LAB 4 TUTORIAL

ADDER AND SUBTRACTOR UNIT

OVERVIEW

In this lab we will:

1. Use VHDL code to design a 4-bit Adder/Subtractor Unit (ASU) that multiplexes add and subtract operations with a common Cin input.
2. Design a combinatorial circuit that takes the decoded output of the ASU as input, and consequently outputs individual digits of your student number.

PROCEDURE

1. 4-bit (Adder/Subtractor Unit) ASU

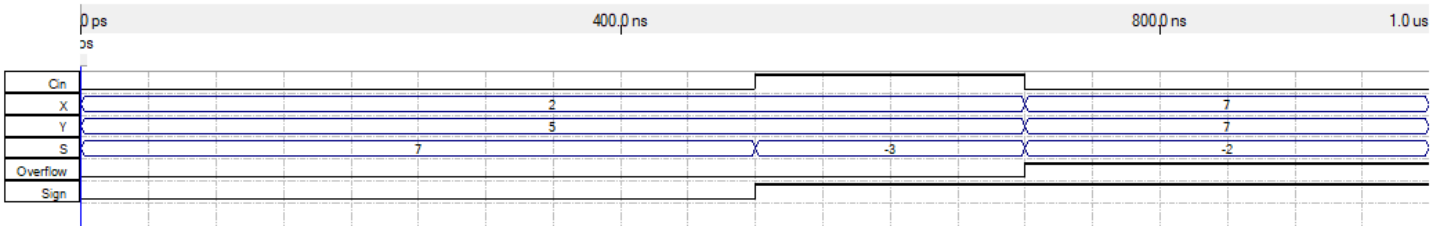
Please follow the following instructions to implement the 4-bit ASU:

1. Create a new folder “Lab4” in your “BME328” folder.
2. Open the Quartus II software, and using the new project wizard, create a new project “Lab4” in your “Lab4” folder.

   MAKE SURE THAT YOU CHOOSE THE “EP2C35F672C6” DEVICE IN THE PROJECT WIZARD.
3. Create a new VHDL file in your “Lab4” project (File > New > VHDL File).
4. Type the following in the Text Editor and save file as “ASU.vhd”:

```
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, Overflow, Sign      :   out std_logic);
10  end ASU;
11
12  architecture Behavior of ASU is
13      signal Sum  :   std_logic_vector(4 downto 0);
14      signal Yp   :   std_logic_vector (3 downto 0);
15  begin
16      -- If Cin = 0, ASU Performs ADD. Yp = Y
17      -- If Cin = 1, ASU Performs SUB. Yp = not Y to generate 2's complement -Y.
18      Yp(3) <= Y(3) xor Cin;
19      Yp(2) <= Y(2) xor Cin;
20      Yp(1) <= Y(1) xor Cin;
21      Yp(0) <= Y(0) xor Cin;
22
23      Sum <= ('0' & X) + ('0' & Yp) + Cin;
24      S <= Sum(3 downto 0);
25      Cout <= Sum (4);
26      Overflow <= Sum(4) xor X(3) xor Yp(3) xor Sum(3);
27      Sign <= Sum(3);
28  end Behavior;
29
```

5. Set “ASU.vhd” as the top-level entity. You can do this by right-clicking on “ASU.vhd” in the “Files” section of the project navigator (located at the right of your Quartus II window) and selecting “Set at Top-Level Entity”.
6. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
7. Create a symbol for your “ASU.vhd” file. You can do this by right clicking on “ASU.vhd” in the “Files” section of the project navigator (located at the right of your Quartus II window) and selecting “Create Symbol Files for Current File”.
8. Create a new University Program VWF (File > New > University Program VWF).
9. Simulate “ASU.vhd” and make sure that your simulation results match the results shown below.
10. Save the VWF file as “ASU.vwf” in your “Lab4” folder and take a screenshot of your results.



2. Combinatorial Unit C

In this part of the lab we are required to create a Combinatorial Unit that takes as input the output S from the ASU and consequently outputs individual digits of your student number.

I will use my student number as an example: 500414487

S = 0 corresponds to the first digit, in my case it is 5.

S= 1 corresponds to the second digit in my case it is 0.

And so on until we get to

S = 8 which corresponds to the 9th digit, in my case it is 7.

For all other values of S, we want to display 0.

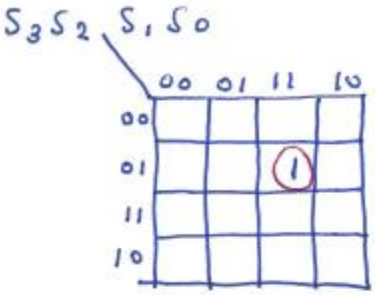
Based on this please follow the following instructions to complete the lab:

1. fill out the truth table given the lab manual with your student ID. Below is an example using my using.

ASU Sum Output (4-bit Binary)	**Student ID (4-bit Binary)**	**Student ID (Decimal)**
S_3 S_2 S_1 S_0	**L_3 L_2 L_1 L_0**	
0 0 0 0	0 1 0 1	5
0 0 0 1	0 0 0 0	0
0 0 1 0	0 0 0 0	0
0 0 1 1	0 1 0 0	4
0 1 0 0	0 0 0 1	1
0 1 0 1	0 1 0 0	4
0 1 1 0	0 1 0 0	4
0 1 1 1	1 0 0 0	8
1 0 0 0	0 1 1 1	7
1 0 0 1	0 0 0 0	0
1 0 1 0	0 0 0 0	0
1 0 1 1	0 0 0 0	0
1 1 0 0	0 0 0 0	0
1 1 0 1	0 0 0 0	0
1 1 1 0	0 0 0 0	0
1 1 1 1	0 0 0 0	0

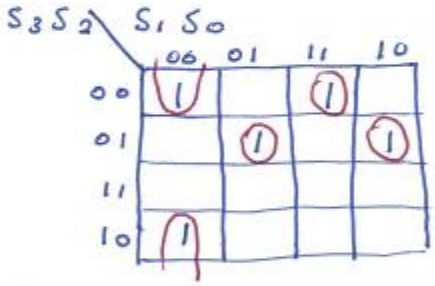
2. Minimize the outputs L3, L2, L1, L0 in your truth table using K-maps. Below is an example using my student ID.

