Kettering University

Digital Systems I

Lab Exercise 3

Switching Algebra

and

Analysis and Design of Digital Circuits

Getting Started

Altera Quartus II Software and DE2 Board

and

ENTITY, ARCHITECTURE, and
Simple Signal Assignments in VHDL

Spring 2019

Prelab: No experiment is required for assignments **3, 8** and **11**, which are your prelab. You will answer these questions before your week03's lab time starts.

Note: The lab report is due on coming Friday in class.

Section

Name **Number**

Lab Partner's Name and Number

Purpose of this lab experiment:

Objectives

- *Understand the basics of FPGAs and VHDL*
- *Get started with QuartusII CAD tool and DE2 development/education board*
- *Examine how switching algebra may be utilized to manipulate logic expressions; more specifically, verify the following theorems by obtaining and analyzing appropriate truth tables:*
  - *Distributivity*
  - *Covering,*
  - *Combining*
  - *Elimination*
  - *DeMorgan's*
- *Get hands-on experience in manual circuit design and empirical circuit analysis*

What to Hand in/Do

- You will show your **prelab** first, which is explained on the cover sheet.
- Once you are done with the assignments, turn in this handout with
  - Fully completed cover sheet (Pages 1 and 2)
  - All truth tables
  - Answers to all questions. Read the whole document carefully. Some questions may not easily be visible!
- Be prepared for possible questions that you may be asked regarding different parts of today's experiments.
- Also, show your functional circuits in Assignments **1**, **6** and **12**.

Introduction

Switching Algebra is a mathematical tool that you need to manipulate a logic expression manually to tailor its format to your needs. For example, when you learn this algebra, you may replace A + 1 and A . A' with a 1 and a 0, respectively, on the spot. Similarly, when you come across A + A . B, you would quickly notice that it equals A. Or when you come up with A + A' . B, you would easily recognize its equivalent expression, A + B. Switching algebra is more crucial for the analysis of more complicated logic expressions. Consider Y = (((A' + B)' . B . C' . D)' + (B . C' + (A . B')' . D)' + (A'. C . D)')' as an example. Switching algebra is your main tool to manually analyze this expression and eventually reach its truth table. Switching algebra also helps you translate a truth table into an expression as examined in Chapter 3, Part I.

You need to get much practice to be able to apply appropriate switching theorems to different problems. Today's lab exercise, as a complementary step to Chapter 3, Part I, is an attempt to reach the goal. However, this of course will not be possible unless you actively take part in all the experiments and **thoroughly understand what you are doing.**

In today's lab, you will see how to translate from the truth table domain into the circuit domain (design) and vice versa (analysis). For more details on circuit analysis and design see Chapter 3, Part I.

Today, you will also get a better understanding of the basics of FPGAs and VHDL. You will become familiar with the DE2 education board and learn how to use Altera Quartus II Software to map your code into the FPGA chip and more!

FPGA Pin Assignment

The DE2 board has hardwired connections between the FPGA pins and some other components on the board such as 18 slide switches, 18 red LEDs, 9 green LEDs and 8 seven-segment displays. As an example, Slide Switch 0 is connected to the FPGA pin N25. If we let the compiler know that we want to assign VHDL name SW(0) (i.e., the first element of a vector called SW) to pin N25, then we can easily use SW(0) in our VHDL code, and the compiler will understand that by SW(0) we mean Slide Switch 0.

The good news is all such pin assignments for the DE2 board are given in a file called *DE2_pin_assignments.csv*, which should be on your desktop. Make a copy of this file into your google or flash drive just in case the desktop copy is accidently deleted! To take advantage of this one-click easy pin assignment, just *import* the *DE2_pin_assignments.csv* file into your project:

Assignments => Import Assignments …

When the DE2_pin_assignments.csv file is imported, input SW(0) will be tied to Pin N25 and therefore to Switch 0 on the DE2 board.

Some of the inputs/outputs names defined in DE2_pin_assignments.csv are as follows:

```
LEDR:   STD_LOGIC_VECTOR (17 DOWNTO 0 )   -- Connected to 18 Red LEDs on DE2 board.
SW:     STD_LOGIC_VECTOR (17 DOWNTO 0)    -- Connected to 18 Slide Switches on DE2 board.
LEDG:   STD_LOGIC_VECTOR (8 DOWNTO 0)     -- Connected to 9 Green LEDs on DE2 board.
```

Note: For the sake of clarity, we normally drop the parentheses wherever possible, such as in tables and diagrams. For example, instead of SW(0), you will just see SW0. But keep in mind that you CANNOT use this abbreviation in VHDL files or the compiler will complain!

