Digital Logic
CSC 244L
Laboratory 2
Algebraic Simplification, DeMorgan's Theorems, and Sum-of-Product and Product-of-Sum Implementations

1 Objectives

We will:

1. simplify logic circuits using Boolean theorems;
2. investigate the relationship between minterms, Maxterms, sum-of-products (SOP) implementation, and product-of-sums (POS) implementation;
3. prove that SOP and POS circuits can be built using only NAND and NOR gates, respectively.

2 Pre Lab

To be completed *before* your lab meets (individually):

2.1 Read the entire lab procedure.

2.2 The following will be turned in, individually, to a D2L dropbox in your lab section before midnight on Monday before the lab day. Complete these items (in this lab manual) for your pre-lab:

- 3.1.1, 3.1.2, 3.1.3
- 3.2.1, 3.2.2 (pre-lab column), 3.2.3, 3.2.4, 3.2.5, 3.2.6
- 3.3.1, 3.3.2, 3.3.3

2.3 When you reduce an expression, show your work (write the number of each theorem/axiom used, one per line).

2.4 Bring your circuit diagrams and pre lab to lab with you so that your lab instructor can check them before you start working. Lab time is for building and debugging, not for initial design. Show your completed diagrams and have your TA sign off on your lab sheet.

3 Lab Procedure

3.1 Simplification using DeMorgan's Theorems

3.1.1 Complete the "F(A,B,C) original by-hand" column of the truth table in Table 1 for the following function (remember the order of operations! parentheses → NOT → AND → OR):

$$F(A, B, C) = (A' + B' \cdot C)' \cdot (A' + B' \cdot C')' \tag{1}$$

3.1.2 Draw a logic diagram for the function in Eq. 1 using AND, OR, and NOT gates (remember the order of operations!).

3.1.3 Create a SV file named "functionABC.sv" that contains one SV `module` named functionABC. Write structural SV using the built-in logic gate modules to describe the operation of the circuit you created. Label any `logic` wires required.

3.1.4 Compile the SV using Quartus Prime Lite. Assign switches SW[2:0] to the three inputs (SW[2] as MSB A, SW[0] as LSB C), and LEDR0 to monitor the output.

3.1.5 Load the circuit to your FPGA. Verify the operation of your SV module by checking every possible input combination and filling in the "F(A,B,C) original on DE10" column of Table 1. Check that the output matches the column you calculated by hand. If they do not match, double check both the SV and the original truth table and correct the output.

3.1.6 Discuss the process with your TA and show your working module and truth table to your TA, and have them sign off on your lab sheet.

Table 1: Truth Table for F(A,B,C)

A	B	C	F(A,B,C) original by-hand	F(A,B,C) original on DE10	F(A,B,C) canonical (or reduced) on DE10
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

3.1.7 Based on your results (in Table 1), write an equivalent expression for the circuit as the canonical sum-of-products:

3.1.8 Create a SV file named "functionABCreduced.sv" that contains one SV `module` named functionABCreduced. Write structural SV using the built-in logic gate modules to describe the operation of the canonical sum-of-products expression. You may be able to further reduce the logic using Boolean algebra.

3.1.9 Compile the SV using Quartus Prime Lite. Assign switches SW[2:0] to the three inputs (SW[2] as MSB A, SW[0] as LSB C), and LEDR0 to monitor the output.

3.1.10 Load the circuit to your FPGA. Verify the operation of your SV module by checking every possible input combination and filling in the "F(A,B,C) canonical (or reduced) on DE10" column of Table 1. Check that the output matches the other two columns. If they do not match debug your canonical sum-of-products expression or SV.

3.1.11 Use DeMorgan's theorems on the original expression (**hint:** you may want to use the distributive property within the parentheses first) and Boolean algebra to prove that the expression you obtained above is equivalent.

3.1.12 Demo your completed DE10 project, and show your truth table, equivalent expression, and proof to your TA and have them sign off on your lab sheet.

