Lab 3: Designing a 4-bit Adder with Overflow Detector

Sign-Off Sheet

Student 1: _____

Student 2: _____

YOU ARE RESPONSIBLE TO COMPLETE ALL THE ASSIGNMENTS IN THE CHECK LIST BELOW IN ORDER TO GET FULL CREDIT FOR THE LAB…

Check List	
Assignments	**TA Sign-off**
Pre-Lab (**MUST** be completed before the start of the lab) Watch Tutorial(s): https://youtu.be/ql3llzXIqVM | https://youtu.be/k7Y_Mejmid4 Using Logisim: **1.** Derive the logic expression for 7-segment display **2.** Derive the logic expression for 1-bit full adder **3.** Derive the logic expression for overflow detection	
Lab part **Create, design an one-bit full adder and add a constraint file** Generating a Test-bench Waveform for Functional Verification **Create, design an 4-bit 2's Complement Adder** Create a constraint file Generate bitfile and implement the logic on Board Output 4-bit Adder to 7-segment display	
Project File(s) Upload on CANVAS (Import all files in one folder, zip it, upload it) 1. Upload Sign-off Sheet 2. Upload your project files (zip the folder), name your file as: lastname_labn_D'20 **Present your work** **3.** Record a 2-3 minute video showing that all parts of your lab functioned properly. **OR** Show your work to TAs for the sign-off over Zoom. **4.** Upload the Writing Assignment (Debugging and Troubleshooting errors FAQ's) – OPTIONAL	

Due Date: 04/29/2020

Pre-lab

1. Derive the logic expression for 7-segment display

Seven segment displays are commonly used as alphanumeric displays by logic and computer systems. A seven segment display is an arrangement of 7 LEDs (Fig. 1) that can be used to show the hex digit for any number between $(0000)_2$ and $(1111)_2$ by illuminating combinations of these LEDs. In most cases all LED's in a seven segment display will have common cathode. To illuminate an LED segment you will assert a logic level on its input. For example, segments *a, b, c, d, e* and *f* must be lit to display a 0 and only segments *b* and *c* are lit to display 1.

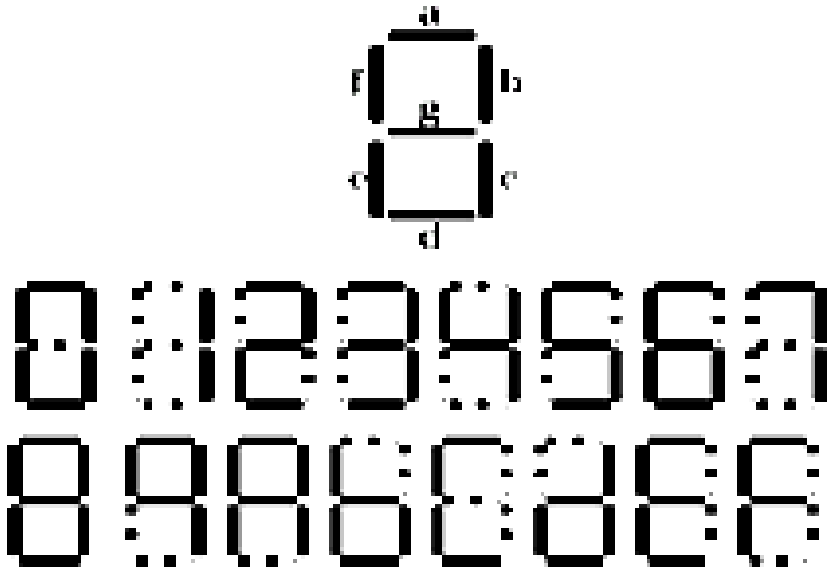

Figure 1. Seven segment display of hex number 0~F

Finish implementing this truth table (see below) for a hexadecimal (0-F) to 7-segment display decoder. This circuit block has 4 inputs and 7 outputs. The 4 inputs D3 D2 D1 D0 can be any of the binary codes for 0-F hex. The output are the logic levels for the segments *a, b, c, d, e, f, g* that need to be lit to display the hex digit. *HOWEVER*, the 7-segment displays on the Basys 3 boards and **"active low"** which means *a* = 0 will turn on segment *a*, and *a* = 1 will turn it off. The same is true for all segments. Applying 0 means the segment is on and 1 means it is off. For more information see Section 2.6 pg 72 of your text book and Fig. 16, 17, and 18 of the Basys 3 Reference Manual.

