Lab EE 271: Circuit Theory

Winter 2018

Lab 3

Shawnna Cabanday
Professor Rania Hussein
Lab Section: Tuesday from 1:15 - 3:15
Date submitted: January 24, 2018

Abstract

The purpose of this lab was to further implement Verilog programming to create programmable circuit designs in Quartus II. The first task of this lab was to create a multi-level logic circuit on Quartus that would implement a seven segment display on the DE1-SOC board. The second task involved developing an electronic detector device that determined which items were on sale, what items were stolen and were attempting to be returned based on a three digit code and a special marking, and a hex display was created for each custom item.

Introduction

Digital logic circuits handle data encoded in binary form through signals that have only two values, 0 and 1. This binary logic that deals with determining whether or not input conditions evaluate to an output of either "true" and "false" allows scientists and engineers to develop complex digital logic circuits and computers that can be built using a few types of basic circuits called gates, each performing a single elementary logic operation. The purpose of this lab is to continue to practice how Verilog programming can be used to produce complex binary logic circuits virtually, which can then be practiced physically on the FPGA. A verilog program will then be created based on simplified expressions and truth tables, which will be verified using generated waveforms by ModelSim and then downloaded to a Field Programmable Gate Array (FPGA) to demonstrate the physical program. The first task of this lab was to create a multi-level logic circuit on Quartus that would implement a seven segment display on the DE1-SOC board. The second task involved developing an electronic detector device that determined which items were on sale, what items were stolen and were attempting to be returned based on a three digit code and a special marking. Finally, a hex display was created for each custom item.

Materials

- Altera Quartus II Lite Program
- (5CSEMA5F31) DE1-SoC FPGA Development Board
- Ethernet to USB Cable
- Power cord for FPGA

Procedures

Part 1: Seven-Segment Display

The first experiment involved designing a multi-level logic driver code on Quartus that would drive the seven segment display on the DE1-SoC board. The goal of the circuit was to have two of the 6 HEX displays operate and display decimal numbers from 0-9 based on four 1-bit inputs controlled by the switches on the FPGA.

A module titled "segment7" was created as the driver code to develop the specific cases for inputted binary number, or the binary coded decimal (BCD). The code had one 4-bit input, which collected information from the four 1-bit input switches, in addition to an output connected to the LEDs, which were 7-bit outputs for each LED strip for a specific HEX display.

To create the design on the HEX display, a case for each BCD, in addition to a default case for "don't care" scenarios, had to be designed by inputting a 7-bit code consisting of 0's and 1's for each bit, with each 0 and 1 either turning on or powering off a particular LED strip in the FPGA. Turning on a specific LED strip required that a 0 be inputted as code within the 7-bit text, which is opposite to what would be originally expected. This is because the FPGA operates in a reverse functionality provided that it operates in inverters. Furthermore, the don't care scenario was indicated by the code "7'bx" which would allow the board to display anything not directly assigned by the Verilog program.

The pin assignment for a hex display is based in order from 1-6. Specifically, the organization pattern would be 7'0123456, and 7'b000001 would represent a decimal number of 0 on the hex display. Figure 1.1 illustrates the specific LED strip assignment with respect to the code pattern mentioned above.

A second module within the same verilog file was created to instantiate the "segment7" code. The module had a total of four 1-bit inputs labeled as SW and 7-bit output connected to the HEX0 display. The "digit0" calls the method "segment7" and inputs the on-off characteristics of each assigned switch and calculates the output needed to be pictured on the HEX0 display. This task to operate one HEX display was repeated on the adjacent HEX display. Powering two displays at once required an additional output to HEX1, and additional line of code was created to instantiate the "segment7" method to the second digit.

Input and output pins were mapped according to Table 1.1 and Table 1.2. Finally, the ".sof" file created from the Verilog script and pin assignment was downloaded to the FPGA by powering the board, connecting it to computer with a USB to Ethernet cable, and adding the file to the FPGA data platform. Figure 1.2 demonstrates the HEX0 and HEX1 output.

