ECE 85L Digital Logic Design Laboratory

Fresno State, Lyles College of Engineering
Electrical and Computer Engineering Department

Spring 2018

Laboratory 5 – Design and Implementation of Combinational Circuits with Logic Gates

1. OBJECTIVES

- Understand the relationship between a Boolean Expression and the corresponding Truth Table
- Develop a Logic Gate implementation of a Boolean Expression
- Algebraically minimize a Boolean Expression and design a Combinational Circuit

2. DISCUSSION

In previous labs, you have been exposed to the 7 basic types of Gates: NOT, AND, OR, NAND, NOR, XOR, and XNOR. To implement any general Boolean function you may use several of these types of gates within the same circuit. However, two of these types of gates (NAND and NOR) are capable of implementing any Boolean function without any additional Gate types.

Although Boolean expressions come in various forms, all Boolean expressions can be reduced and shown to be equivalent to either of two "standard" forms: the Sum of Products (SOP) form, and the Product of Sums (POS) form.

2.1 Sum of Products (SOP)

The Sum-of-Products (SOP) form is a summation of a set of product terms, each of which contains a set of variables or their complements.

SOP Example

$$X = A*BC + AB + AB*C*$$

By observation, either an SOP form can be directly implemented using AND, OR, and NOT components.

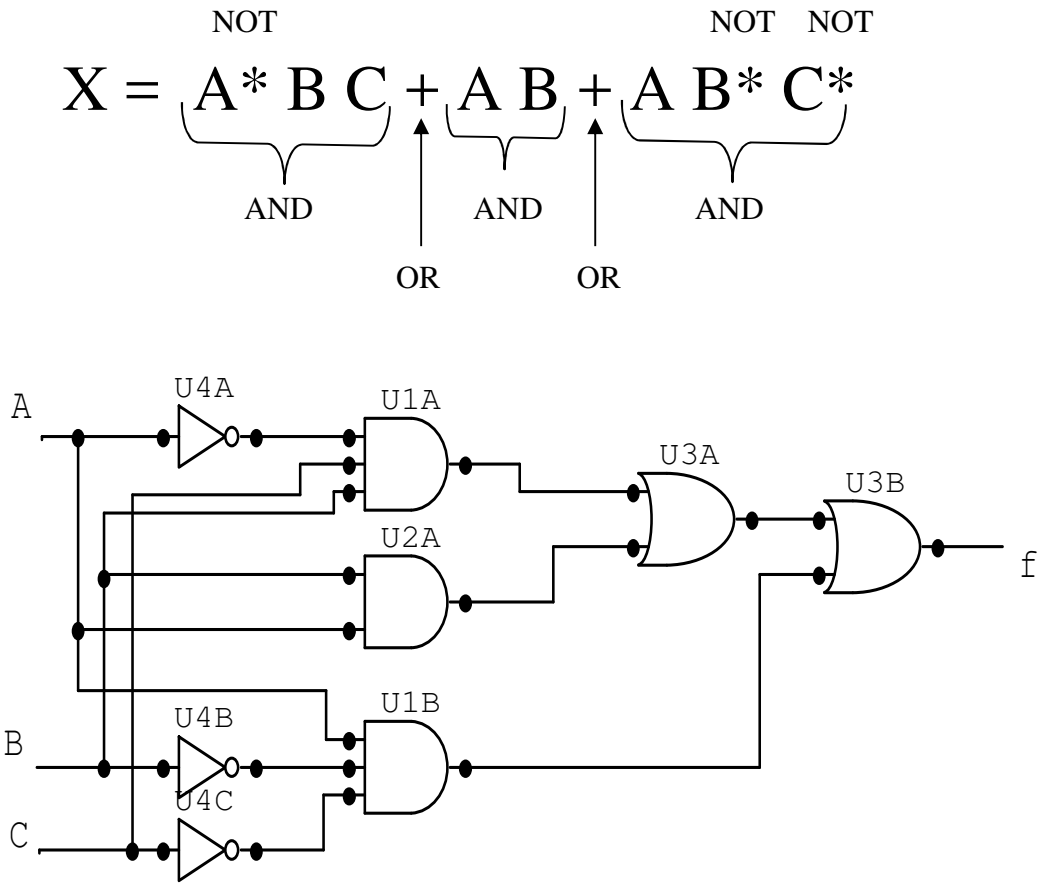


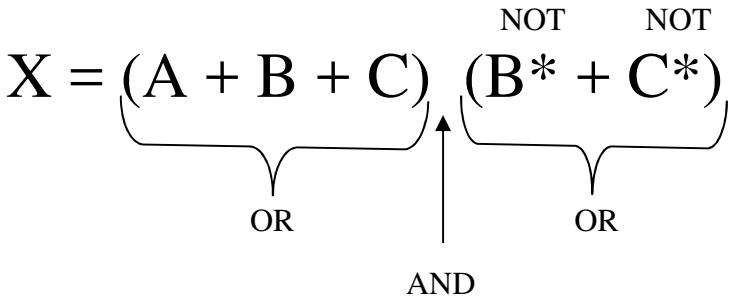
Figure 2.1: Logic Gate Implementation of an SOP Expression using NOT, AND, and OR Gates

2.2 Product of Sums (POS)

POS Example

$$X = (A + B + C)\ (B^* + C^*)$$

Similar to the SOP form, a POS Boolean expression can also be directly implemented using AND, OR, and NOT Gates, also by inspection.