***

Quartus II is the software that you will use in this and the future labs to model digital circuits and map them into the FPGA chip on the DE2 board. See Appendix A of this handout for a Quick Start Guide of Quartus II. The lab instructor will explain the DE2 board. The tool's manual is available online.

Assignments

1. Read the incomplete code shown in Figure 1 carefully and then fill in the blank to complete the code. The code is supposed to tie the 18 slide switches to the corresponding red LEDs. Create a lab03 directory on your desktop. In lab03, create a project called tb. Then create a VHDL file called tb.vhd and type the code in this file. **Follow the implementation procedure described in Appendix A**. Do not forget to import DE2_pin_assignments.csv. **Show your functional circuit to the lab instructor.**

Note: It is a good habit to use the "entity name" as the "file name" as well.

Figure 1. VHDL incomplete code for Assignment 1

```
-- Place your comments after "--".
-- VHDL is case iNsEnSItivE.

LIBRARY ieee;

USE ieee.std_logic_1164.all;

ENTITY tb IS
      PORT (   SW        : IN       STD_LOGIC_VECTOR (17 DOWNTO 0);
               LEDR      : OUT      STD_LOGIC_VECTOR (17 DOWNTO 0)    -- No semicolon here!
            );
END tb;

ARCHITECTURE Algebraic OF tb IS
BEGIN
-- ********************* Fill in the blank. You need only one assignment! *********************




END Algebraic;
```

***

Note: Do not create a new project; use the same project, tb, in the rest of today's lab assignments. Also reuse the same file, tb.vhd, after you have made necessary changes; this will save you a lot of time ☺

2. **Distributivity (right):** You will map the two sides of T8-R, **a + (b . c) = (a + b) . (a + c)**, into the FPGA chip. A test bench for your design is shown in Figure 2.

   Note: The frequently used theorems examined in class as well as the axioms of switching algebra are shown in Appendix B at the end of this lab handout.

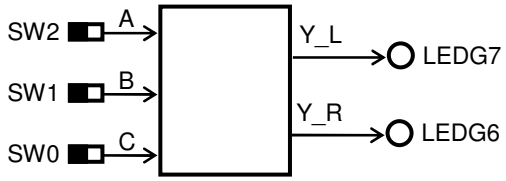


Figure 2. Test bench for Assignment 2

Your circuit will be driven by three slide switches 2, 1 and 0. The two outputs assigned to the two (right and left) expressions will be displayed on two green LEDs 6 and 7, respectively. You will find out whether or not these two outputs come from the *same* LUT.

A partially written code for the two left and right expressions (with shared input signals) is shown in Figure 3. **Read the code carefully** and then fill in the blank to complete the code. Edit the same file, tb.vhd, accordingly to model the two sides of T8-R.

Note: Read but (to save time) don't type the *general* comments!

Figure 3. Incomplete code for Assignment 2

```
-- Remember:
-- Add readable but short comments to your codes. The following code has already been commented.
-- Place your comments after "--".
-- VHDL is case iNsEnSItivE.

-- Distributivity (right)
-- The following goes to tb.vhd

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY tb IS  -- Read but (to save time) don't type the general comments!

-- Entity tells us what the inputs and outputs of circuit are, PERIOD.

-- Entity does not explain what the circuit performs. Circuit operation is described in the architecture.

-- Inputs and outputs (specified in entity) are visible in the outside world.

-- So, you may call a logic symbol "the graphical version of the entity".
```

Code continues on next page …

```
    PORT (  SW   : IN     STD_LOGIC_VECTOR (2 DOWNTO 0);     -- Inputs: Switches 2, 1 and 0
            LEDG  : OUT    STD_LOGIC_VECTOR (7 DOWNTO 6)     -- Outputs: Green LEDs 7 and 6
            );
END tb;

ARCHITECTURE Algebraic OF tb IS          -- The architecture tells us what the circuit performs:
    -- Let us also assign more convenient (shorter) names to inputs and outputs.
    -- Use ALIAS declaration to give an alternative name to an existing signal:
    ALIAS A      : STD_LOGIC is SW(2);          -- SW(2) is also called A.
    ALIAS B      : STD_LOGIC is SW(1);
    ALIAS C      : STD_LOGIC is SW(0);
    ALIAS Y_L    : STD_LOGIC is LEDG(7);
    ALIAS Y_R    : STD_LOGIC is LEDG(6);

BEGIN                -- Describe your circuit algebraically.
-- ********** Fill in the blanks **********
-- Hint: Use parentheses for ORs as well as ANDs
    Y_L <=                                                  ;
    Y_R <=                                                  ;
END Algebraic;
```

***

Map your code into the FPGA chip; then apply all possible input combinations and fill out the truth table shown in Figure 4 based on what you see on the two green LEDs.