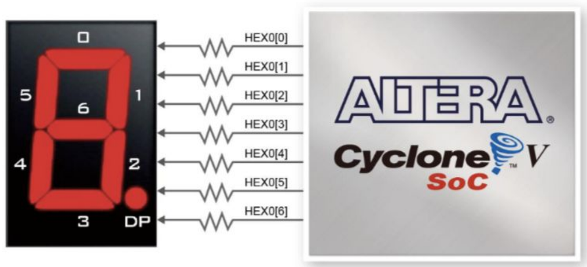


Figure 1.1: HEX Display Pin Assignment

Figure 1.2: HEX0 and HEX1 Output Example

Table 1.1: Input Pin Assignment

Input		Pin Placement
Switches for HEX0 Display	SW0[3]	PIN_AF10
	SW0[2]	PIN_AF9
	SW0[2]	PIN_AC12
	SW0[1]	PIN_AB12
Switches for HEX1 Display	SW1[7]	PIN_AC9
	SW1[6]	PIN_AE11
	SW1[5]	PIN_AD12
	SW1[4]	PIN_AD11

Table 1.2: Output Pin Assignment

Output		Pin Placement
HEX0 Display	HEX0[0]	PIN_AE26
	HEX0[1]	PIN_AE27
	HEX0[2]	PIN_AE28
	HEX0[3]	PIN_AG27
	HEX0[4]	PIN_AF28
	HEX0[5]	PIN_AG28
	HEX0[6]	PIN_AH28
HEX1 Display	HEX1[0]	PIN_AJ29
	HEX1[1]	PIN_AH29
	HEX1[2]	PIN_AH30
	HEX1[3]	PIN_AG30
	HEX1[4]	PIN_AF29
	HEX1[5]	PIN_AF30
	HEX1[6]	PIN_AD27

Part 2: Multi-level logic on the DE1-SoC Board

The second experiment required developing an electronic detector device that determined which items were on sale, what items were stolen and were attempting to be returned based on a inputted UPC code consisting of three binary digits. The electronic detector device also determined if there was a special mark on it and it created a hex display for each custom item "scanned". The same procedures described in part 1 were reproduced for this experiment, specifically verilog programming in Quartus Prime II and downloading the verilog file to the FPGA for physical tests. The circuit would have one output representing the sale light, which would light up whenever a sale item's UPC code is applied, another light representing a stolen item, which would light up when the item was shoplifted. There are four cases for the stolen light logic to consider when determining whether or not it should light up:

- An expensive item with the mark is not stolen.
- An expensive item without the mark is stolen.
- A non-expensive item without the mark is not stolen.
- A non-expensive item with the mark will never occur, so it is a Don't Care.

The behavior for items that were not inputted as real items represented in the provided lab table did not matter in this circuit. A boolean equation was built out of the truth table using K-Maps. Input and output pins were mapped according to Table 2.2 and Table 2.3.

As for the custom items that were powered onto the HEX displays, there were six cases to consider:

- Glasses
- Top Hat
- Water bottles
- Pistol
- Laughing Gas
- Husky T-shirt
- In addition to a final default item that represented a random output to the HEX display.

Table 2.1 illustrates the BCDs needed to be inputted using the four 1-bit switches to create the displays. The switches on the board were in order of M, U, P, and C.

To compile the verilog code, there was a process of three steps:

1. Creating the inputs and outputs for the driver code, which includes switches, HEX displays from 0-4, and sale and stolen outputs.
2. Creating separate modules to establish the case statements for each item (i.e. glasses, top hat, etc.), with one module being specific to each HEX display.
3. Instantiating the HEX displays using the created modules.
4. Instantiating the Sale and Stolen LEDs using the prior code from lab 2.