L_3 :



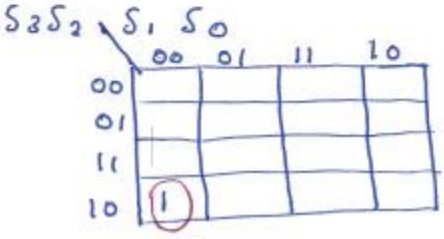
$L_3 = \overline{S_3}\, S_2\, S_1\, S_0$.

L_2 :



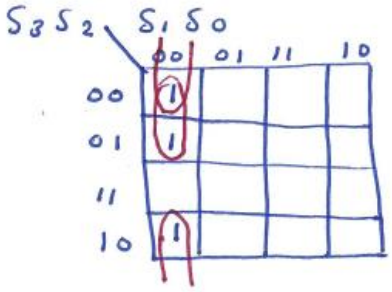
$L_2 = \overline{S_2}\,\overline{S_1}\,\overline{S_0} + \overline{S_3}\,S_2\,\overline{S_1}\,S_0 + \overline{S_3}\,\overline{S_2}\,S_1\,S_0 + \overline{S_3}\,S_2\,S_1\,\overline{S_0}$.

L_1 :



$L_1 = S_3\,\overline{S_2}\,\overline{S_1}\,\overline{S_0}$.

L_0:



$$L_0 = \overline{S_3}\,S_1\,\overline{S_0} + \overline{S_2}\,\overline{S_1}\,\overline{S_0}$$

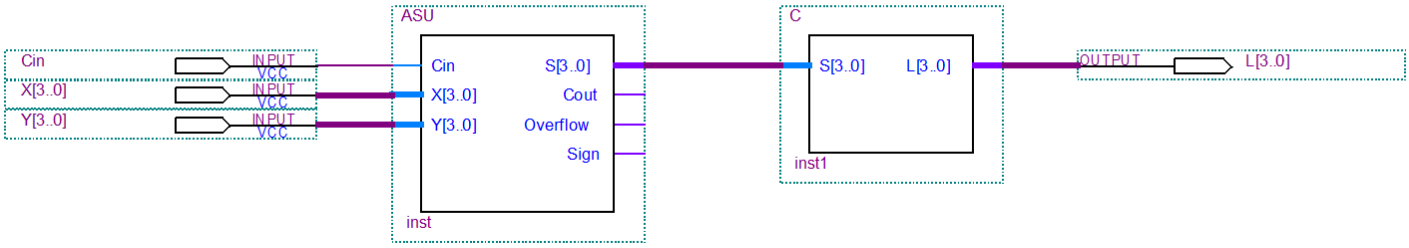
3. Use the VHDL code below to implement your Combinatorial Unit C. Remember to replace the L(3), L(2), L(1), and L(0) output assignment functions in the code with your minimized functions.
4. Save the VHDL file as “C.vhd”.

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity C is
5   port(
6       S   :   in      std_logic_vector(3 downto 0);
7       L   :   out     std_logic_vector(3 downto 0)
8       );
9   end C;
10
11  architecture Behavior of C is
12  begin
13      L(3) <= (not S(3) and     S(2) and     S(1) and     S(0));
14
15      L(2) <= (             not S(2) and not S(1) and not S(0)) or
16              (not S(3) and     S(2) and not S(1) and     S(0)) or
17              (not S(3) and not S(2) and     S(1) and     S(0)) or
18              (not S(3) and     S(2) and     S(1) and not S(0));
19
20      L(1) <= (    S(3) and not S(2) and not S(1) and not S(0));
21
22      L(0) <= (not S(3) and                 not S(1) and not S(0)) or
23              (             not S(2) and not S(1) and not S(0));
24
25  end Behavior;
26
27
```

6. Set “C.vhd” as the top-level entity. You can do this by right-clicking on “C.vhd” in the “Files” section of the project navigator (located at the right of your Quartus II window) and selecting “Set at Top-Level Entity”.
7. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
8. Create a symbol for your “C.vhd” file. You can do this by right clicking on “C.vhd” in the “Files” section of the project navigator (located at the right of your Quartus II window) and selecting “Create Symbol Files for Current File”.
9. Create a new University Program VWF (File > New > University Program VWF).
10. Simulate “C.vhd” and make sure that your simulation results match your student number.
11. Save the VWF file as “C.vwf” in your “Lab4” folder and take a screenshot of your results.

Finally, we need to connect the ASU and Combinatorial Unit C:

1. Create a new Block Diagram/Schematic File in your “Lab4” project (File > New > Block Diagram/Schematic File.
2. Build the circuit shown below.
3. Save your file as “CombinedASU1.bdf”



4. Set “CombinedASU1.bdf” as the top-level entity. You can do this by right-clicking on “CombinedASU1.bdf” in the “Files” section of the project navigator (located at the right of your Quartus II window) and selecting “Set at Top-Level Entity”.
5. Start the compiler. Fix any errors and re-compile. Once the compiler compiles without any errors, move to the next step.
6. Create a new University Program VWF (File > New > University Program VWF).
7. Simulate “CombinedASU1.bdf” and make sure that your simulation results match your student number.
8. Save the VWF file as “CombinedASU1.bdf” in your “Lab4” folder and take a screenshot of your results.