Row	A B C	Y_L	Y_R
0	0 0 0		
1	0 0 1		
2	0 1 0		
3	0 1 1		
4	1 0 0		
5	1 0 1		
6	1 1 0		
7	1 1 1		

Figure 4. Truth table for Assignment 2

Does Quartus generate/use the same LUT for the two sides of T8-R? Circle the correct answer:

Yes No

Let us trust Quartus. Does the above experiment prove T8-R? Circle the correct answer:

Yes No

3. No experiment required (Pre-lab): In the spaces provided for Figure 5*a* and Figure 5*b*, draw a digital circuit to realize the left side of T8-R, and another digital circuit to realize the right side of T8-R, respectively. Use only logic gates (not LUTs).

$Y\_L = A + (B \cdot C)$ (a)

$Y\_R = (A + B) \cdot (A + C)$ (b)

Figure 5. Left and right circuits for distributivity property

Does Figure 5 *literally* illustrate the individual logic gates (e.g., AND, OR) for the circuits created in the FPGA chip in Assignment 2? Circle the correct answer:

Yes No

4. Covering (right): You will map the left side of T9-R, **y = a . (a + b)**, into the FPGA chip. The test bench is illustrated in Figure 6.

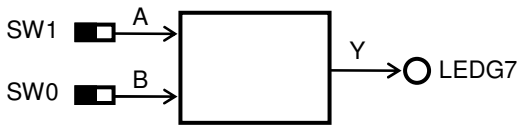


Figure 6. Test bench for Assignments 4, 5, 6 and 8

Apply all the possible input combinations to fill out the truth table shown in Figure 7.

a	b	y
0	0	
0	1	
1	0	
1	1	

Figure 7. Truth table for Assignment 4

Let us trust Quartus. Which theorem is proved by the above experiment? Write it in the *algebraic* format:

How many LUTs are used for this project?

5. **Elimination (right):** You will map Y = A . (A' + B) into the FPGA chip. The test bench remains the same as illustrated in Figure 6. Apply all the possible input combinations to fill out the truth table shown in Figure 8. **Note:** This theorem is not shown in Appendix B.

a	b	y
0	0	
0	1	
1	0	
1	1	

Figure 8. Truth table for Assignment 5

Let us trust Quartus. Which theorem is proved by the above experiment? Write it in the *algebraic* format:

How many LUTs are used for this project?

6. **Combining (left)**: You will map Y = A . B + A . B' into the FPGA chip. The test bench remains the same as illustrated in Figure 6. Apply all the possible input combinations to fill out the truth table shown in Figure 9. **Show your functional circuit to the lab instructor.**

A	B	Y
0	0	
0	1	
1	0	
1	1	

Figure 9. Truth table for Assignment 6

Let us trust Quartus. Which theorem is proved by the above experiment? Write it in the *algebraic* format:

DeMorgan's and Covering Theorems:

7. In this assignment, the following function will be simplified *empirically*:

$$Y = ((A \cdot B)' + (A + C)')'$$

DeMorgan's and Covering Theorems may be used to get the same result *theoretically*.

Map Y into the FPGA chip. The test bench is illustrated in Figure 10.

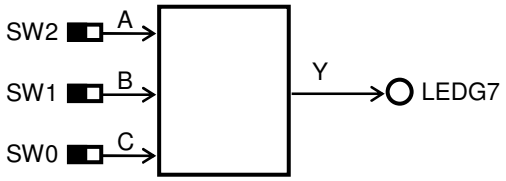


Figure 10. Test bench for Assignments 7 and 12

Apply all the possible input combinations to fill out the truth table shown in Figure 11.

Row	A B C	Y
0	0 0 0	
1	0 0 1	
2	0 1 0	
3	0 1 1	
4	1 0 0	
5	1 0 1	
6	1 1 0	
7	1 1 1	

Figure 11. Truth table for Assignment 7

Let us trust Quartus. Considering your results in Figure 11, write the theorem that is proved by the above experiment in the *algebraic* format:

Circuit Analysis

8. No experiment required (Pre-lab): The logic diagram of Figure 12 consists of four 2-input NAND gates. It has two inputs, A and B, and one output, Y. In this diagram, there are also three intermediate signals called Z, W and U.