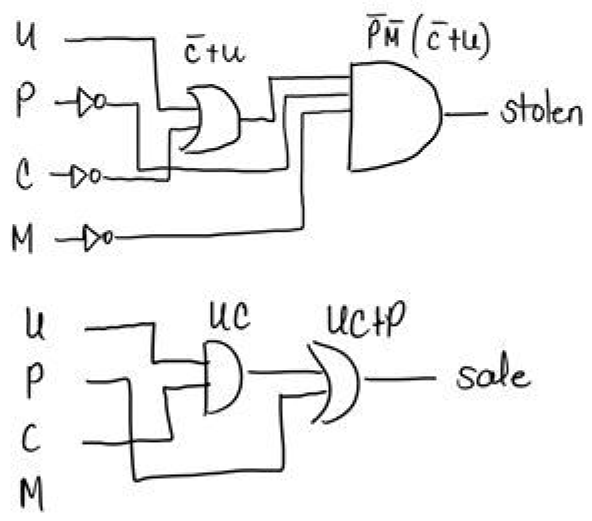


Figure 2.1: Electronic Detector Device Circuit Design

$$Sale = UC + P$$

Equation 2.1: Boolean Equation for Sale Output

$$Stolen = \overline{P}\,\overline{M}\,(\overline{C} + U)$$

Equation 2.2: Boolean Equation for Stolen Output

Table 2.1: Electric Detector Device Truth Table

Item	INPUTS				OUTPUT		Expensive?
	M	U	P	C	SALE	STOLEN	
Glasses	0	0	0	0	0	1	1
	1	0	0	0	0	0	
Top Hat	0	0	0	1	0	0	0
	1	0	0	1	0	X	
Water Bottles	0	0	1	1	1	0	0
	1	0	1	1	1	X	
Pistol	0	1	0	0	0	1	1
	1	1	0	0	0	0	
Laughing Gas	0	1	0	1	1	1	1
	1	1	0	1	1	0	
Husky T-shirt	0	1	1	0	1	0	0
	1	1	1	0	1	X	

Table 2.2: Input Pin Assignment

Input	Pin Placement
U	PIN_AF10
P	PIN_AF9
C	PIN_AC12
M	PIN_AB12

Table 2.3: Output Pin Assignment

Output	Pin Placement	Output	Pin Placement
Sale	PIN_W16	HEX3[0]	PIN_AD26
Stolen	PIN_V16	HEX3[1]	PIN_AC27
HEX0[0]	PIN_AE26	HEX3[2]	PIN_AD25
HEX0[1]	PIN_AE27	HEX3[3]	PIN_AC25
HEX0[2]	PIN_AE28	HEX3[4]	PIN_AB28
HEX0[3]	PIN_AG27	HEX3[5]	PIN_AB25
HEX0[4]	PIN_AF28	HEX3[6]	PIN_AB22
HEX0[5]	PIN_AG28	HEX4[0]	PIN_AA24
HEX0[6]	PIN_AH28	HEX4[1]	PIN_Y23
HEX1[0]	PIN_AJ29	HEX4[2]	PIN_Y24
HEX1[1]	PIN_AH29	HEX4[3]	PIN_W22
HEX1[2]	PIN_AH30	HEX4[4]	PIN_W24
HEX1[3]	PIN_AG30	HEX4[5]	PIN_V23
HEX1[4]	PIN_AF29	HEX4[6]	PIN_W25
HEX1[5]	PIN_AF30		
HEX1[6]	PIN_AD27		
HEX2[0]	PIN_AB23		
HEX2[1]	PIN_AE29		
HEX2[2]	PIN_AD29		
HEX2[3]	PIN_AC28		
HEX2[4]	PIN_AD30		
HEX2[5]	PIN_AC29		
HEX2[6]	PIN_AC30		

Results

The first experiment that designed a multi-level logic driver code on Quartus to drive the seven segment display on the DE1-SoC board did not operate correctly initially. After further troubleshooting and evaluating the pin assignments, the logic driver code compiled accurately and the pin assignments were correct, however, the HEX displays were still failing to light up at specific pins. Figure 3.1 and 3.2 demonstrate the lack of certain LED power strips to create the number ‘8’ or ‘0’. The FPGA board was determined to be the issue and after replacing it, the code and board behaved properly according to expectations outlined by the truth table.

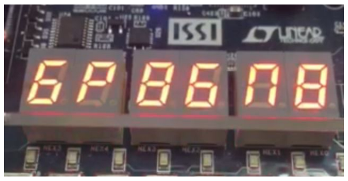


Figure 3.1: Failing to Power up ‘8’

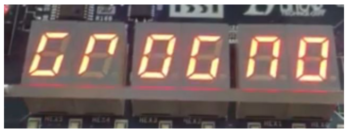


Figure 3.2: Failing to Power up ‘0’

Similarly, the second experiment that designed the UPC electronic device did not operate correctly initially. After troubleshooting, recompiling code, reassigning the pins, and assuring that the 7-bit hex code was correctly inputted into Verilog, the FPGA was responding according to the outlined truth tables. Figure 3.3 and 3.4 illustrate two of the six outputs designed for the items to be recognized by the UPC electronic device. Figure 3.3 represents a top hat and correctly corresponded to an input code of both 0001 and 1001. Figure 3.4 represents the laughing gas item and correctly corresponded to an input codes 0101 and 1101. For cases where the BCD input was 2 and 7, the FPGA responded with don’t care scenarios and hex displays that were not designed. Figure 3.5 and 3.6 illustrate those responses, respectively.

