EGR 400
Advanced Digital
System Design Using FPGAs

Lab 1: Gate-Level Binary Decoder

Prepared for: Dr. Foist

Christopher Parisi

College of Engineering
California Baptist University

09/28/12

Introduction

The objective for this lab was to design, simulate, synthesize, and verify a 2-to-4 binary decoder as well as a 3-to-8 binary decoder. The instructions were found in section 2.9.2 of the *FPGA Prototyping by Verilog Examples* textbook by Pong P. Chu. A truth table was given and we were instructed to derive the logic expressions, write the HDL code, derive a testbench, then verify our design on the Spartan 3E-Starter board. Once the 2-to-4 decoder was verified, we were instructed to repeat the same steps but instead use our 2-to-4 decoder to create a 3-to-8 decoder.

Procedure

Step 1
"Determine the logic expressions for the 2-to-4 decoder with enable and derive the HDL code using only logical operators."

Based on the truth table shown below, we were able to derive the basic logic expressions.

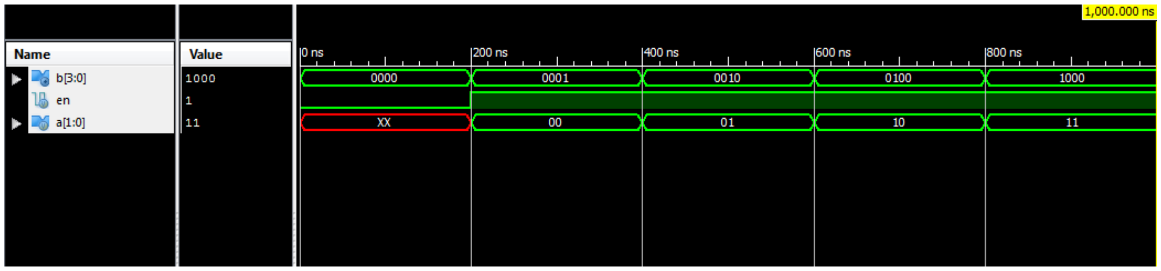
en	***a(1)***	***a(2)***	***bcode***
0	-	-	0000
1	0	0	0001
1	0	1	0010
1	1	0	0100
1	1	1	1000

Logic Expressions:
b3 = en & a1 & a0
b2 = en & a1 & !a0
b1 = en & !a1 & a0
b0 = en & !a1 & !a0

With the logic expressions we derived, we were able to easily write the HDL code using only logical operators. Our code for the 2-to-4 decoder can be found in the Appendix (Pg. 4).

Step 2
"Derive a testbench for the decoder. Perform a simulation and verify the correctness of the design."

We took our Verilog code and derived a testbench to simulate our design. This testbench utilized every possible input in order to confirm correct functionality. The VHDL for the testbench can be found in the Appendix (Pg. 4-5). The resulting waveform confirming our design is shown on the next page. A resource summary for our design can be found in the Appendix (Pg. 7).



Testbench simulation for 2-to-4 decoder

Step 3
"Use two switches as the inputs and four LEDs as the outputs. Synthesize the circuit and download the configuration file to the prototyping board. Verify its operation.

Once we had confirmed our design worked with the testbench, we then specified which inputs and outputs we would use on the prototyping board. We used the User Guide for the Spartan 3E-Starter board to determine the switch and LED names used for User Constraint file. The code we wrote for this file can be found in the Appendix (Pg. 5). We then downloaded our design to the board and tested the circuit. The design was functional and acted exactly as the simulation predicted.

Step 4
"Use the 2-to-4 decoders to derive a 3-to-8 decoder. First draw a block diagram and then derive the structural HDL code according to the diagram."

The steps for creating a 3-to-8 decoder was the same as the 2-to-4 decoder. We first drew the block diagram and then found the logic expressions from that and the truth table. We then wrote the Verilog code implementing our logic which utilized two 2-to-4 decoders. Our block diagram and code for the 3-to-8 decoder can be found in the Appendix (Pg. 6).