Table 1. Complete the truth table for 7-segment hex display

D3	D2	D1	D0	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	1	1	1	0	0	0	0	0
1	1	0	0	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0	0	1	0
1	1	1	0	0	1	1	0	0	0	0
1	1	1	1	0	1	1	1	0	0	0

Apply Karnaugh map technique that we learned in class to obtain the logic expression for each output a, b, c, d, e, f, g.

a: ~D3 &~D2& ~D1& D0 | ~D3 &D2& ~D1& ~D0 |D3 &~D2& D1& D0 | D3& D2& ~D1& D0;

b: ~D3 &D2& ~D1 &D0 | D2 &D1& ~D0 | D3& D1 &D0 | D3& D2 &~D0;

c: ~D3 &~D2 &D1 &~D0 | D3 &D2& ~D0 | D3& D2& D1;

d: ~D2 &~D1& D0 | ~D3 &D2& ~D1& ~D0 | D2 &D1& D0 | D3 &~D2& D1 &~D0;

e: D3& D0 | ~D2 &~D1& D0 | ~D3& D2 &~D1;

f: ~D3 &~D2 &D0 | ~D3 &~D2& D1 | ~D3& D1 &D0 | D3 &D2& ~D1& D0;

g: ~D3 &~D2 &~D1 | ~D3& D2 &D1& D0 | D3 &D2 &~D1 &~D0;

2. Derive the logic expression for 1-bit full adder

The next component that we will use in Lab 2 is a 1-bit full-adder. Fig. 2 shows the block diagram. It has 3 inputs, A, B, and Carry_in, and 2 outputs Sum and Carry_out.

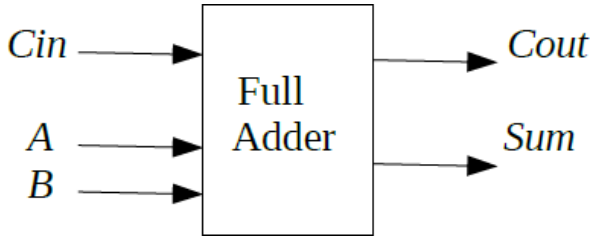

Figure 2. Block diagram of 1-bit full adder Using a

Karnaugh map, find minimized expressions for Cout and Sum.

Sum: ~A ~B Cin + A ~B ~Cin + A B Cin

Cout: B Cin + A Cin + A B

A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

3. Derive the logic expression for 2's complement addition overflow detection

In Lab 2, we will build a 4-bit adder to perform 2's complement addition. As we learned in class, we can check if overflow has occurred in the output by check the sign of the input and output binary numbers. As shown below, if A3 = B3 = 0 but S3 =1 then overflow is occurred. Similarly, if A3 = B3 = 1 but S3 = 0 then overflow is occurred. If A3 ≠ B3 then overflow should not occur.

$$\begin{array}{rr} & A_3\ A_2\ A_1\ A_0 \\ + & B_3\ B_2\ B_1\ B_0 \\ \hline & S_3\ S_2\ S_1\ S_0 \end{array}$$

Write out a Boolean expression for detecting OVERFLOW in addition of two 4-bit *signed* numbers.

OF_S = ~A ~B Cin + A B~Cin

Write out a Boolean expression for detecting OVERFLOW in addition of two 4-bit *unsigned* numbers.

OF_U = Cout

Note: Take help from the lectures 9 and 10 in which instructor covered 4-bit 2s complement adder.