ECE2883 HP: Lab 3- Logic Experts (LEs)

Combinational logic lab

Implementing combinational logic with Quartus

We should be starting to realize that you, the "SMEs" in this course, are just a specific type of SME – you are "logic experts." Everyone has relevant skills, but you have a background in logic. So this track of learning is now called LE, instead of SME.

In your previous logic course, a lot of material was covered, but only two major areas are essential to begin designing logic circuits of all kinds:

1. Combinational logic
2. Sequential logic

We have already touched upon combinational logic in both the seminar materials and in the previous Quartus exercise where you implemented the light-switch (exclusive-or) example as part of the tutorial. This exercise spans some of the material covered in Lab 2 and Lab 3 of ECE2031. Some notable differences here between your exercises and those of ECE2031 include:

- In ECE2031, students spend some time prototyping a circuit with discrete logic chips. Because you will get more experience with prototyping hardware later in the semester, with your project, we save some time by not working with chips now.
- In ECE2031, there is a strong emphasis on the collection of specific results in a particular format. Because ECE2883 has a larger component of technical communication, we will not require as much care with the checkoffs and results.

Optionally, read Chapter 2 of Harris and Harris, up to and including section 2.7.

K-maps and circuit design

Boolean equations can be reduced to simpler forms by algebraic manipulation, Karnaugh maps, or algorithms that are more suited for computer implementation. Karnaugh maps, or K-maps, are a useful tool for equations of 4 or fewer variables.

Exercise 1

Figure 1 shows a truth table for a four-variable Boolean function Y. Use your background in solving Karnaugh maps to find a minimal sum of products (SOP) expression for Y. Use the K-map provided on the answer sheet at the end of this document, and write the SOP expression there, too.

ABCD	Y
0000	1
0001	1
0010	1
0011	0
0100	0
0101	0
0110	0
0111	1
1000	1
1001	1
1010	1
1011	0
1100	0
1101	0
1110	0
1111	0

Figure 1. Truth table assignment.

Gate design

Now that you have a minimal SOP expression for Y, you can implement it with AND gates, OR gates, and inverters (NOT gates).

Exercise 2

On the answer sheet at the end of this document, draw the circuit for Y, using AND gates for the product terms and a single OR gate to produce Y.

When a bubble appears on a wire in a circuit, it has the same effect as an inverter. Two bubbles on the same wire cancel.

Exercise 3

On the answer sheet, redraw the same circuit, but put a bubble on the output of each AND gate (making them NAND gates), and in order to not change the implementation, put a corresponding bubble on the inputs of the OR gate (making it something we would call a bubbled-OR gate).

Note that the bubbled-OR gate, by DeMorgan's theorem, is identical to a NAND gate. Do not draw it as a NAND gate in the previous exercise, and keep it in this bubbled-OR format, because it is clearer this way. It shows that it is functioning to produce the "sum" (OR) of the product terms.

But the result of the previous exercise is that we now have a representation for the circuit that uses all NAND gates and inverters. NAND gates, NOR gates, and inverters can be implemented with fewer transistors than AND gates and OR gates (both in TTL logic and CMOS logic), so they are faster and occupy less space inside an integrated circuit.

Revisiting Quartus

In Lab 2, you followed a step-by-step tutorial to create a combinational logic circuit. Here, you will go through the same process with your new circuit. Refer to the Quartus tutorial from Lab 2 as needed, but you may find that you do not need it at all (http://powersof2.gatech.edu/2883HPC/curriculum/Lab2SME.pdf).

Exercise 4

Create a new project, as was done previously in Section B.1 of tutorial. Again, your target device is the EP2C35F672C6 Cyclone II FPGA.

Again, your project has only one design file, a Block Diagram File, or BDF. Create it as was done previously in Section B.2 of the tutorial, but this time draw your circuit from Exercise 3. Note that Quartus has NAND gates and bubbled-OR gates of the sizes that you will need.