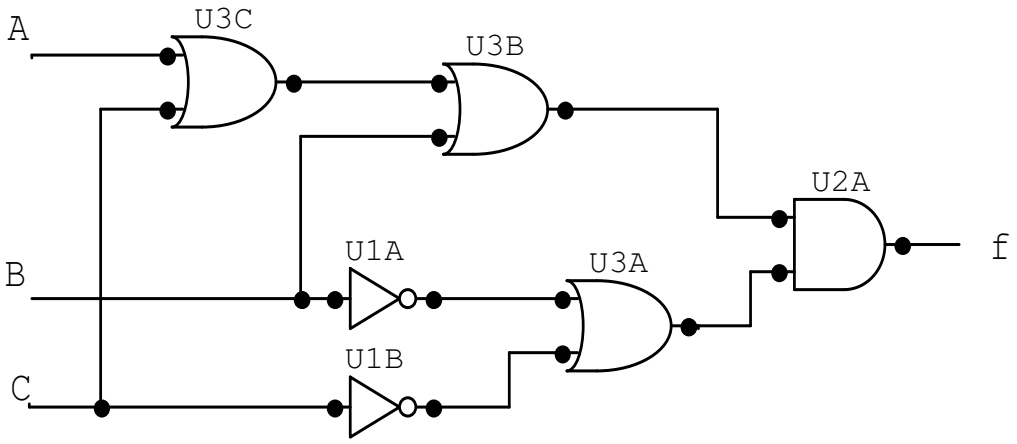


Figure 2.2: Logic Gate Implementation of a POS Expression using NOT, AND, and OR Gates

2.3 Non-Standard Form

A Non-Standard form is a Boolean expression that is not fully in SOP or POS form, or it contains other terms (i.e., XOR; XNOR) aside from sum and product terms.

Non-Standard Example

$$X = AB + C(D^* + E) + (A \oplus B^*)$$

2.4 SOP and POS Boolean Expressions using Truth Tables

An SOP or POS Boolean Expression may be obtained directly from a Truth Table (or vice versa) by creating a sum of product terms from the input variables that produce an output of a 1 (for SOP), or by creating a product of sum terms from the input variables that produce an output of a 0 (for POS).

Table 2.1: SOP and POS Expressions Derived from a Truth Table

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	1	1	0
1	1	0	1
1	1	1	0

SOP: X = A*BC + AB*C* + ABC*

POS: X = (A+B+C) (A+B+C*) (A+B*+C) (A*+B+C*) (A*+B*+C*)

3. PRELAB

1. Use Multisim to capture the schematics of Figure 2.1 and 2.2. Use components of your choice, and verify the operation of your circuit by simulating your circuit and developing a Truth Table. Label your schematic with the date and your name. Print it. Bring a copy of the printed schematic to the Laboratory to verify your work.

2. Study the example of the Two Bit Binary Comparator shown below, a portion of which you will design, fabricate and test in the laboratory. Come to the lab period with an initial design (Truth Table, Boolean Expression, and Schematic Design) prepared for your circuit.

Two-Bit Binary Comparator

A two-bit binary comparator is a digital circuit which compares two, **two-bit** binary numbers (**A** and **B**) and produces one of three outputs depending upon the result of the comparison (**A > B**, **A = B**, or **A < B**).

From this written statement you must first translate the description into a Boolean expression (either algebraic form or using a Truth Table) before implementation.

2.1 From the verbal description of the problem, decide how many Boolean input variables exist. Assign each input variable a symbol. In a comparable manner, decide how many Boolean output variables exist. Likewise, assign each output variable an appropriate symbol.

