

EXPERIMENT

3

COMBINATIONAL LOGIC CIRCUITS

VERSION F07

In this experiment, you will design, implement and verify two combinational logic circuits. One circuit implements a very simple logic function and the other implements a BCD-to-7-segment code converter. After performing this experiment, you should be able to:

- 1) Design combinational circuits for implementation in an FPGA,
- 2) Use simulation to verify circuit function,
- 3) Use timing analysis and simulation to identify a worst case delay path,
- 4) Use an FPGA-prototyping system to verify circuit function, and
- 5) Perform circuit-level propagation delay measurements

3-1 PRELAB

FPGA IMPLEMENTATION AND DELAY

In the Altera Cyclone technology which we are using, logic functions are decomposed by the implementation tools into 4-input subfunctions. (Even though you are drawing gates in your schematic, what you see is NOT what you get!) Each subfunction is then implemented by a truth table defined by programming the SRAM that defines the FPGA's functionality. Each of the truth tables has a delay which contributes to the delay down one or more paths from inputs to outputs of the function being implemented. In addition, the connections in the FPGA between the inputs, truth tables, and outputs pass through buffers, multiplexers and pass transistors as determined by the circuit specification and the routing paths determined by the implementation tools.

The decomposition into subfunctions combined with the routing of the interconnections between them yields considerable uncertainty in the propagation delay from input to output of an implemented circuit. A sophisticated user of the tools may use constraints to specify the maximum delays allowable, forcing the tools to attempt to meet or better these delays. Whether or not delays are specified, it is useful to know either whether delay specifications are met or some measure of the delays present. Since most combinational circuits are placed in a sequential environment, there is usually interest in the worst case delay that can occur in the operation of the circuit from any combinational logic input to any combinational logic output (equivalently, in the sequential circuit: from input to output, from input to flip-flop, from flip-flop to flip-flop, and from flip-flop to output). Worst case delay estimates are determined by adding up the maximum expected delays through the combinational circuit including both logic and interconnections. Because of the uncertainty discussed in the previous paragraph, worst case delay can be known only after the implementation process has been completed including the decomposition into subfunctions and the interconnect routing.

In this experiment, we will construct two circuits to be discussed later. With first circuit we will investigate some of

the types of simulation available to us, and see the possible effects of the decomposition process that implements our actual physical circuit. With the second circuit, in addition to verifying its functionality, we will determine the worst case delay from any input to any output and will use a very crude simulation method applicable only to small circuits to find the actual path along which this delay occurs. Knowing the path for the worst case delay, we can then measure the delay along this path in the lab. This measured delay will not be worst case but will be close to what is referred to as typical delay, which can be loosely viewed as the mean expected delay for the path in the actual implementation.

SIMULATION AND IMPLEMENTATION OF A SIMPLE CIRCUIT

We will implement the simple circuit shown below in Figure 1. (Note that when creating this circuit in the FPGA, we will need to add output pads to get Y and XBAR out of the FPGA, and an input pad to get X into the FPGA).

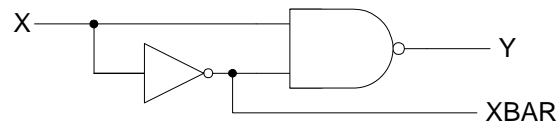


Figure 1 Simple Circuit

1. Give the truth table for this circuit.
2. Assume that this circuit was constructed by interconnecting real devices (an integrated circuit inverter and NAND gate). If we applied a square wave signal to X, what would output Y look like?

BCD-TO-7-SEGMENT CODE CONVERTER

The BCD-to-7-segment code converter (commonly referred to as a BCD-to-7-segment decoder) is used to convert a binary coded decimal (BCD) value to the appropriate segment pattern for a 7-segment display. Since BCD values are 4-bit numbers in the range 0-9, how we treat the remaining possible values of 10-15 will impact our design. While we might consider those values to be ‘don’t-cares’, simplifying the logic required compared to making the display blank for those values, for this lab you will design the decoder so that it displays a specific pattern for all non-BCD values. The segments in the display are typically identified in industry by the letters a-g, corresponding to the segments shown below.

