24          Sum <= ('0' & X) + ('0' & Y);
25          S <= Sum(3 DOWNTO 0) ;
26          Cout <= Sum(4) ;
27          neg <= '0' ;
28
29          END IF;
30          END PROCESS;
31
32  END Behavior ;
```

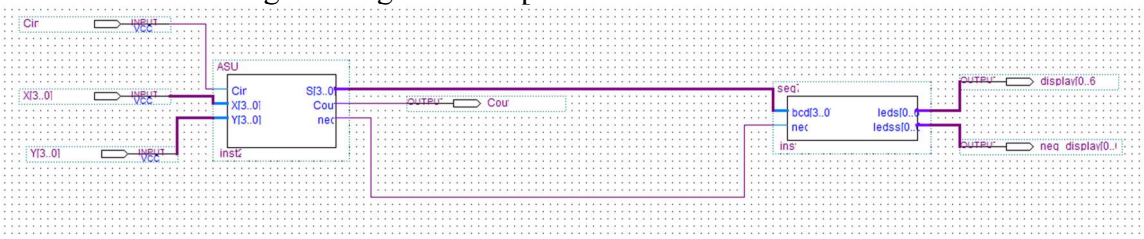
The code was copied from the textbook from figure 5.28. The only changes that were made to the code were the if and else statements from lines 14 to 30 to show a negative sign on the g segment of a second 7-segment display if the sum of the 2 numbers is negative to indicate that.

VHDL code for the 7-segment display:

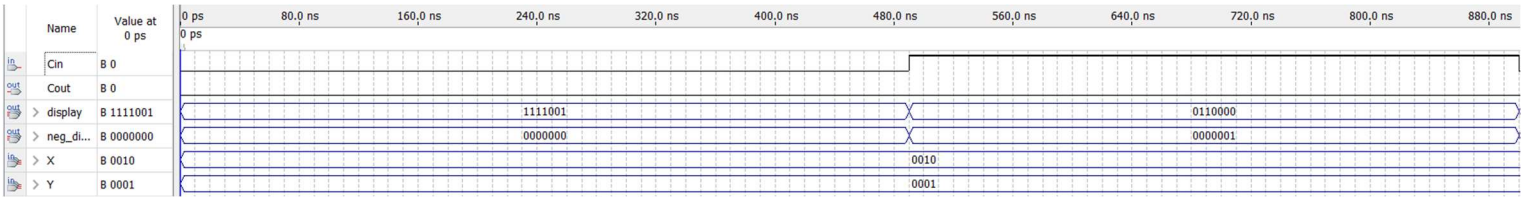
```
1  LIBRARY ieee ;
2  USE ieee.std_logic_1164.all ;
3
4  ENTITY seg7 IS
5      PORT ( bcd : IN STD_LOGIC_VECTOR(3 DOWNTO 0) ;
6           neg : IN STD_LOGIC ;
7         leds : OUT STD_LOGIC_VECTOR(0 TO 6);
8         ledss: OUT STD_LOGIC_VECTOR(0 TO 6) ) ;
9  END seg7 ;
10
11 ARCHITECTURE Behavior OF seg7 IS
12 BEGIN
13     PROCESS ( bcd )
14     BEGIN
15         CASE bcd IS -- abcdefg
16             WHEN "0000" => leds <= "1111110";
17             WHEN "0001" => leds <= "0110000";
18             WHEN "0010" => leds <= "1101101";
19             WHEN "0011" => leds <= "1111001";
20             WHEN "0100" => leds <= "0110011";
21             WHEN "0101" => leds <= "1011011";
22             WHEN "0110" => leds <= "1011111";
23             WHEN "0111" => leds <= "1110000";
24             WHEN "1000" => leds <= "1111111";
25             WHEN "1001" => leds <= "1110011";
26
27             WHEN "1010"    => leds <= "1110111"; --a
28             WHEN "1011"    => leds <= "0011111"; --b
29             WHEN "1100"    => leds <= "1001110"; --c
30             WHEN "1101"    => leds <= "0111101"; --d
31             WHEN "1110"    => leds <= "1101111"; --e
32             WHEN "1111"    => leds <= "1000111"; --f
33
34         END CASE ;
35     END PROCESS ;
36
37 PROCESS (neg)
38     BEGIN
39     IF (neg = '1') THEN
40     ledss <= "0000001";
41
42     ELSE
43     ledss <= "0000000";
44
45     END IF;
46     END PROCESS;
47 END Behavior ;
```

The code was copied from the textbook from figure 6.47 and was modified to have lines from 27 to 32 to display the sum of the 2 binary numbers in hexadecimal form showing A for number 10 lowercase b for 11 C for 12 lower case d for 13 E for 14 and F for 15. It was also modified to have a second process to turn on the g segment of a second 7-segment display to indicate that a number is negative when the carry in is 1.

Compiling both codes and generating a block component of each code and connecting them as the following circuit gives a simple adder subtractor circuit.



Compiling the circuit then running it in a wave function file to test if it is running correctly.



Adding Y and X which are “1” and “2” in binary form results in a “3” displayed on the first 7-segment display (lights up the a b c d and g segments) and nothing on the second since it is a positive number but when Y-X is implemented (Carry in is 1) the sum is “-1” so the process results in a “1” on the first 7 segment display (lights up the b and c segments) and a “–” on the second display since it is a negative number (lights up the g segment).

Part 2:
VHDL code for the encoder/combinatorial block circuit:

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.all;
3   USE ieee.std_logic_unsigned.all;
4
5   ENTITY C IS
6   PORT (
7   S : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
8   L : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
9   END C;
10
11  ARCHITECTURE Behaviour OF C IS
12  BEGIN
13      PROCESS (S)
14      BEGIN
15      L(0) <=((Not S(2) and Not S(1) and Not S(0))
16              Or
17              (Not S(3) and Not S(2) and S(0))
18              Or
19              (S(2) and S(1) and Not S(0))
20              Or
21              (Not S(3) and Not S(1) and S(0)));
22
23      L(1) <=(S(3) and S(0));
24
25      L(2) <=((S(3) and Not S(0) )
26              Or
27             (Not S(3) and Not S(2) and Not S(1) and S(0)));
28
29      L(3) <=(S(2) and S(1) and S(0));
30      END PROCESS;
31  END Behaviour;
```