When you have completed the diagram, using INPUT and OUTPUT devices for pins, make sure that it compiles. Then, make pin assignments for the pins (A, B, C, D, and Y). Pick any of the DE2 switches for inputs, and pick any DE2 LED for the output. Note that all DE2 pins are defined in the DE2 reference manual, which you can find at the DDL web site at http://powersof2.gatech.edu/resources/Altera/DE2_UserManual.pdf

In Quartus, make sure that "View… Pin Assignments" is checked, so that your chosen pin assignments show up in the BDF. Export the block diagram as a JPEG, or print as a PDF. Include a hardcopy with the answer sheet.

Number systems

You should have had some experience with binary numbers. Converting numbers between binary and decimal can be done by different procedures that are not especially fun, for example. In the long run, we typically use computers or calculators to do this for all but the smallest numbers. So rather than learn some efficient algorithm for conversion, focus instead on the underlying principle, so you could do it yourself on an exam. Specifically, if you want to convert a number to binary, just list all the powers of two that are smaller than the decimal number you are converting, and see which ones "fit" in your decimal number, starting with the largest.

Example:

If your decimal number were 1745, then list: 1, 2, 4, 8,16, 32, 64, 128, 256, 512, 1024. There's no point in listing 2048, the next number in the sequence, because it is bigger than the number we are converting.

Starting with the largest power of two in your list, write a 1 if you can fit it in the number you are converting, subtract it from the number, and repeat for the next-smaller power of two. (Write a 0 if you cannot fit it.) Go until you get to the last power of two.

1745 : 1024 - 1 (1024 "fits" in 1745, and after removing it, produces 721 for the next step)
721 : 512 - 1 (fits, producing 209)
209: 256 - 0 (does not fit, so still 209)
209: 128 - 1 (fits, producing 81)
81: 64 - 1 (fits, producing 17)
17: 32 - 0 (does not fit, so still 17)
17: 16 - 1 (fits, producing 1)
1: 8 - 0 (still 1)
1: 4 - 0 (still 1)
1: 2 - 0 (still 1)
1: 1 - 1 (fits, and done, since there are no more powers of two to consider)

Write the bits down in order from largest power to smallest: $11011010001_2 = 1745_{10}$.

Converting binary to decimal is usually more obvious, since the binary "places" have clear meaning, and it is like coming home to our "normal" decimal base system. The rightmost position is the ones place. The next one to the left is the twos place, then the fours place, and so on. So converting a number like the one we just found, 11011010001_2, is straightforward. There is 1 one, 0 twos, 0 fours, 0 eights, 1 sixteen, and so on:

$1(1)+0(2)+0(4)+0(8)+1(16)+0(32)+1(64)+1(128)+0(256)+1(512)+1(1024)$

$=1+16+64+128+512+1024$

$=1745_{10}$

Exercise 5

Convert 363_{10} to binary.

Exercise 6

Convert 1100110_2 to decimal.

Motivation and revisiting project-based learning

You have seen some abstract concepts in logic, and some very simple Boolean equations, and it may be unclear why this important or how it can be used.

This is where ECE2883 can really help. We can immediately see some applications relevant to the hardware components that we have been discussing in class. One of the candidate projects found by a classmate was the Waterfall Swing. If you need a refresher about how it worked, see https://youtu.be/p1uwQVtHHOQ . Let's pick out part of the swing, a cross section such as shown in Figure 3.

Here, the water is allowed to flow when a solenoid valve is opened, and the solenoid is controlled by a Boolean value V: V=1 opens the valve, V=0 closes the valve. There are a variety of ways that the position of the swing could be sense, but suppose there are several camera-like sensors that produce a Boolean value of 1 when the swing is passing over them. Figure 3 shows five such sensors, and in the exact situation shown, most of them are producing a value of 0 (e.g., B=0, C=0, …), but A is producing the value of 1.