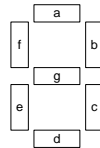


Figure 2 Seven-Segment Display Convention

So, the input to your circuit will be a 4-bit number (label these signals D3, D2, D1, D0; with D3 being the most significant bit), and there will be 7 outputs (label these signals A, B, C, D, E, F, G to correspond to the segment pattern shown above) that will light the display in the patterns shown below in Figure 3 for all valid inputs. **All invalid inputs shall be considered as don’t-care conditions in the logic design.**

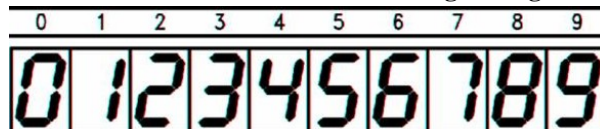


Figure 3 Seven-Segment Display Patterns

To make our BCD-to-7-segment decoder design easier to use and to easily create multiple instances of it, we will also embed it in a hierarchy block, and then make connections to the block in order to test our design.

CIRCUIT DESIGN

In this prelab, you will create two projects, the first will be called *simple* and contain only a single schematic, the second will be called *bcd* and will require two schematic sheets.

Step-by-step instructions are no longer given here as they were in the tutorial-like Experiment 2. Please refer to Experiment 2 as necessary for details of the steps. It is better to check than have lots of problems!

FILE MANAGEMENT

- 1) On your network drive, create a new directory **Lab3**, and under it create directories **simple** and **bcd**. Download the file `ece351_pin_assignments.zip` from the course web page, and extract the file `ece351_pin_assignments.qsf` to the Lab3 directory. You can open this file in Notepad, and note that it contains pin name assignments for the FPGA that match the programmer's model. It also establishes the configuration scheme to be used by the FPGA and the FPGA speed grade. We will import this into our projects from now on so that we can simply select pins by name.

PROJECT SIMPLE CREATION

- 1) Create a new Quartus project named **simple** in the **simple** directory.
- 2) To get the pin assignments, open the file `ece351_pin_assignments.qsf` in Quartus. Select the file text in its entirety, and copy. Then, open the file `simple.qsf` in Quartus (it is in the **simple** directory), and paste the text into it. Save the `simple.qsf` file, and then close both files. Now, to check to see if the pin assignment import worked, select **Assignments** → **Pins**, and verify that the pin assignments shown in the Assignment Editor match those shown in the programmer's model.
- 3) Create a new schematic diagram file named `simple.bdf`, add it to your project, and implement the circuit shown in Figure 1. Name the input pin CLK0 to connect the 2.00MHz clock as the input. Name the output pin connected to the NAND gate (Y in Figure 1) BAR1_1, and name the output pin connected to the inverter (XBAR in Figure 1) BAR1_2.

PROJECT COMPILATION AND SIMULATION

- 1) Compile your project. If there are errors, fix and recompile before proceeding. You will see a large number of warnings – the following warnings can be ignored;
 - a. **Warning: Output pins are stuck at VCC or GND** – this warning is due to the synthesis process deciding that our circuit output Y should always be a one.
 - b. **Warning: Following nodes are assigned to locations or regions, but do not exist in design** - this warning is due to the fact that we imported assignments for all of the FPGA pins, but only used 3 of them.
 - c. **Warning: Following 1 pins have nothing, GND, or VCC driving datain port -- changes to this connectivity may change fitting results** – this warning is due to the synthesis process deciding that our circuit output Y should always be a one.
- 2) First, we will **functionally** simulate the circuit. Start the simulator, and ensure that the simulation mode is set to **Functional**. Select and add X, XBAR, and Y as nodes, placing them in logical order in the Waveform Viewer. Use **Overwrite Clock** in the **Stimulators** panel (on the left side of the waveform window) as X and set its period to 10ns.
- 3) **Save** the simulation state.
- 4) Print out your simulation report on a **single** page in **Landscape**.
- 5) Now, change the simulation mode to **Timing**, and run the simulation again.
- 6) Print out your simulation report on a **single** page in **Landscape**.
- 7) Answer the following questions

3. What is the difference between the two simulation modes and what they represent?
4. Which one would more accurately model how we would expect the circuit to behave when implemented on the FPGA?
5. Would you expect the likely actual FPGA performance to be better, worse, or the same as the answer you selected for number 4, and why?