Step 5
"Derive a testbench for the 3-to-8 decoder. Perform a simulation and verify the correctness of the design."

We again used our Verilog code and derived a testbench that simulated the circuit. We ran the testbench and confirmed our designs correctness. Our Verilog for the testbench can be found in the Appendix (Pg. 7-8).

Step 6
"Use three switches as the inputs and eight LEDs as the outputs. Synthesize the circuit and download the configuration file to the prototyping board. Verify its operation."

Lastly, we specified the switches and LEDs then downloaded our design to the Spartan 3E-Starter board. The design worked and the LEDs lit up according to the switched used. We showed the professor our working circuit and finished the lab.

Conclusion

In summary, we used our knowledge of Digital Logic and Verilog to design and implement two different sizes of binary decoders. We designed a 2-to-4 decoder module and then utilized two of these modules to create a 3-to-8 decoder. We learned more about Verilog and its syntax, as well as how to use the Xilinx ISE software.

We enjoyed the mixture of logic and hands on experience taught in this lab. We were able to review what we learned in Digital Logic and apply it to what we are currently leaning in FPGA design. Our favorite part of the lab was seeing the result of our design on the physical board. There was nothing about this lab assignment that needs to be improved.

Appendix

2-to-4 Decoder Verilog Code

```
`timescale 1ns / 1ps

module two_to_four_decoder

//IO Ports
(
		input [1:0] a,
		input en,
		output [3:0] b
);

//Body
assign b[3] = en & a[1] & a[0];
assign b[2] = en & a[1] & ~a[0];
assign b[1] = en & ~a[1] & a[0];
assign b[0] = en & ~a[1] & ~a[0];

endmodule
```

2-to-4 Decoder Testbench Verilog Code

```
`timescale 1ns / 10ps

module test_b;

		// Inputs
		reg en;
		reg [1:0]a;

		// Outputs
		wire [3:0] b;

		// Instantiate the Unit Under Test (UUT)
		two_to_four_decoder uut
				(.en(en), .a(a), .b(b));

		initial begin
				//Test Vector 1
				en = 1'b0;

				a[1] = 1'bx;
				a[0] = 1'bx;
				#200;

				//Test Vector 2
				en = 1'b1;
				a[1] = 1'b0;
				a[0] = 1'b0;
				#200;
```

```
            //Test Vector 3
            en = 1'b1;
            a[1] = 1'b0;
            a[0] = 1'b1;
            #200;

            //Test Vector 4
            en = 1'b1;
            a[1] = 1'b1;
            a[0] = 1'b0;
            #200;

            //Test Vector 5
            en = 1'b1;
            a[1] = 1'b1;
            a[0] = 1'b1;
            #200;

      end
endmodule
```

__User Constraints File for 2-to-4 Decoder__

```
#==========================================================
# buttons & switches
#==========================================================

# 3 slide switches
NET "a<0>" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP;  # SW0
NET "a<1>" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP;  # SW1
NET "en"   LOC = "H18"| IOSTANDARD = LVTTL | PULLUP;  # SW2


#==========================================================
# 8 discrete LEDs
#==========================================================
NET "b<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; #LD3
NET "b<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; #LD2
NET "b<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; #LD1
NET "b<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; #LD0
```

