CPE 166/ EEE 270: Advanced Logic Design

Spartan3E Kit and ISE Tool Introduction

Lab Section: 02

Student Name: Daniel Komac

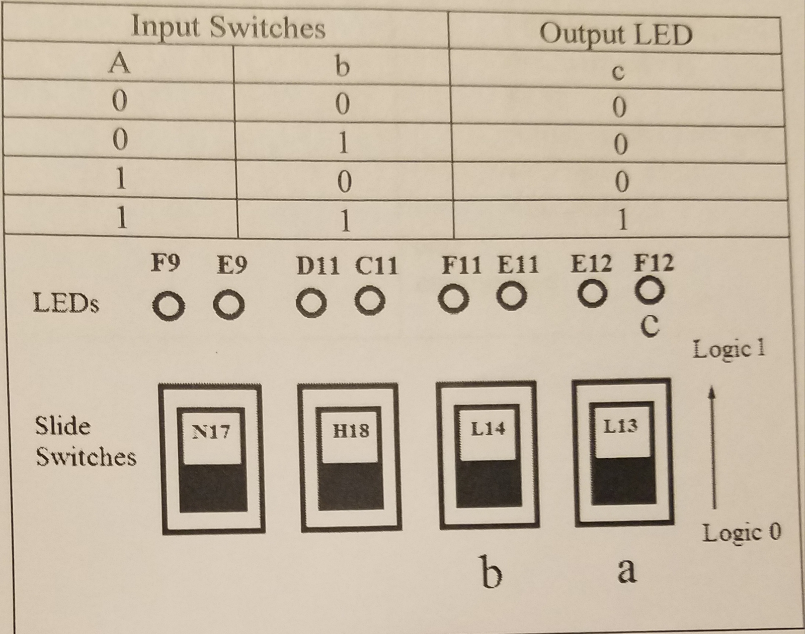
**Table of Contents:**

1. Lab 1 Part 1
2. Lab 1 Part 2
3. Lab 1 Part 3
4. Conclusion

**Introduction:** In this Lab we will be using the NEXYS4 DDR board to test and implement Verilog code written in Vivado and Notepad++. In this lab we will familiarize ourselves with these tools.

**Part 1:**

1. Design Purpose: To implement a AND gate onto our FPGA device (NEXYS4 DDR) and display the results using onboard LEDs.
2. Engineering Data:



1. Source Code:
   1. --Author: Daniel Komac

--date: 2/4/2018

--File Name: Test1.v

--Purpose: Programming AND gate in verilog.

--Project Part Number: Part 1

--Hardware FPGA/CPLD Device: xc7a100tcsg324-1

module test(a,b, c);

input a,b;

output c;

and g1(c, a, b);

endmodule

1. Constraints File:

##Switches

set\_property –dict{PACKAGE\_PIN M13 IOSTANDARD LVCMOS33}[get\_ports{a}]; #IO\_L6N\_T0\_D08\_VREF\_14 Sch=sw[2]

set\_property -dict { PACKAGE\_PIN R15   IOSTANDARD LVCMOS33 } [get\_ports { b }]; #IO\_L13N\_T2\_MRCC\_14 Sch=sw[3]

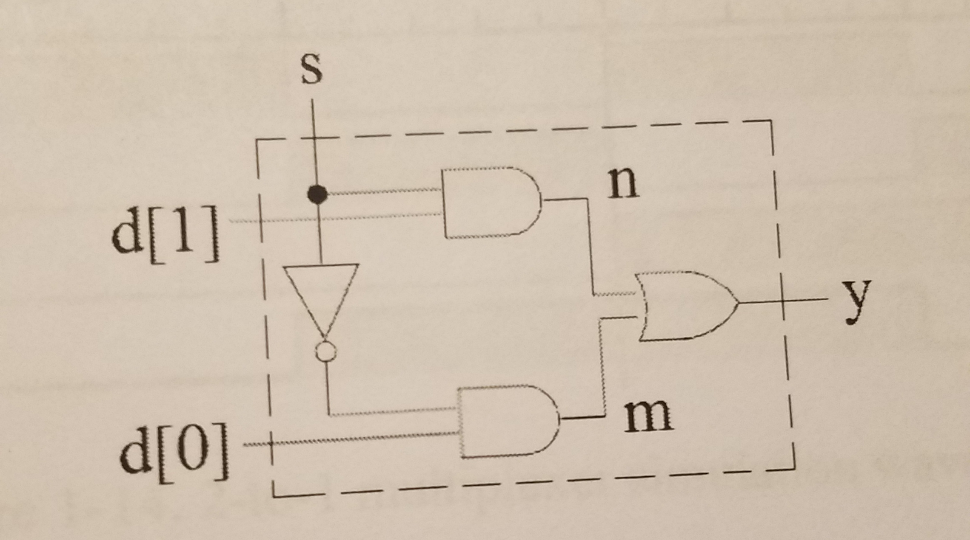
#LEDS

set\_property –dict {PACKAGE\_PIN N14 IOSTANDARD LVCMOS33} [get\_ports{c}]; #IO\_L6N\_T0\_D08\_VREF\_14 Sch=sw[2]