PROJECT BCD CREATION

- 1) Create a new Quartus project named **bcd** in the **bcd** directory.
- 2) Import the pin assignments as described previously.
- 3) Create two new schematic diagram files named **bcd_7seg.bdf** (do NOT add this file to your project!) and **bcd_test.bdf**.

SCHEMATIC CREATION

- 1) Design the BCD-to-7-segment decoder circuit as specified above by developing a truth table, creating K-maps, and finally extract the minimal sum-of-products Boolean equations. The inputs to the LED displays are active high, that is, a 1 will turn ON the display segment and a 0 will turn it OFF. Use the truth table and K-map blanks provided for you at the end of this document, paying careful attention to the preprinted definition of variables on the maps.

6. Develop the truth table for your BCD-to-7-segment decoder.
7. Develop the K-maps for your design, and determine the required prime implicants.
8. Write the Boolean equations for the seven output functions.

- 2) **Sheet BCD_7SEG:** On sheet BCD_7SEG, you are to implement your BCD-to-7-segment decoder circuit that you designed as specified above. Also, rather than using inverters, you can use n-input gates from the library having n bubbled (inverted) inputs. For example, the gate BAND3 has three inputs that are inverted by bubbles, permitting implementation of terms such as $\overline{X_2} \bullet \overline{X_1} \bullet \overline{X_0}$ without using inverters. Use input or output pins as appropriate, setting the pin names to be D3-D0 and A-G as appropriate. When your circuit is complete, select **File**→**Create/Update**→**Create Symbol Files for Current File**. Now, your BCD-to-7-segment decoder will be available to use as a component in other schematics. (It has become a part of the library components for this project.)
- 3) **Sheet BCD_TEST:** On Sheet BCD_TEST, you are to design a circuit that uses the **BCD_7SEG** block, and that connects the inputs to the DIP switches S1_1 - S1_4 and the outputs to the LED display 1 (DIS1) on the FPGA board. Place your **BCD_7SEG** block (it will be available in the **Symbol Toolbox** window under the library **PROJECT**), and then connect the inputs and outputs as listed below. Name the pins as shown below. Do the same for the outputs. This will assign pin numbers to those nets automatically using the pin locations you imported into the project. This allows us to specify physical pins by using names that make sense to us, instead of having to look up the pin numbers each time.

<u>Pin connected to...</u>	<u>Use net name</u>
D3	S1_1
D2	S1_2
D1	S1_3
D0	S1_4
A	DIS1_A
B	DIS1_B
C	DIS1_C
D	DIS1_D
E	DIS1_E
F	DIS1_F
G	DIS1_G

NETLIST GENERATION AND FUNCTIONAL SIMULATION

- 1) Set the current sheet (BCD_TEST) as the Top level entity in the hierarchy. This can be done by selecting **Project** → **Set as Top-Level Entity**.
- 2) Functionally simulate the circuit for all possible input combinations: Select and add S1_4, S1_3, S1_2, S1_1, and DIS1_A through DIS1_G as signals. Select individual signals and place them in logical order in the **Waveform Editor**. Use **Overwrite Clock** in the **Tools** panel on the left for S1_1 and set its period to 160ns. Similarly add Clock simulators to S1_2 with 80ns period, S1_3 with 40ns period, and S1_4

with 20ns period. In the Simulation Tool, enter 160ns for the simulation interval so that you will include all 16 combinations of S1_1 through S1_4.

- 3) **Save** the simulation state.
- 4) Answer the following questions.

9. Is your decoder functioning correctly? Explain how you determined this.

- 5) If the simulation is correct, go back and print the necessary copies of the schematic sheets. If not, correct the design and re-simulate and print later.
- 6) Annotate the simulation so that it is clearly shown that the circuit functions correctly.
- 7) Print out your simulation on one page. Be careful to not print out large number of pages of output and to use **Landscape**.