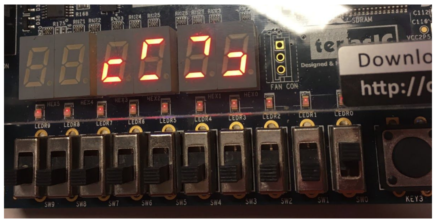


Figure 3.3: Top Hat Item

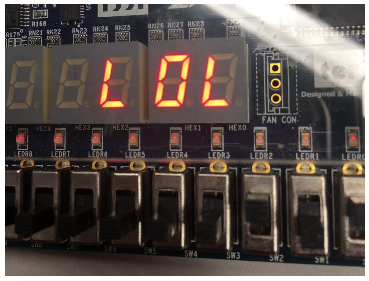


Figure 3.4: Laughing Gas Item

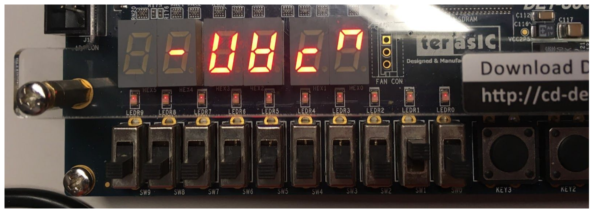


Figure 3.5: Don't Care Output for BCD 2

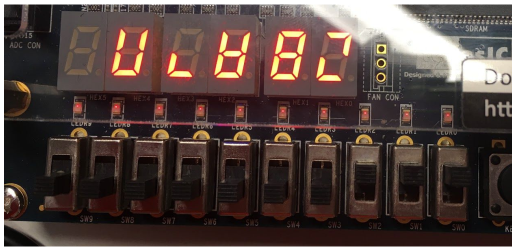


Figure 3.6: Don't Care Output for BCD 7

The UPC codes for each HEX display follow that :

- Glasses were displayed for codes 0000 and 1000
- A top hat was displayed for codes 0001 and 1001
- A water bottle indicated by the text 'H20' was displayed for codes 0011 and 1011
- A pistol and a bullet were displayed for codes 0100 and 1100
- Laughing gas was displayed by the text 'LOL' for codes 0101 1101
- A husky t-shirt was indicated by the text 'UWB' for codes 0110 and 1110

Analysis

The HEX display practice for part 1 and electronic detector device performed according to expectations by outputting the proper displays established by the corresponding truth tables. It can be safely assumed that the circuits were designed accurately and effectively according to requirements. The following paragraphs will address the reasons for why the correct results were not outputted initially and why these results were received.

The first experiment that designed a multi-level logic driver code on Quartus to drive the seven segment display on the DE1-SoC board did not operate correctly initially for two reasons:

1. The code was incorrectly instantiating the bit size for each input and output.
2. The pins were not lighting up correctly because of missed pin assignment and the board appeared to be missing powered LED strips.

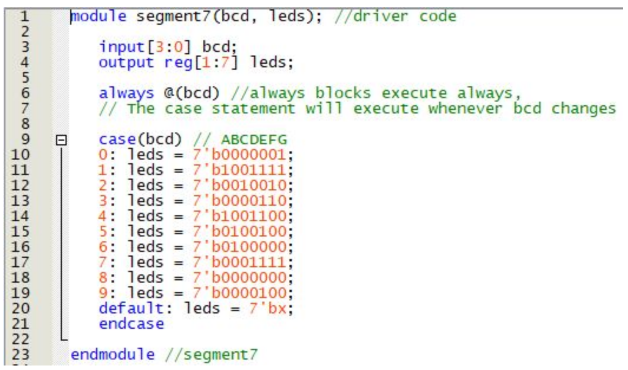
The second experiment that designed the UPC electronic device did not operate correctly initially for four main reasons.

1. The driver code was incorrectly setup.
2. The code was incorrectly instantiated.
3. The 7-bit hex display was incorrectly displayed.
4. The pins were not assigned correctly.

Conclusion

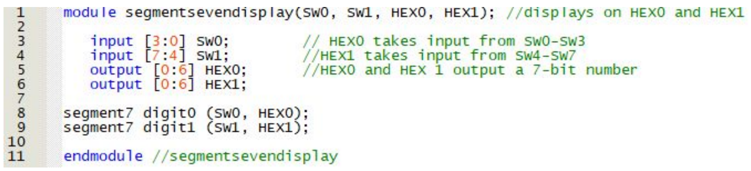
The use of computer aided simulations in Quartus and Verilog programming allows for increased efficiency in the steps and processes needed to develop and verify digital logic circuits on the FPGA. Debugging and determining the major sources for issues is a long process for all simulations and Verilog programming.