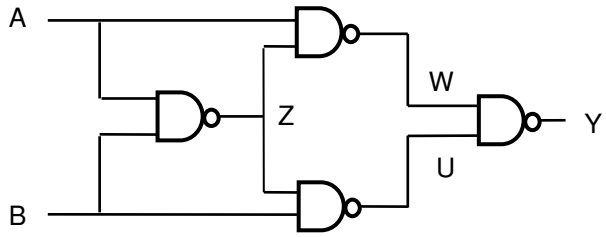


Figure 12. Logic diagram for Assignment 8

Write algebraic equations for signals Z, W, U and Y: (Fill in the blanks)

Z (in terms of A and B) =

W (in terms of A and Z) =

U (in terms of B and Z) =

Y (in terms of W and U) =

9. You will map the above expressions (and in the same order) into the FPGA chip. The test bench remains the same as shown in Figure 6. Apply all the possible input combinations to fill out the truth table shown in Figure 13. **Note:** Using the following format, **declare intermediate signals U, W and Z in the ARCHITECTURE** but before the BEGIN keyword:

```
SIGNAL      U, W, Z:      STD_LOGIC;
```

Row	A B	Y
0	0 0	
1	0 1	
2	1 0	
3	1 1	

Figure 13. Truth table for Assignment 8

According to your results in Figure 13: (fill in the blank)

Y (in terms of A and B. Use only AND, OR and NOT operators) =

How many LUTs are used for this project?

10. Back to Assignment 9 and in your .vhd file, use the following (different) order for the four simple signal assignments:

Y (in terms of W and U) <= *[same as before]*

W (in terms of A and Z) <= *[same as before]*

U (in terms of B and Z) <= *[same as before]*

Z (in terms of A and B) <= *[same as before]*

Map the reordered expressions into the FPGA chip and apply all the possible input combinations again. Does function Y depend on the *order* of these signal assignments?

Yes No

Mark the right choice to fill in the blank: Simple Signal Assignments are …

☐ **Concurrent** Statements ☐ **Sequential** Statements

Circuit Design

11. No experiment required (Pre-lab): Three-variable *majority function*: Suppose that there is a three-member committee with regular meetings in a three-seat conference room. Each seat has a sensor that generates a 1 when somebody sits on that seat. A 0 is produced otherwise. Also there is a light outside the room monitoring the situation: it turns on when the majority of members (two or three people) attend the meeting and take seats, otherwise the light turns off. Further suppose that the light used in this assignment turns on with a 1 and turns off with a 0. You are going design a light controller for the conference room.

Fill out the truth table shown in Figure 14 to describe the light controller. The table has three input variables, A, B and C (representing the sensors' outputs) and one output, L, (driving the light). Write the minterm list, the maxterm list, the canonical SOP and the canonical POS of L. **Use switching algebra** to minimize the two canonical expressions to get a minimal SOP and a minimal POS for L. **Note:** The logic minimization step is only for your more practice. Quartus II knows how to minimize logic expressions! Moreover, its criteria are normally different from yours as the target technology is now FPGAs. Quartus II does not even need canonical expressions; truth tables are sufficient! You will see how to describe truth tables in VHDL in the coming lab assignments.

L(A, B, C) = Σ( )

L(A, B, C) = ∏( )

Canonical SOP of L =

Canonical POS of L =

Row	A B C	L
0	0 0 0	
1	0 0 1	
2	0 1 0	
3	0 1 1	
4	1 0 0	
5	1 0 1	
6	1 1 0	
7	1 1 1	

Minimized L (SOP) =

Minimized L (POS) =

Figure 14. Logic expressions and truth table for Assignment 11

If standard CMOS gates were used, which format, SOP or POS, would be more cost-effective? Circle the correct answer:

A- SOP **B-** POS **C-** They are almost the same

Which expression do you prefer for FPGA mapping purposes?

canonical SOP canonical POS minimal POS minimal SOP

For FPGA mapping, does it matter which (correct) expression is chosen?

Yes No

Briefly but clearly explain why you may prefer not to use a minimal expression in this assignment:

12. Back to Assignment 11, choose one of the expressions (canonical SOP or POS, or minimal POS or SOP) and map it into the FPGA chip. The test bench is shown in Figure 10. Apply all the possible input combinations to fill out the truth table shown in Figure 15. **Show your functional circuit to the lab instructor.**

Row	A B C	L
0	0 0 0	
1	0 0 1	
2	0 1 0	
3	0 1 1	
4	1 0 0	
5	1 0 1	
6	1 1 0	
7	1 1 1	

Figure 15. Truth table for Assignment 12

Are the two truth tables shown in Figure 14 and Figure 15 identical?

Yes No

Appendix A: How to Use Altera Quartus Software

A Quick Start Guide

Create a folder called “lab03” on your desktop.

