SANTA CLARA UNIVERSITY	ELEN 21L Fall 2022

Laboratory #3: Multiplexer-based Design for 2-bit Adder
For lab sections Monday-Friday October 17-21, 2022

I. OBJECTIVES

In this laboratory you will

- Use Shannon's expansion to synthesize a logic function.
- Design the circuit using both schematic capture and hardware description language.
- Create a hierarchical design that connects multiple smaller circuits together.

PROBLEM STATEMENT

In this laboratory, we will design a 2-bit adder circuit which has two 2-bit integers, *A* and *B*, as inputs. Its output *S* is a 3-bit integer which is the sum of A and B. The two bits of the A input, *A1* and *A0*, can represent integer values from 0 to 3. Similarly, inputs *B1* and *B0* also can represent values from 0 to 3. The 3-bit output *S2*, *S1*, and *S0* can represent the values of the sum from 0 to 6.

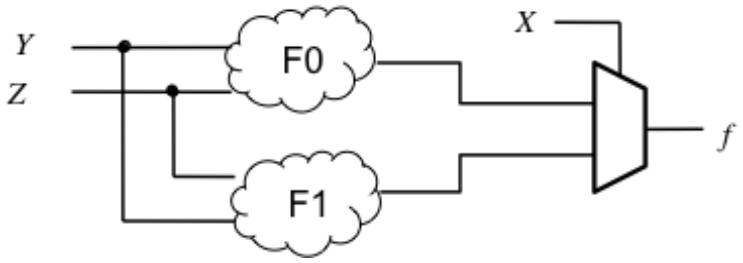
This circuit could be designed using methods of earlier labs to create three functions of four variables, one for each of the adder outputs, *S2*, *S1*, and *S0*. Instead, using Shannon's expansion, each output will be created by designing two functions of three variables and then using the fourth variable to select one of the two functions using a 2:1 multiplexer (MUX).

The 2-bit adder circuit will be created by connecting nine smaller circuits. The description of the six functions (two for each *S* output) and the 2:1 MUX will be entered into the circuit design using two different approaches: schematic capture and Verilog, a hardware description language. The Verilog approach may use either structural Verilog or continuous assignment (behavioral) Verilog. In the reference section at the end of this lab document, a 2:1 MUX is described using each method.

II. PRE-LAB

Prelab Part 1: Create the truth table
Given the problem description for the two-bit adder, create the complete truth table for this circuit with four inputs and three outputs (see table on following page).

Prelab Part 2: Create schematic options for the circuit outputs
Each of the three outputs of the 2-bit adder will be generated by a 2:1 MUX. The figure below shows an example of using a 2:1 MUX to create an output function of three variables, X, Y, and Z, using X as the select input. If X is 0, f = F0. If X = 1, f = F1. Both F0 and F1 are functions only of Y and Z, the two input variables not used for the select.
F(X,Y,Z) = X' F(0,Y,Z) + X F(1,Y,Z) = X' F0(Y,Z) + X F1(Y,Z)



Shannon's expansion is described in Section 4.1.2 of the recommended text. For the two-bit adder with 4 inputs, each of the three outputs will be created by a 2:1 MUX and two functions of the three input variables not used as the MUX select. Any input variable could be chosen as the MUX select. For example:

Mux select	**f(q,r,s,t)**
q	f(q,r,s,t) = q'·f(0,r,s,t) + q·f(1,r,s,t)
r	f(q,r,s,t) = r'·f(q,0,s,t) + r·f(q,1,s,t)

a) In order to apply the Shannon's expansion, you could use any of the four inputs *A1*, *A0*, *B1*, and *B0* as your MUX select bit for each of the three outputs *S2*, *S1*, and *S0*, but for this lab, **choose A1** as the MUX select for all three. Keep reading for further details.

b) For output *S0*

   i. Divide the truth table into two subtables – one where the S0 MUX select variable A1 is always 0 (the top half of the table) and one where the S0 MUX select variable is always 1 (the bottom half of the table).

   ii. Write a logic function for each of these two three-variable functions that exclude A1. These two logic functions will be the two 2:1 MUX data inputs. We will call these S0F0 and S0F1.

c) Repeat for S1 and S2, to generate the functions S1F0, S1F1, S2F0, and S2F1.

A1	A0	B1	B0	S2	S1	S0
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Table 1: Partial Truth Table using input A1 as MUX select

Prelab Part 3: Create a block diagram of the circuit
After completing Prelab Part 2, you will have six different functions using *A1* as the MUX select input: the *F0* and *F1* functions for each of the three outputs *S2*, *S1* and *S0*.

a. Draw a block diagram that shows three 2:1 MUXes. Label the output of each MUX as one of the 2-bit adder outputs (S2, S1, and S0).

b. For each MUX, use *A1* as the mux select signal.

c. Draw six blocks to represent the six sub-functions providing the data inputs to the three MUXes and label them.

d. Indicate the correct inputs to each of the six sub-function blocks.