1. There was no simulation to be ran for the first part.
2. Results:

The results of this lab that the associated LED M13 on the NEXYS4 DDR board lit up when the associated truth table was displayed. This mean that both input pins we used for the and gate are flipped into their “on” state, outputting a 1 within the board and if both switches are on then the light flicks on

**Part 2:**

1. Design Purpose: The designing of a 2-to-1 multiplexer with a simulation testbench
2. Engineering Data: 
   1. Source Code:

--Author: Daniel Komac

--date: 2/4/2018

--File Name: mymux2to1.v

--Purpose: Source code for 2-1 multiplexer

--Project Part Number: Part 3. A

--Hardware FPGA/CPLD Device: xc7a100tcsg324-1

module mux2to1(d,s,y);

input [1:0] d;

output y;

wire m, n;

assign m=(~s) & d[0];

assign n= s & d[1];

Assign y = m | n;

Endmodule

--Author: Daniel Komac

--date: 2/4/2018

--File Name: mymux\_tb.v

--Purpose: Test bench for 2-1 multiplexer

--Project Part Number: Part 3. A

--Hardware FPGA/CPLD Device: xc7a100tcsg324-1

module mymux\_tb;

reg [1:0] d; reg s;

wire y;

test1 u1(.d(d), .s(s), .y(y));

initial

begin

   d=2'b10; s=0;

   #10;

   d=2'b01; s=0;

   #10;

   d=2'b10; s=1;

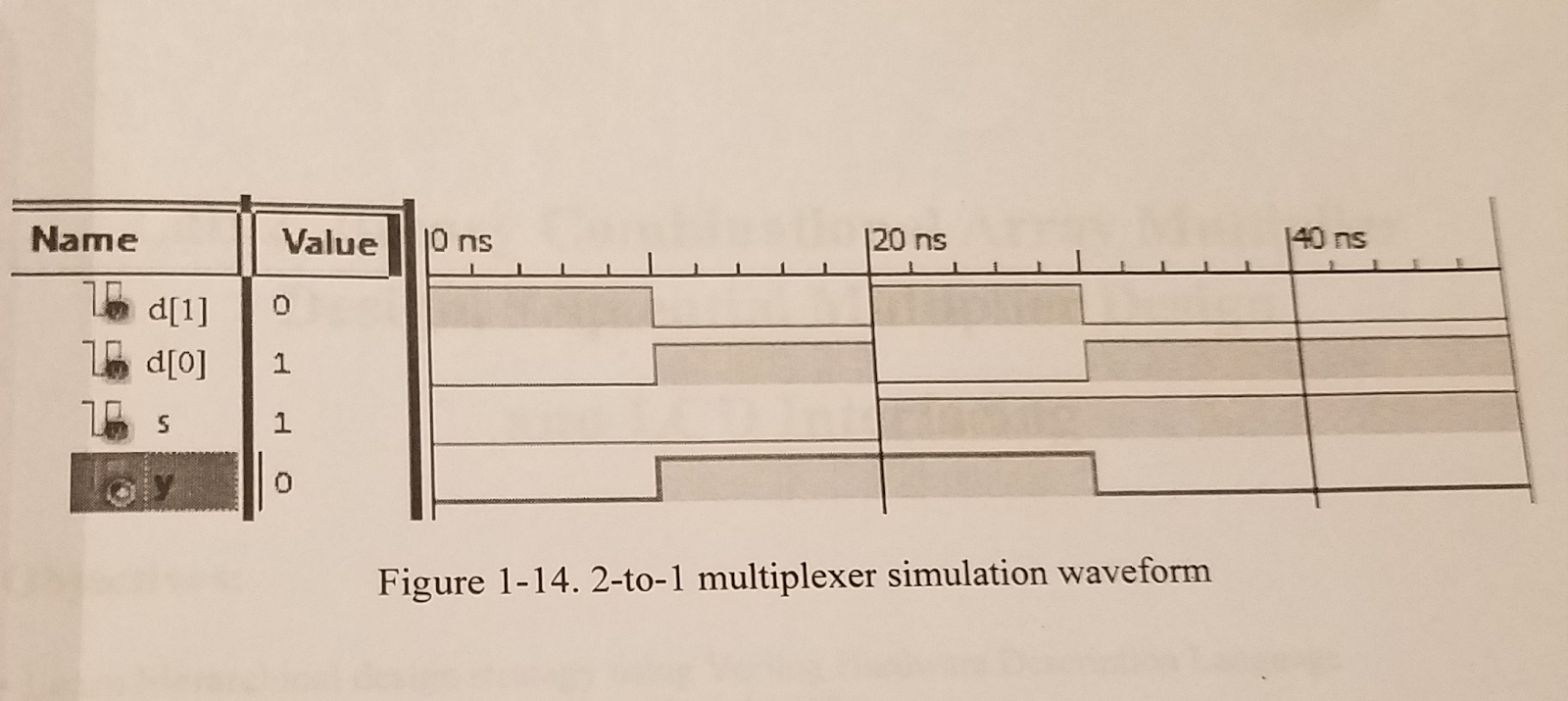
   #10;

   d=2'b01; s=1;