IMPLEMENTATION AND REPORT ANALYSIS

- 1) Click on **Start Compilation** from the **Processing** menu. This implements the overall design.
- 2) Check the **Pin-out File** to make sure that all pins have been assigned correctly.

TIMING SIMULATION

- 1) Perform a timing simulation on the design using the same signals and waveforms you used for the functional simulation above. Make sure the simulation is set for **Timing** and not **Functional**.
- 2) Execute the simulation and verify that the outputs (except for some delays and glitches) are identical to those for the functional simulation.
- 3) Do not close the timing simulation window.

WORSE CASE DELAY SIMULATION

- 1) Examine the **Timing Analyzer Summary** and **tpd** tabs of the **Compilation Report** for the **maximum delay** from any of the inputs, S1_3, S1_2, S1_1 and S1_0 to any of the outputs, DIS1_A through DIS1_G. The input for this maximum delay will be referred to as the **path source**, Xi, and the output will be the **path destination**, Yj.
- 2) By examining the truth table for function Yj, find **all** value sets for the three input variables other than Xi, such that when Xi changes between 0 and 1, Yj also changes value. For example, if Xi is S1_3 and truth table rows S1_3 = 1, S1_2 = 1, S1_1 = 0, S1_0 = 1, Yj = 0 and S1_3 = 0, S1_2 = 1, S1_1 = 0, S1_0 = 1, Yj = 1 appear, then S1_2, S1_1, S1_0 = (1,0,1) is a value set. Since a change in Xi causes Yj to change when this value set is applied, the delay can be measured from Xi to Yj. If there are no such value sets, then the delay identified cannot be reliably measured since the value before and after the delay is the same. In this case, pick the next longest delay and repeat the process for finding value sets.

Provide answers to the following.

10. What is the maximum delay for the decoder?
11. What input Xi is the path source?
12. What output Yj is the path destination?
13. What are all input variable sets and their values for which a change in Xi as determined above produces a change in Yj?

- 3) Perform a new timing simulation for all inputs and output Yj only, applying each of the input variable set values in succession with the path source Xi changing from 0 to 1 to 0 for each value set. This will let us identify the delays that correspond to each possible path.
- 4) Open the **Simulation Report**. Measure the delays from Xi to Yj for each input variable set. The simplest way to do this is to move the time bar so that it is at the point where the input Xi changes, then place the mouse cursor on the point where the output changes and read the interval at the top of the window. Be sure to zoom in on the waveform to get an accurate answer.
- 5) Find the maximum delay time that is measured and identify the value set for which it occurs. This pair is the one that causes the signal to propagate down the longest path from Xi to Yj.

Provide answers to the following.

14. What is the maximum delay from the simulation?

15. How does it compare with the maximum delay from the Post Layout Timing Report?

16. For what value set does the maximum delay appear? (If there are ties in the delay value, arbitrarily select a set.)

The value set and delay value found here are to be saved for use in the Lab Work.

PRELAB WRITE-UP

The prelab write-up is to contain all of the printouts requested and answers to all of the questions, both in the order presented.

3-2 LAB WORK

WARNING: All lab results and all answers to questions or discussion are to appear in the lab reports of individual students. All tangible lab results are to be identical (unless indicated otherwise). When a printout of results is specified, a copy should be made for each team member. All answers to questions or discussion are to be the work of individual students, not the lab team. Evidence of collaboration on these aspects of a report within or between teams will be noted and is subject to University disciplinary action.

EQUIPMENT NEEDED

In addition to the equipment already on the lab bench, your instructor will have you check out a plastic tray containing:

- 1) an Altera Cyclone FPGA board.
- 2) a power supply module for the FPGA board,
- 3) two scope probes,
- 4) a logic analyzer probe,
- 5) four black logic analyzer extension wires in a plastic bag, and
- 6) several blue connection leads.

Warning – Lab Equipment Handling: Much of the lab equipment is small and delicate. In particular, this applies to the XESS prototyping boards, scopes, scope probes, and logic analyzer probes. So please be careful and handle the equipment with a light and careful touch and do not use the scope probes with the grabbers removed. Perform wiring on the board **ONLY** with the power disconnected.