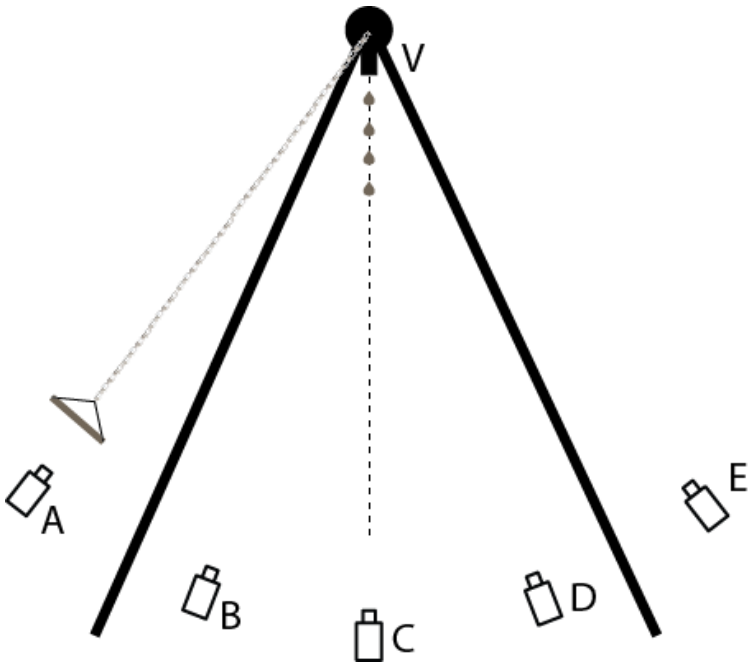


Figure 2.The Waterfall Swing could be implemented with a water valve and multiple sensors of swing position..

A simple way to control the valve, preventing water from flowing when the swing is passing through the low point, is

$$V = \bar{C}$$

In other words, V is true (open) whenever C is not true (does not sense a swing passing over it). Or V is false (closed) whenever C is true (DOES sense a swing passing over it).

This may be an inadequate solution. The swing is probably moving too fast for this strategy to stop the water soon enough. Because the water is already falling as the swing approaches the low point, the swing will be hit with water. One way to improve the solution is to stop the water whenever the swing is at the low point or even approaching the low point:

$$V = \overline{B + C + D}$$

In other words, V is false (closed) whenever either B or C or D is true.

Either of these solutions is trivial to implement with digital logic. The second solution, although better, is probably too conservative to make the swing very exciting. The water would be off most of the time. This example foreshadows the next major topic, sequential logic. Using sequential logic (and possibly more than just five sensors), it is possible to always keep track of the direction that the swing is moving, and only close the valve just early enough to stop it as the swing passes through the vertical. This is done by always using a history of sensor values, as opposed to just the current values.

Oscilloscope usage

As you begin working with project hardware, you will have occasion to use the lab oscilloscopes to make measurements and debug. A video tutorial is available for you to learn more about them.

Exercise 4

Watch the video at http://powersof2.gatech.edu/2031/labs.html .

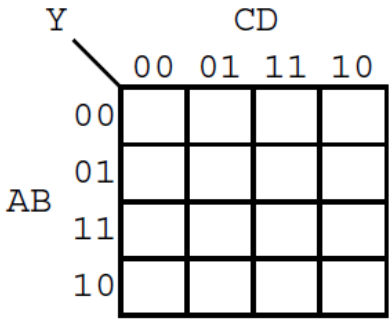
Results to turn in

The answers below are due by our seminar session on Thursday, September 17.

Exercise 1

Find a minimal sum of products expression for the function Y, given in Figure 1 and repeated here.

ABCD	Y
0000	1
0001	1
0010	1
0011	0
0100	0
0101	0
0110	0
0111	1
1000	1
1001	1
1010	1
1011	0
1100	0
1101	0
1110	0
1111	0



Y =

Exercise 2

Draw the gate diagram with ANDs and ORs:

Exercise 3

Draw the gate diagram with bubbles, as described.

Exercise 4

Attach a hardcopy of your Quartus BDF for the same circuit.

Exercise 5

363_{10} =

Exercise 6

1100110_2 =