   #20 $stop;

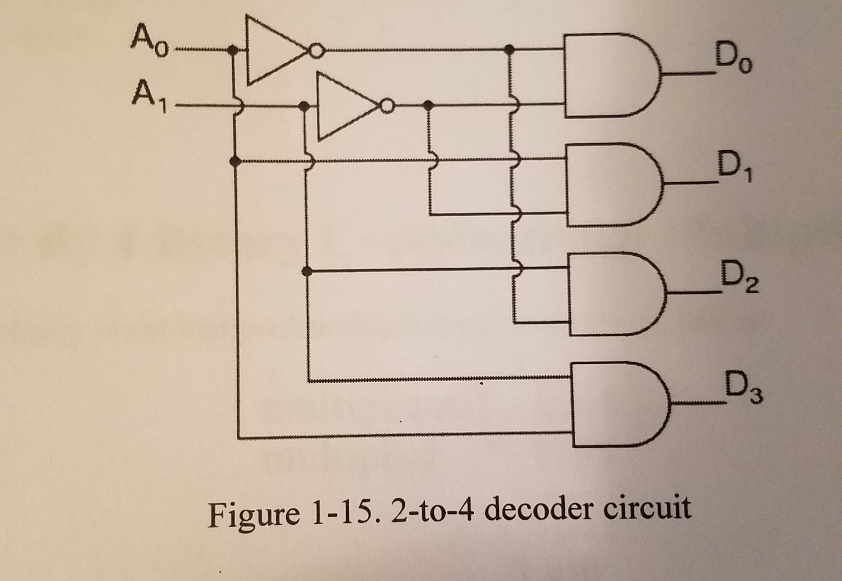
end

endmodule

1. Constraints File: The Constraints file was not needed for this section of the lab since we are running simulations of our code to test several different inputs and measure/read the outputs of our design.
2. Simulation: When we ran these simulations we obtained waveforms of the values of our d/s inputs and the outputs of our y. The only problem is that we did not snip a shot of these waveforms but through a confirmation that we did the demo the waveforms are identical to those within the lab manual
3. Results: The results of this lab was a 2-to-1 multiplexer used to obtain desired inputs according to our s value, the s value selects a channel for information to come in through. Our Y value only outputs according to s, if S is 1 it is Y=d[1], else Y=d[0].

**Part 3:**

1. Design Purpose: Build a 2-to-4 Decoder Design to be implemented on the board to understand the design of an encoder.
2. Engineering Data:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A\_1 | A\_0 | D\_0 | D\_1 | D\_2 | D\_3 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

1. Source Code:

--Author: Daniel Komac

--date: 2/4/2018

--File Name: part3.v

--Purpose: 2-4 Decoder circuit in Verilog source file.

--Project Part Number: Part 1

--Hardware FPGA/CPLD Device: xc7a100tcsg324-1

module test(a,b, l1,l2,l3,l4);

input a,b;

output l1,l2,l3,l4;

assign l1 =(~a)& (~b);

assign l2 = (~a) & b;

assign l3 = a & (~b);

assign l4 = a & b;

endmodule

1. Constraints File:

#Switches

set\_property -dict { PACKAGE\_PIN M13   IOSTANDARD LVCMOS33 } [get\_ports { a }]; #IO\_L6N\_T0\_D08\_VREF\_14 Sch=sw[2]

set\_property -dict { PACKAGE\_PIN R15   IOSTANDARD LVCMOS33 } [get\_ports { b }]; #IO\_L13N\_T2\_MRCC\_14 Sch=sw[3]

## LEDs

set\_property -dict { PACKAGE\_PIN N14   IOSTANDARD LVCMOS33 } [get\_ports { l1 }]; #IO\_L8P\_T1\_D11\_14 Sch=led[3]

set\_property -dict { PACKAGE\_PIN R18   IOSTANDARD LVCMOS33 } [get\_ports { l2 }]; #IO\_L7P\_T1\_D09\_14 Sch=led[4]

set\_property -dict { PACKAGE\_PIN V17   IOSTANDA

D LVCMOS33 } [get\_ports { l3 }]; #IO\_L18N\_T2\_A11\_D27\_14 Sch=led[5]

set\_property -dict { PACKAGE\_PIN U17   IOSTANDARD LVCMOS33 } [get\_ports { l4 }]; #IO\_L17P\_T2\_A14\_D30\_14 Sch=led[6]

1. Simulation: There was no simulation ran for this program, we only tested on the board
2. Results: With proper coding we were able to hook up two of the switches with the NEXYS4 and using the corresponding truth table above we mapped out these switches with a 2-to-4 layout, every input having a different LED that responds to the correct matching of switches.

**Conclusion:**

This lab gave a go introductory step into learning how to program our new NEXYS4 DDR boards with the Vivado application along with using Notepad++ to right code that can be used by Vivado. Vivado is the building block of these labs in our current state because it is what allows us to implement our hardware designs by allocating resources on the board to our design. These boards are also reprogrammable allowing us to experiment and do many different examples with one board. This lab was a general run down of introductory code, and implementation of the NEXYS4 DDR and Verilog code.