<u>3-to-8 Decoder Block Diagram</u>

<u>3-to-8 Decoder Verilog Code</u>

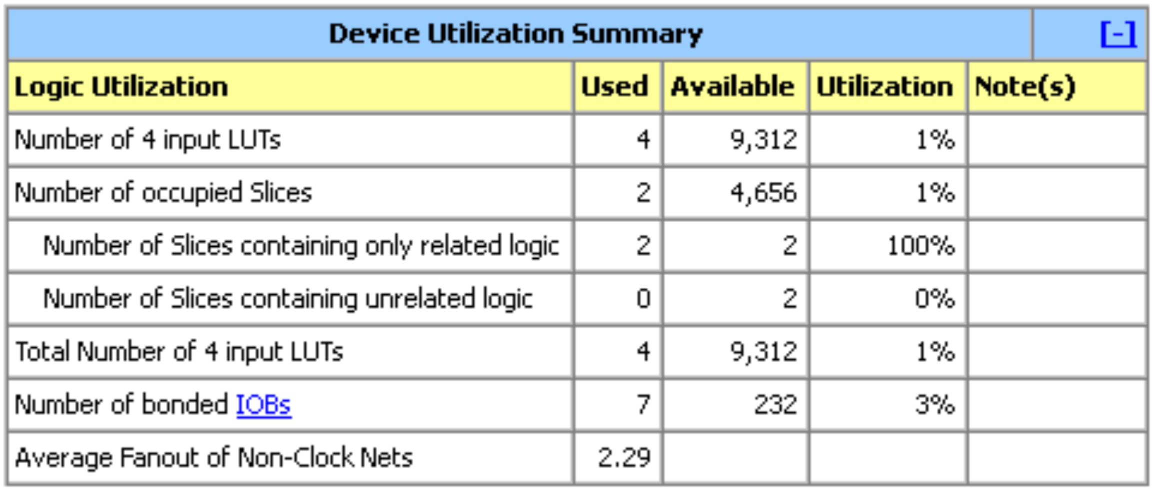
```
`timescale 1ns / 1ps
module three_to_eight
(
	input [2:0]a,
	input eq,
	output [7:0] b
 );

wire e0, e1

two_to_four_decoder dec1(.a[1](a[2]), .a[0](a[1]), .eq(e0));
two_to_four_decoder dec2(.a[1](a[2]), .a[0](a[1]), .eq(e1));

assign e1 = ~a0 & en;
assign e0 = a0 & en;

endmodule
```

3-to-8 Decoder Testbench Verilog Code

```
`timescale 1ns / 1ps
module testbench;

	// Inputs
	reg [2:0] a;
	reg en;

	// Outputs
	wire [7:0] b;

	// Instantiate the Unit Under Test (UUT)
	three_to_eight_decoder uut (
		.a(a), .en(en), .b(b)
	);

	initial begin
		// Initialize Inputs
		a = 0;
		en = 0;

		// Wait 100 ns for global reset to finish
		#100;

		//Test Vector 1
		en = 1'b0;
		a[2] = 1'bx;
		a[1] = 1'bx;
		a[0] = 1'bx;
		#200;

		//Test Vector 2
		en = 1'b1;
		a[2] = 1'b0;
		a[1] = 1'b0;
		a[0] = 1'b0;
		#200;

		//Test Vector 3
		en = 1'b1;
		a[2] = 1'b0;
		a[1] = 1'b0;
		a[0] = 1'b1;
		#200;

		//Test Vector 4
		en = 1'b1;
		a[2] = 1'b0;
		a[1] = 1'b1;
		a[0] = 1'b0;
		#200;

		//Test Vector 5
```

```
                en = 1'b1;
                a[2] = 1'b0;
                a[1] = 1'b1;
                a[0] = 1'b1;
                #200;

                //Test Vector 6
                en = 1'b1;
                a[2] = 1'b1;
                a[1] = 1'b0;
                a[0] = 1'b0;
                #200;

                //Test Vector 7
                en = 1'b1;
                a[2] = 1'b1;
                a[1] = 1'b0;
                a[0] = 1'b1;
                #200;

                //Test Vector 8
                en = 1'b1;
                a[2] = 1'b1;
                a[1] = 1'b1;
                a[0] = 1'b0;
                #200;

                //Test Vector 9
                en = 1'b1;
                a[2] = 1'b1;
                a[1] = 1'b1;
                a[0] = 1'b1;
                #200;
        end
endmodule
```

Resource Summary for 2-to-4 Decoder



Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	4	9,312	1%	
Number of occupied Slices	2	4,656	1%	
Number of Slices containing only related logic	2	2	100%	
Number of Slices containing unrelated logic	0	2	0%	
Total Number of 4 input LUTs	4	9,312	1%	
Number of bonded IOBs	7	232	3%	
Average Fanout of Non-Clock Nets	2.29			