3.2 Boolean Simplification

3.2.1 Examine the circuit in Fig. 1 and write the Boolean expression for the output, F.

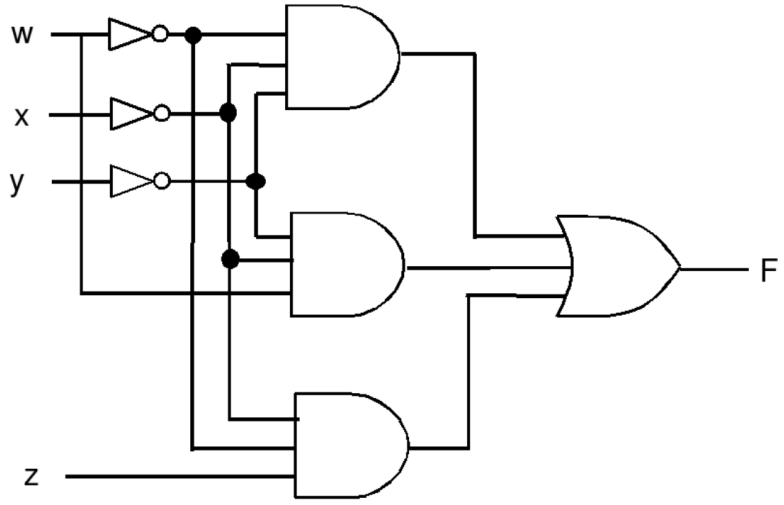

Figure 1: Circuit to Simplify: F(w,x,y,z)

3.2.2 Complete the "Original F theory-Pre Lab" column of the truth table for F in Table 2.

3.2.3 Create a SV file named "functionWXYZ.sv" that contains one SV `module` named functionWXYZ. Write structural SV using the built-in logic gate modules to describe the operation of the circuit you created. Label any `logic` wires required.

3.2.4 Simplify the expression you obtained above, using Boolean theorems. Your lab report should show each step of this simplification, and you should list the theorem (or axiom) used in each step. Draw a logic diagram for the simplified expression.

3.2.5 Create a SV file named "functionWXYZsimplified.sv" that contains one SV `module` named functionWXYZsimplified. Write structural SV using the built-in logic gate modules to describe the operation of the circuit you created. Label any `logic` wires required on your diagram.

3.2.6 Watch "demorgansTopLevel_video.mp4" on D2L that describes how to re-use your own modules in a SV project. During lab you can ask your TA for a refresher.

3.2.7 Create a third SV file named "fWXYZtoplevel.sv" that contains one SV `module` named fWXYZtoplevel. Add all three SV files to the same Quartus project, where the project should also be named fWXYZtoplevel. Use structural SV and **explicit port mapping** to connect together `functionWXYZ` and `functionWXYZsimplified` to the inputs and outputs of your DE10-Lite board.

Table 2: Truth Table for various implementation of F(w,x,y,z)

w	x	y	z	Original F theory-Pre Lab	Original F empirical on DE10	Simplified F empirical on DE10
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			

3.2.8 Compile the SV using Quartus Prime Lite. Assign switches SW[3:0] to the four inputs (SW[3] as MSB W, SW[0] as LSB Z) in the `fWXYZtoplevel` as inputs to both `functionWXYZ` and `functionWXYZsimplified` modules. Connect LEDR1 to monitor the output of `functionWXYZ` and LEDR0 as the output of `functionWXYZsimplified`.

3.2.9 Verify the operation of your modules by checking every possible input combination. Record these values in the appropriate columns of Table 2. If the two outputs do not match, debug your Boolean algebra and/or SV modules.

3.2.10 Demo your completed top-level module, and show your completed truth table to your TA and have them sign off on your lab sheet.

3.3 SOP and POS Implementations

Consider the following function:

$$f(A, B, C) = \sum m(0, 1, 5)$$

3.3.1 Write this function as a *canonical sum-of-products* and design the corresponding circuit. Fill out the "F(A,B,C) original by-hand" column of Table 3.

3.3.2 Express the function as a list of Maxterms (or shorthand using our $\prod$ notation), give the corresponding *canonical product-of-sums*, and design the corresponding circuit.

3.3.3 Use Boolean algebra to simplify both of the above expressions to determine the *minimal sum-of-products* and the *minimal product-of-sums* representation for the above function. Draw the corresponding logic diagrams for the minimal SOP and POS, including first-level inverters.

Table 3: Truth Table for F(A,B,C)

A	B	C	F(A,B,C) original by-hand	F(A,B,C) NAND SOP circuit	F(A,B,C) NOR POS circuit
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Laboratory 2 Signoff Sheet

Student Name (Print in Black Ink):

To be turned in (scan a copy) with your lab report (back sheet) to D2L.

TA Signature	**Section**
	2.4
	3.1.6
	3.1.12
	3.2.10
	3.3.3

Lab Completed: by signing this, I affirm on my honor that I am aware of the student disciplinary code and that I have successfully completed this laboratory assignment.

Student Signature: Date: