SANTA CLARA UNIVERSITY

Fatemeh Tehranipoor, James Lewis, and Tokunbo Ogunfunmi **ELEN/COEN 21L**

Laboratory #4: Multiplexer-based Design for 2-bit Adder

For lab sections Monday-Friday Oct. 19 – Oct. 23, 2020

I. OBJECTIVE7

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.

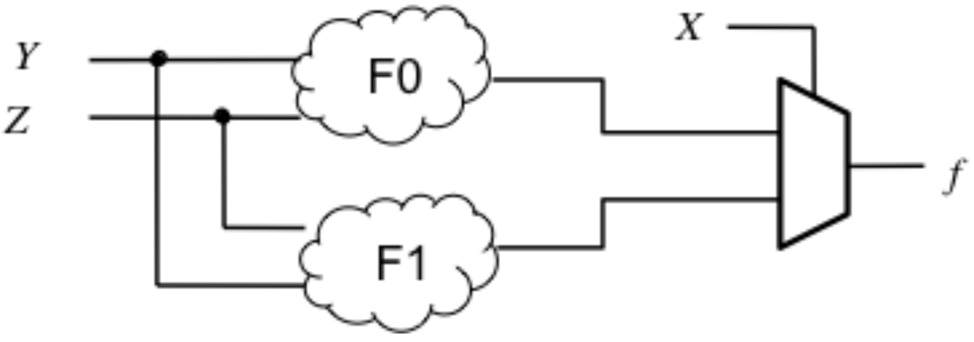
The 2-bit adder circuit will be created by connecting nine smaller circuits. The description of the six functions (two for each S output) and the 2:1 multiplexer will be entered into the circuit design using two different approaches: Schematic capture and Verilog, a hardware description language. In the reference section at the end of this lab description, a schematic representation of a 2:1 MUX is shown, and there are two Verilog descriptions: one describing the structure of the circuit using gate-level primitives, the other describing the behavior of the mux using a continuous assign statement.

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.

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, S0. But for this lab, we **choose A1** as the MUX select for each of S2, S1, S0. Use the same variable A1 for each output. This will amount to the truth table shown below in Table 1:

b) For output S0
   i. Divide the S0 K-map or S0 truth table into two parts – one where the S0 MUX select variable A1 is always 0 and one where the S0 MUX select variable is always 1.
   ii. Write a logic function for each of these two three-variable functions. 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.
b. For each MUX, use A1 as the mux select signal.
c. Draw the six blocks to represent the six sub-functions providing the data inputs to the three MUXes
d. Indicate the correct inputs to each of the six sub-function blocks.
e. Give each of the six blocks a unique name.

Prelab Part 4: Create schematics and write simple Verilog modules

Given the block diagrams 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 then we will do 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 tables, K-maps, block diagrams, schematic circuit diagrams, and Verilog source code as your prelab.

III. LAB PROCEDURE

Capture your implementation decisions in Quartus

- Create the Quartus II schematic and Verilog files for your seven blocks. Note that, in the pre-lab, you did not need to provide a mux implementation, since the answer is essentially given to you in the reference section at the end of this lab document. But you will still need to implement a mux now. You can choose for yourself whether to implement your mux in Verilog or to capture its schematic.
- Make symbols for each of the seven blocks
- Create a top level design that instantiates and connects your seven blocks
- Simulate your design to make sure it works properly
  - 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. Note that, when either A1 or B1 is set to 1, you should get a different output than when you set either A0 or B0 to 1.
  - Verify that the outputs are correct for all input combinations.
- Demonstrate your working circuit to your TA

.IV. REPORT

To be completed 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 K-maps and the logic expressions for the functions you used.
- Include your final schematics, your final Verilog code.
- 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

Right: Schematic circuit for 2:1 MUX

Below: Structural Verilog description of 2:1 MUX

Lower right: Continuous assignment Verilog description of 2:1 MUX

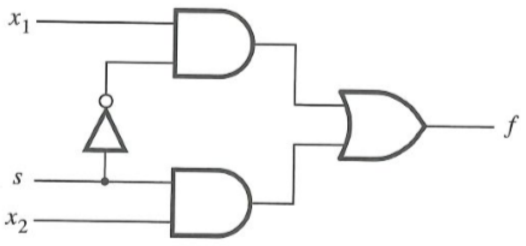


Figure 2.36 The logic circuit for a multiplexer.

```
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
```

Figure 2.37 Verilog code for the circuit in Figure 2.36.

```
module example3 (x1, x2, s, f);
   input x1, x2, s;
   output f;

   assign f = (~s & x1) | (s & x2);

endmodule
```

Figure 2.40 Using the continuous assignment to specify the circuit in Figure 2.36.

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).