Prelab Part 4: Create schematics and write simple Verilog modules
Given the block diagram you created in part 3, there are seven blocks that need to be implemented and connected: the six sub-functions, which you will have named uniquely, and the 2:1 MUX, which will be instantiated three times. Each of these seven blocks can be implemented either as a **schematic** or written in either **structural or behavioral Verilog.** Structural Verilog instantiates AND/OR/NOT primitives. Behavioral Verilog uses assign statements. (See the reference at the end of this assignment.) Note behavioral Verilog is synonymous with continuous assignment Verilog.

We will do both of the *S2* sub-functions **using schematic capture**, and all of the S1 and S0 sub-functions **using Verilog.**

a) Draw the schematics for *S2*.

b) Write the Verilog for *S1*.

c) Write the Verilog for *S0*.

Turn in the completed truth table, block diagram, schematic circuit diagrams, and Verilog source code as your prelab.

III. LAB PROCEDURE

Lab Part 1: Work with your partner to make sure you both have correct implementations for *S2*, *S1* and *S0*, as specified in the pre-lab.

Lab Part 2: Capture your implementation decisions in Quartus

- Create the Quartus II schematic and Verilog files for your seven blocks.
- Make symbols for each of the seven blocks.
- Create a top level design that instantiates and connects your seven blocks.
- Instantiate the seven segment display module and connect the outputs of your three MUXes to the appropriate inputs of the display module. What should the fourth input to the display module be?

Lab Part 3: Download and test your design

- Assign pins to connect to four switches to the data inputs *A1*, *A0*, *B1*, and *B0* on the evaluation board. Specifically, use *SW*[3] for *A1*, *SW*[2] for *A0*, *SW*[1] for *B1*, and *SW*[0] for *B0*.
- Assign pins to connect your three outputs to individual LEDs.
- Assign pins for the seven segment display module output to drive a seven segment display on the evaluation board.
- Download your design onto the FPGA and test it.
    - First set all four inputs to 0 and make sure the outputs are all 0.
    - Set each input to 1 while the other three inputs are 0 and verify that the outputs are correct.
    - Verify that the outputs are correct for all input combinations.
- If the circuit is not functioning correctly, make circuit corrections and download and test the corrected circuit.
- Demonstrate your working circuit to your lab assistant.

IV. REPORT

To be completed with your lab group and submitted to Camino by the specified deadline.

- Write an introduction describing the behavior of your circuit for the implementations of all three outputs *S2*, *S1* and *S0* that you made.
- Include the logic expressions for the functions you used.
- Include your final schematics and Verilog code, and proof of successful download and functioning on the FPGA.
- If, during testing, you found problems with the circuit that required correction, explain what you observed that demonstrated the problem and describe how you determined the cause of the problem and how you fixed it.
- Write a conclusion about the challenges of this lab and what you learned in this lab.

V. REFERENCES

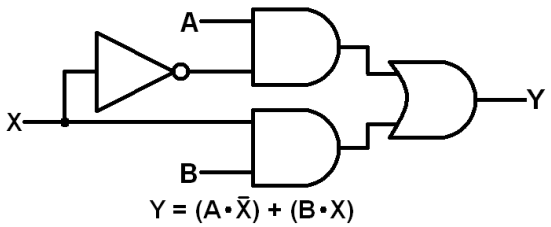
- Shannon's Expansion Theorem

Shannon's Expansion Theorem Any Boolean function $f(w_1, \ldots, w_n)$ can be written in the form

$$f(w_1, w_2, \ldots, w_n) = \overline{w}_1 \cdot f(0, w_2, \ldots, w_n) + w_1 \cdot f(1, w_2, \ldots, w_n)$$

This expansion can be done in terms of any of the n variables. We will leave the proof of the theorem as an exercise for the reader (see Problem 4.9).

- Schematic circuit for 2:1 MUX



- Verilog descriptions of 2:1 MUX

Structural	Continuous assignment
module example1 (x1, x2, s, f); **input** x1, x2, s; **output** f; **not** (k, s); **and** (g, k, x1); **and** (h, s, x2); **or** (f, g, h); **endmodule**	**module** exmpale2 (x1, x2, s, f); **input** x1, x2, s; **output** f; **assign** f = (~s & x1) | (s & x2); **endmodule**