PROJECT SIMPLE

TESTING SET UP

In the first part of the lab, you will implement the simple circuit that you created and simulated for the prelab.

- 1) Start **Quartus II** and load your project *simple*.
- 2) Make sure the board power and parallel cable are disconnected.
- 3) Implement your design by clicking the **Start Compilation** icon.
- 4) When the implementation completes, look at the **Pin-out File** and verify that all pins have been assigned correctly.
- 5) On the oscilloscope, deactivate all analog channels and activate digital channels **D0-D7** and individually deactivate **D3-D7**.
- 6) Attach the logic analyzer GND leads to the GROUND pins on the FPGA board.
- 7) Make sure you have your schematic printout available for reference. Connect the oscilloscope channels to the corresponding pins on the FPGA board headers as follows.

D0	CLKIN	X
D1	BAR1_2	XBAR
D2	BAR1_1	Y

- 8) Set up the signal labels on the three channels.
- 9) Make sure the threshold voltage separating 0 and 1 is set for **TTL** for both **D0-D7** and **D8-D15**.
- 10) Perform the POWER UP sequence. Remember to attach the parallel port cable.

FUNCTIONAL TEST

In this step, we will load the simple.sof file into the FPGA and test the circuit's operation.

- 1) Configure the FPGA by running **Tools→Programmer**, click on the file displayed (should be pointing to simple.sof). You can also browse for the file from the **Add File...** pushbutton. Remember to do the **Hardware Setup** for the **ByteBlaster [LPT1]**, and use **Passive Serial** mode. Verify that the red LED on the FPGA board goes out after configuration – if it does not, the FPGA is NOT configured!
- 2) Adjust the timebase so that the clock waveform on X appears on the screen aligned with the grid. Use the scope **Vernier** to do this. (Recall that the **Vernier** can be turned on using the **Main/Delayed** menu.). Scan across the waveform to answer the following.

1. Are the outputs **XBAR** and **Y** as expected from your earlier pre-implementation simulations?
2. What happened to output **Y**? What does that tell you about how the synthesis tools work when they implement your schematic design?

- 3) **CHECKPOINT: Have your instructor observe the waveforms.**

PROJECT BCD

PERFORMING FUNCTIONAL TEST

First, we will apply inputs to the BCD decoder using the DIP switches and observe the seven segment display to verify the functional correctness of your decoder. First, we will load the bcd_init.sof file into the FPGA.

- 1) In Quartus, open your **bcd** project. Configure the FPGA by running **Tools→Programmer**.
- 2) Connect the D3-D0 inputs of the oscilloscope to the FPGA board header pins corresponding to S1_1 - S1_4 so that you can verify the actual input to the FPGA.
- 3) Test your circuit using all possible input combinations and observing the LED display DIS1.
- 4) **CHECKPOINT: Have your instructor observe the output of your decoder.**

DECODER DELAY TEST

We now want to evaluate the worst case time delay through the decoder that we observed during timing simulation. Because there are only four input variables and three of them are fixed, this will be greatly simplified. We will use a clock signal to stimulate the input with the longest delay, and the DIP switches to set the other inputs.

- 1) Perform the POWER DOWN sequence.
- 2) Set the DIP switch corresponding to variable Xi identified in the timing simulation as the source of a longest path to the **OFF** position. Then, connect the oscillator signal (**CLK0**) output to the Xi input by using a blue jumper from the CLKIN pin of JP12 to the pin on JP7 corresponding to the Xi DIP switch. (Refer to the Board Locations document if you can't find these on the FPGA board.) Ensure that the DIP switch corresponding to variable Xi remains in the **OFF** position as long as this connection is made.
- 3) Set the remaining DIP switches to give you the value set that you identified in the prelab that is associated with the worst case delay.
- 4) Connect the ground clips from the scope probes to the GROUND header on the FPGA board.
- 5) Connect the oscilloscope channel A1 probe to CLKIN pin of JP12.
- 6) Connect the oscilloscope channel A2 probe to Yj, the variable that is the destination for the longest path. The signals to the displays are all available on JP8.