Double click on the Quartus icon to run the tool. Create a project called “tb” (6 steps):

File > New Project Wizard …

“Introduction” window pops up: Click on “Next”; Page 1 of 5 pops up:

In the first field “what is the working directory for this project?” browse and select your lab03 directory.

In the second field “what is the name of this project?” type: tb

Note that the same name, “tb”, will be automatically typed in the third (last) field as well. This means that the “project name” and the “top-level module name” should be the same: This is why you will call the entity of your VHDL code “tb” as well.

Click on “Next”, page 2 of 5 appears.

Click on “Next”; Page 3 of 5 will pop up. In the “family” drop-down list choose Cyclone II. In the “Available devices” table select: EP2C35F672C6

Click on “Next”; Page 4 of 5 will appear. We are not going to use other EDA tools; so

Click on “Next”; Page 5 of 5 (the summary page) will pop up. Check the information and make sure it is correct.

Question: Based on the information on this page, how much is the core Vcc? Write it here:

Click on “Finish”.

The Project is created ☺

***

To see what you did: Open your lab03 folder, you will see the following:

Folder db; File tb.qpf; File tb.qsf

Note that “tb.qpf” (Quartus Project File) is your project file. If you come back tomorrow, you need to open this file to open your project: File > Open project … (ctrl + J)

Do NOT add anything to folder “db”. You will add your VHDL files to folder “lab03”.

***

Create your VHDL file, “tb.vhd”, in your lab03 directory: To do this, you may use the built-in editor of Quartus: (the file is shown in your lab handout)

File > New … > VHDL file

If you have already created the file, just copy and paste it in your lab03 folder.

To edit an existing VHDLfile:

File > Open …

Note: Your project needs only one file in lab03. **In general, a project may have more.**

***

Add your VHDL file(s) to your project:

Project > add/remove files in project …

Note: Your project needs only one file in lab03. **In general, a project may need more.** So you have to add ALL the files to the project. Do NOT add any irrelevant files to the project.

***

Import the pin assignment file called *DE2_pin_assignments.csv* file into your project:

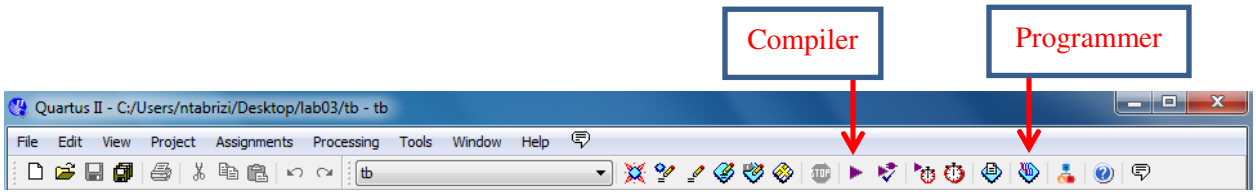
Assignments > Import Assignments …

DE2_pin_assignments.csv should be on your desktop.

See page 4 of lab03 handout for more details.

***

Click on the *triangle* on the menu bar to compile your VHDL file.



Once you get the successful compilation message, click on the *programmer* button on the menu bar. The "programmer" window appears. Click on "Start" button to map your code into the FPGA chip.

***

Test your circuit.

***

Appendix B: Axioms of switching algebra and some frequently used theorems

Axiom No	Left Axiom	Right Axiom	Comments
A1	a ≠ 0 => a = 1	a ≠ 1 => a = 0	"a" may take on either 1 or 0
A2	a = 1 => a' = 0	a = 0 => a' = 1	Definition of NOT
A3	1 + 1 = 1	0 . 0 = 0	The last 3 lines
A4	0 + 0 = 0	1 . 1 = 1	define OR and AND
A5	1 + 0 = 0 + 1 = 1	0 . 1 = 1 . 0 = 0	

Table 1. Axioms of switching algebra

Theorem No	Left Theorem	Right Theorem	Theorem Name
T1	a + 0 = a	a . 1 = a	Identities
T2	a + 1 = 1	a . 0 = 0	Null elements
T3	a + a = a	a . a = a	Idempotency
T4	(a')' = a		Involution
T5	a + a' = 1	a . a' = 0	Complements
T6	a + b = b + a	a . b = b . a	Commutativity
T7	(a + b) + c = a + (b + c)	(a . b) . c = a . (b . c)	Associativity
T8	a . (b + c) = a . b + a . c	a + (b . c) = (a + b) . (a + c)	Distributivity
T9	a + a . b = a	a . (a + b) = a	Covering
T10	a . b + a . b' = a	(a + b) . (a + b') = a	Combining
T11	(a + b)' = a' . b'	(a . b)' = a' + b'	DeMorgan's theorems

Table 2. Some frequently used theorems