Appendix



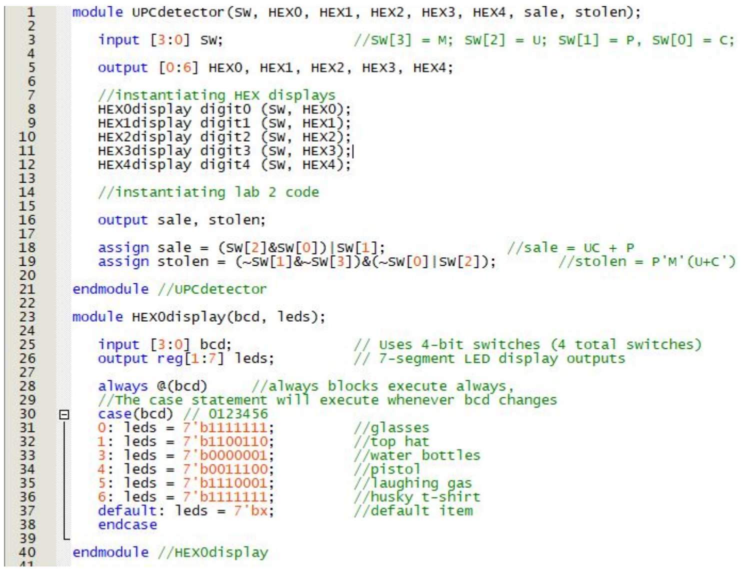
```verilog
module segment7(bcd, leds); //driver code

   input[3:0] bcd;
   output reg[1:7] leds;

   always @(bcd) //always blocks execute always,
   // The case statement will execute whenever bcd changes

   case(bcd) // ABCDEFG
   0: leds = 7'b0000001;
   1: leds = 7'b1001111;
   2: leds = 7'b0010010;
   3: leds = 7'b0000110;
   4: leds = 7'b1001100;
   5: leds = 7'b0100100;
   6: leds = 7'b0100000;
   7: leds = 7'b0001111;
   8: leds = 7'b0000000;
   9: leds = 7'b0000100;
   default: leds = 7'bx;
   endcase

endmodule //segment7
```

Example of Code 1: Driver Code for "Segment7"



```verilog
module segmentsevendisplay(SW0, SW1, HEX0, HEX1); //displays on HEX0 and HEX1

   input [3:0] SW0;         // HEX0 takes input from SW0-SW3
   input [7:4] SW1;         //HEX1 takes input from SW4-SW7
   output [0:6] HEX0;       //HEX0 and HEX 1 output a 7-bit number
   output [0:6] HEX1;

segment7 digit0 (SW0, HEX0);
segment7 digit1 (SW1, HEX1);

endmodule //segmentsevendisplay
```

Example of Code 2: HEX0 and HEX1 Outputs



```verilog
module UPCdetector(SW, HEX0, HEX1, HEX2, HEX3, HEX4, sale, stolen);

   input [3:0] SW;                 //SW[3] = M; SW[2] = U; SW[1] = P, SW[0] = C;

   output [0:6] HEX0, HEX1, HEX2, HEX3, HEX4;

   //instantiating HEX displays
   HEX0display digit0 (SW, HEX0);
   HEX1display digit1 (SW, HEX1);
   HEX2display digit2 (SW, HEX2);
   HEX3display digit3 (SW, HEX3);
   HEX4display digit4 (SW, HEX4);

   //instantiating lab 2 code

   output sale, stolen;

   assign sale = (SW[2]&SW[0])|SW[1];                //sale = UC + P
   assign stolen = (~SW[1]&~SW[3])&(~SW[0]|SW[2]);          //stolen = P'M'(U+C')

endmodule //UPCdetector

module HEX0display(bcd, leds);

   input [3:0] bcd;                // Uses 4-bit switches (4 total switches)
   output reg[1:7] leds;           // 7-segment LED display outputs

   always @(bcd)      //always blocks execute always,
   //The case statement will execute whenever bcd changes
   case(bcd) // 0123456
   0: leds = 7'b1111111;           //glasses
   1: leds = 7'b1100110;           //top hat
   3: leds = 7'b0000001;           //water bottles
   4: leds = 7'b0011100;           //pistol
   5: leds = 7'b1110001;           //laughing gas
   6: leds = 7'b1111111;           //husky t-shirt
   default: leds = 7'bx;           //default item
   endcase

endmodule //HEX0display
```

Example of Code 3: Electronic Detector Device