- 7) Perform the POWER UP sequence and reconfigure the FPGA.
- 8) Set up the scope for a delay measurement from a change in Xi to a change in Yj as you did in Exp#1.
- 9) Perform the delay measurements for Yj going from 0 to 1 and from 1 to 0 and determine the largest value using a 50% level measurement. (Note: The 50% level may be different for the input and output signals.)
- 10) CHECKPOINT: Have your instructor observe the waveforms for the delay measurement.**
- 11) **Print** out the waveforms for the delay measurement.
- 12) Answer the following questions.

3. What is the maximum measured delay?
4. Compare the maximum measured delay with that from the timing simulation. It is expected that the measured delay will be smaller. Is it? Why is it expected to be smaller?

- 13) Perform the POWER DOWN sequence.
- 14) Very carefully remove all leads including the power from the board. Put the board and the power supply in the bottom of the tray and place the three probes and the bag of leads carefully on top.
- 15) Your instructor will check the tray back in.

CONCLUSIONS

Comment on Delay: With respect to delay, if one has specific needs to control delay in all or part of an FPGA, then one should not depend on the implementation tools to handle it without directing them. In order to specify that all or certain parts of the design have upper bounds on delay, constraints can be placed on the synthesis tools. When such delay constrained designs are done, place and route are typically more difficult and, for complex, tightly-constrained circuits, the process can become quite long.

Comment on Schematics: This lab should have vividly demonstrated that what you see is not necessarily what you get, at least when you draw a schematic. Always bear in mind that the FPGA does not have gates that are simply interconnected as you draw them, rather your schematic design must be converted to the resources actually available on the FPGA. In this mapping process, the synthesis tools will attempt to optimize your design. However, if your design is delay-dependent (as the **simple** project was) and the tools perform their optimization without regard to delay (as they did here); then the optimization can lead to undesired and/or unexpected results. As an engineer, you will need to always be aware of exactly what your tools are doing for you (or to you, as the case may be). (Note that post-implementation simulation did reveal the problem without implementing it.)

As your reward for completing the lab, turn off all traces and the grid on the scope, select **Print/Utility** and then simultaneously press the second and third buttons from the left below the scope screen.

REPORT

Your individual report on 8.5 by 11 inch pages should follow the report format available on the course web page and consist of:

- 1) A cover with the experiment number, experiment name, your name, name(s) of other team member(s), and your instructor's name, your lab section, the number on the equipment tray used, and name on the PC used.
- 2) Description of Laboratory Exercise and Conclusions as shown in the report format.
- 3) Your individual answers to all of the questions given in the green-bordered boxes (online version has color).
- 4) Copies of all printouts for the lab properly annotated and in the order produced, including
 - a. sheets simple, bcd_7seg, and bcd_test
 - b. output sheets for the functional simulation and for the worst case delay simulation of the BCD decoder
 - c. sheet(s) showing the delay measurements for the BCD decoder.

REFERENCES

1. MANO, M. M AND C. R. KIME. *Logic and Computer Design Fundamentals*, 2nd and 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2000 and 2003.

K-map for segment ____

		D1 D0			
		00	01	11	10
D3 D2	00				
	01				
	11				
	10				

K-map for segment ____

		D1 D0			
		00	01	11	10
D3 D2	00				
	01				
	11				
	10				

K-map for segment ____

		D1 D0			
		00	01	11	10
D3 D2	00				
	01				
	11				
	10				

K-map for segment ____

		D1 D0			
		00	01	11	10
D3 D2	00				
	01				
	11				
	10				

K-map for segment ____

		D1 D0			
		00	01	11	10
D3 D2	00				
	01				
	11				
	10				

K-map for segment ____

		D1 D0			
		00	01	11	10
D3 D2	00				
	01				
	11				
	10				

K-map for segment ____

		D1 D0			
		00	01	11	10
D3 D2	00				
	01				
	11				
	10				

K-map for segment ____

		D1 D0			
		00	01	11	10
D3 D2	00				
	01				
	11				
	10				