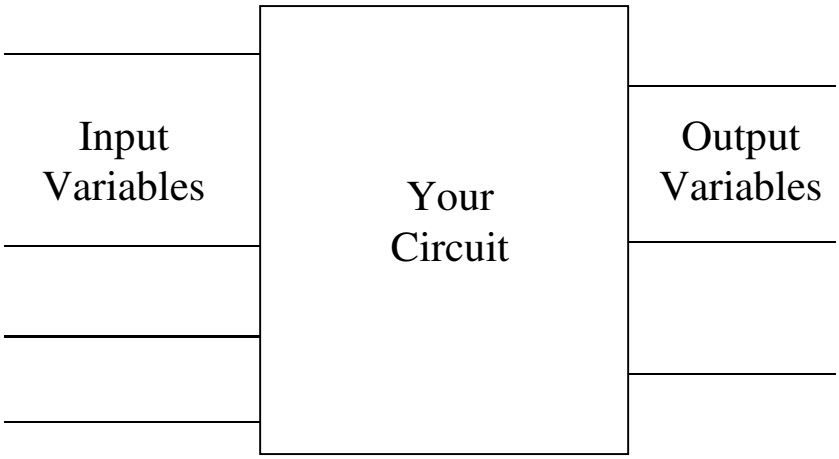


Figure 3.1: Block Diagram Representation of your initial circuit design

2.2 Even though the process above indicates "*a*" circuit, in actuality there are *N* circuits contained within the block diagram – one for each output variable in the design. Since all circuits use the same set of input variables, a more appropriate block diagram might be:

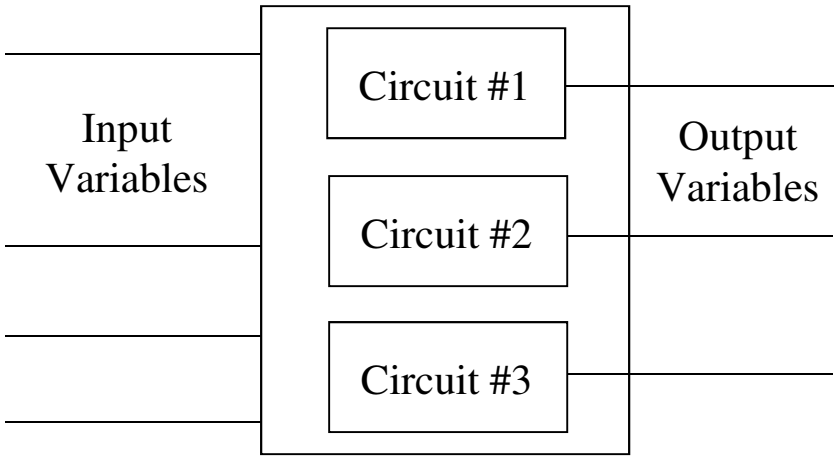


Figure 3.2: Expanded Block Diagram illustrating a sub-circuit for each Output Variable.

2.3 In order to design these circuits, you must develop the Boolean expression or the Truth Table for each output. Develop the Truth Tables that relate the input variables to each output variable. It is highly recommended that you verify your Truth Tables with your Laboratory Instructor before coming to lab.

2.4 From the Truth Tables, write the Boolean expression for each output variable in Sum of Products form. For this assignment do not minimize your logic functions.

4. LAB ASSIGNMENT

4.1 Two-Bit Comparator Implementation for A>B

1. Although you will learn formal techniques to minimize (reduce) each of these expressions in later labs, in this experiment you will implement the expressions directly without minimization. Select gates of your choice (based on the set available in your lab kit) to implement the comparator only for the circuit which produces an output for A>B. Document and verify your design using Multisim.

2. Fabricate your design. Although there are various techniques for fabrication, the one used most often is called Connection Mark-Off. This technique uses a labeled

schematic diagram (such as that you produced in step 1 from Multisim) as a basis for verifying your circuit wiring. Each time a wire is placed on the circuit, mark off the corresponding connection on the schematic (using for example a yellow felt tip pen). In this manner one can be relatively certain that all connections have been made. This technique also lends itself to partner participation, with one partner calling out the interconnections and marking them off, and the other partner physically interconnecting the circuit.

3. Using the Logic Bit Input Switches to simulate the input variables and one Logic Bit Output LED to simulate the output variable, verify the operation of your circuit by developing a (measured) Truth Table that you should compare with that which you developed in Step 2 of the Prelab. If they do not agree, you must "debug" your circuit. Debugging is a learned skill, the more you do the better you are! Initially you probably will need some help from your instructor. However, debugging someone else's circuit without a schematic diagram is a wasted exercise. Hence if you wish to have help from your instructor, you **must** have a schematic diagram available (this "rule" will apply for the remainder or the semester).

4. Demonstrate your working circuit to your Laboratory Instructor.

4.2 Extra Credit

1. Implement the circuit that implements the comparison for **A = B**. Demonstrate your working circuit to your Laboratory Instructor.

2. Implement the circuit that implements the comparison for A < B. Demonstrate your fully functioning two-bit comparator to your Laboratory Instructor.

5. REVIEW QUESTIONS

1. Which of the 7 basic Gate types can be used (by itself) to implement any Boolean function?

2. How do you make an NOT Gate from a NAND Gate? A NOR Gate from a NAND Gate? An XOR Gate from a NAND Gate?

3. For the comparison of two four bit numbers that differ in the MSB (Most Significant Bit), is it necessary to compare the remaining 3 bits to obtain the comparison result (given the result of the MSB comparison)?