

INTRODUCTION TO DIGITAL SYSTEMS

SYSC2310 (Fall 2020)

Dr. Mostafa Taha

Lab 2: Using and Simplifying Boolean Functions

Objectives

- Get familiarized with simplifying Boolean functions.
- Convert Boolean functions to actual circuits.
- Build a 2-bit x 2-bit binary multiplier.

Relation to course outcomes

The work in this lab is related to the two following course outcomes:

- Manipulate or design basic combinational operators.
- Apply analysis skills to correctly describe the behavior of given combinational logic circuits.


Preparation for the lab

1. Take the time to read this document prior to arriving to the lab in order to understand the essence of the work to be performed.
2. Under your 'sysc2310lab/lab2' folder, create three folders 'Ex1', 'Ex2' and 'Ex3', to save your work during this lab. Remember, by the end of the term, you are required to have the files for all the exercises in every lab. TAs and lab technicians will not be able to recover any deleted file, or any file that you have overwritten by mistake.
3. Run Logisim.exe following instructions in the previous lab.

Exercise 1: Simplify a Boolean Logic Function.

1. Simply the following function (F1) to a new function (F2) with at most 5 literals. You can review the examples discussed in the lecture to help you in solving this problem. The number of literals is the total number of inputs to a function, considering both the variables and their complements. For instance, currently F_1 has 7 literals (x, x', y, x', w, w', z).

$$F_1 = xy'z + x'y'z + w'xy + wx'y + wxy$$

2. Record and compare, in your lab notes, the number of gates that should be used to implement F1 vs F2. How many gates did you manage to save?
3. Build a circuit in Logisim to implement function F1.
4. In the same file, and using the same inputs, build an independent circuit to implement F2.
5. Connect F1 and F2 to a simple comparator to confirm that they are equivalent. For this design, we will use an XNOR gate, where the output of the XNOR gate is a '1' if its inputs are equal, and '0' if its inputs are different.
6. Connect a LED to the output of the XNOR gate.
7. Enable logging to 'Lab2_Ex1_log.txt', and use the poke tool () to change the values of the input bits. Notice output LED.
8. Save the circuit as 'Ex1.circ'.

Exercise 2: Convert a set of Boolean functions to a circuit.

1- Implement the following four functions:

$$M_0 = A_0 B_0$$

$$M_1 = A_0 B_1 \oplus A_1 B_0$$

$$M_2 = (A_0 B_1 \cdot A_1 B_0) \oplus A_1 B_1$$

$$M_3 = A_0 B_1 \cdot A_1 B_0 \cdot A_1 B_1$$

Each of the inputs A and B as well as the output M should be implemented as input ‘Pins’ and output Pins, with data bits of 2, 2 and 4 respectively. Splitters should be used to divide the two 2-bit line inputs and the 4-bit output line to individual lines for the internal gates.

3- Enable logging of the inputs (A and B) and output M to a file name ‘Lab2_Ex2_log.txt’, while using radix-10

4- Simulate the circuit by using the poke tool, to test all the possible inputs.

5- Save the circuit as ‘Ex2.circ’.

As you should have observed, the above circuit implements a 2-bit \times 2-bit binary multiplier. In order to find the above equations, we perform a regular multiplication as follows:

$$\begin{array}{r}
 \begin{array}{cc} A_1 & A_0 \\ \times & B_1 & B_0 \end{array} \\
 \hline
 \begin{array}{ccc} C_1 & A_1 B_0 & A_0 B_0 \\ + & A_1 B_1 & A_0 B_1 \end{array} \\
 \hline
 \begin{array}{ccc} [C_2 S_2] & [C_1 S_1] & \\ M_3 & M_2 & M_1 & M_0 \end{array}
 \end{array}$$

The LSB (C_0) equals to ($A_0 B_0$). Multiplication between two 1-bit inputs is just an AND gate. I.e., the previous equations means ($C_0 = A_0 \cdot B_0$). $A_1 B_0$ and $A_0 B_1$ are added using the ‘half-adder’ from Lab#1 since we are adding only no terms (no C_{in}), leading to a sum S_1 and a Carry C_1 . The sum is M_1 . The Carry C_1 is added to the $A_1 B_1$ again using a ‘half adder’ since we are adding only two terms ($A_1 B_1$ and C_1). The resulting sum is M_2 . The final carry C_2 is the output MSB, M_3 .

Note that, a half adder with inputs x and y has the following equations:

$$\begin{aligned}
 S &= x \oplus y \\
 C &= x \cdot y
 \end{aligned}$$

Exercise 3: Design a 3-bit by 2-bit binary multiplier.

1. Derive the function of the full adder studied in Lab#1.
2. Derive a closed form formulation to the multiplication of a 3-bit input $[A_2, A_1, A_0]$ by a 2-bit input $[B_1, B_0]$. The output would be $[M_4, M_3, M_2, M_1, M_0]$
3. Build the corresponding circuit.

“Please note, in Logisim, they used a not very common definition for the XOR gates with 3 or more inputs instead of the industry common standard of using it as a parity (equals to cascading

two XOR gates, each with 2 inputs). Therefore, in Logisim, use only XOR gates with two inputs.

4. Do not use XOR gates with more than 2 inputs in Logisim.
5. Enable logging of the inputs (A and B) and output M to a file name 'Lab2_Ex3_log.txt', while using radix-10
6. Simulate the circuit by using the poke tool, to test all the possible inputs.
7. Save the circuit as 'Ex3.circ'.

Congratulations. Now you can design a fully functioning MUL instruction
with arbitrary size inputs/outputs.

Demonstration:

- In order to get full credit, complete all exercises and be prepared to answer questions asked by the TA during the one-on-one meeting.
- Before you leave the lab, even if you could not complete the lab, make sure you sign-out with the TA.
- If you could not complete the lab during the scheduled lab time, finish up the lab on your own time. In this case, the demonstration points will be reduced in accordance to the percentage of work completed.
- If the student is absent or does not show the TA the work completed, 0 points are attributed for this part.

INTRODUCTION TO DIGITAL SYSTEMS

SYSC2310 (Fall 2020)

Dr. Mostafa Taha

Lab 2: Answer Sheet

Exercise 1: Simplify a Boolean Logic Function.

$$\begin{array}{ll} \text{Original} & F_1 = xy'z + x'y'z + w'xy + wx'y + wxy \\ \text{Simplified:} & F_1 = \end{array}$$

Exercise 3: Convert a set of Boolean functions to a circuit.

Equations of a half adder:

$$S = x \oplus y$$

$$C = x.y$$

Equations of a full adder:

$$S =$$

$$C =$$

$$\begin{array}{rcccc} & A_2 & A_1 & A_0 & \\ & \times & B_1 & B_0 & \\ \hline & & A_2B_0 & A_1B_0 & A_0B_0 \\ + & A_2B_1 & A_1B_1 & A_0B_1 & \\ \hline [C_3S_3] & [C_2S_2] & [C_1S_1] & & \\ M_4 & M_3 & M_2 & M_1 & M_0 \end{array}$$

$$M_0 =$$

$[C_1S_1]$ are computed using a [half / full] adder?

$$M_1 = S_1 =$$

$$C_1 =$$

$[C_2S_2]$ are computed using a [half / full] adder?

$$M_2 = S_2 =$$

$$C_2 =$$

$[C_3S_3]$ are computed using a [half / full] adder?

$$M_3 = S_3 =$$

$$C_3 =$$

$$M_4 =$$