L0 L1 L2 and L3 are resultant logical functions from the k-maps of the following table which's outputs is based on my student number's digits in binary system.

S (ASU Output)				STUDENT_ID (C Output)			
S_3	S_2	S_1	S_0	L_3	L_2	L_1	L_0
0	0	0	0	0	0	0	1
0	0	0	1	0	1	0	1
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	0	1	0
X	X	X	X	1	1	1	0
X	X	X	X	1	1	1	0
X	X	X	X	1	1	1	0
X	X	X	X	1	1	1	0

Fill your Student ID in Binary form: 5 0 1 0 1 1 8 5 2

Don't Cares

A truth table and a k-map is made for each output L(n) and then simplified to a logical function.

S3	S2	S1	S0	L0
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0

(Not S2 and Not S1 and Not S0) Or (Not S3 and Not S2 and S0) or (S2 and S1 and Not S0) Or (Not S3 and Not S1 and S0)

S3	S2	S1	S0	L1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1

S3 and S0

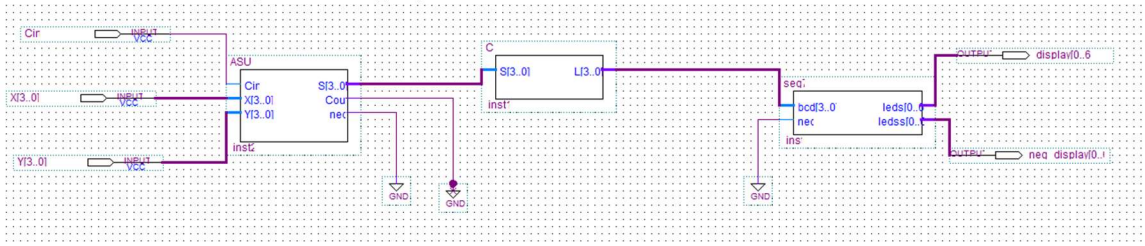
S3	S2	S1	S0	L2
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0

(S3 and Not S0) Or (Not S3 and Not S2 and Not S1 and S0)

S3	S2	S1	S0	L3
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0

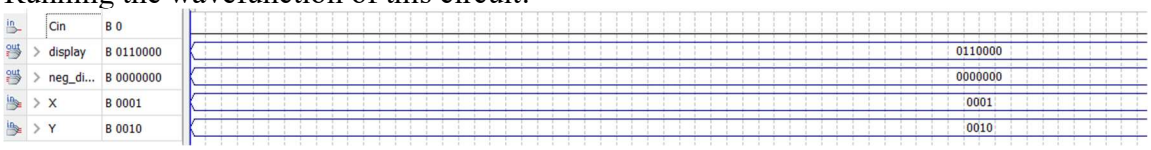
S(2) and S(1) and S(0)

After compiling the code for this block circuit, the circuit is simple. It is very similar to the circuit from part 1 but with the encoder/combinatorial block circuit in between the ASU and the 2 7-segment displays.



The carry out and negative terminals are grounded because a position of a digit cannot be negative so these terminals should always carry a “0” signal since they must always be positive.

Running the wavefunction of this circuit:



Adding Y and X which are “1” and “2” in binary form is equal “3” which then goes through the encoder and displays the third number of my student number on the 7-segment display which is 1 lighting up the b and c segments and since it cannot be a negative number the second display shows nothing.

References

Brown, S. D., & Vranesic, Z. G. (2009). *Fundamentals of Digital Logic with VHDL Design*. New York, United States: McGraw-Hill Education.