**CMPEN 371: Advanced Digital Design**

**Fall 2017**

**Basic Combination Design Lab**

**Due: October 4th, 2017**

**Team Serial Code: 2B**

**Siti Nadira, Nor Badrul Aman, sfn5077**

**Morayo Ogunsina, mao5270**

**ACKNOWLEDGEMENT**

This work is entirely our own and I did not provide any assistance except as noted. The approximate contribution of each team member is as follows:

50% Morayo Ogunsina

50% Siti Nadira, Nor Badrul Aman

**Grading Fabric**

|  |  |
| --- | --- |
| **Criteria** | **Grade** |
| Design (Sound algorithm, good block diagrams / state diagrams / state tables, efficient design, etc.)  Excellent: 6; Good: 4; Satisfactory: 2; Unsatisfactory: 0; Failure: -2 or worse | / 10 |
| Translate to VHDL (VHDL matches design docs, good comments / whitespace, follows guidelines, efficient coding, etc.)  Excellent: 6; Good: 4; Satisfactory: 2; Unsatisfactory: 0; Failure: -2 or worse | / 10 |
| Test in Simulator (test bench or script for each component and FSM, good coverage, components / FSMs verified, etc.)  Excellent: 6; Good: 4; Satisfactory: 2; Unsatisfactory: 0; Failure: -2 or worse | / 15 |
| Test in Hardware (demo in lab)   * Describe connection between your VHDL and the block diagrams * Describe your test benches * Demonstrate simulation of top level, how to drill down to lowest levels for debugging * Run Tcl test script (provided at demo time) to fully verify top level | / 15 |
| Bonus (optional challenge, etc.) |  |
| Penalty (late demo, late submission on CANVAS, team member absent from demo, etc.)  Demo late in lab: -2; Demo up to 5 days late: -5; Demo more than 5 days late: -10  Submitted late or incompletely on CANVAS: -5; Not submitted on CANVAS: no grade for lab |  |
| TOTAL | / 50 |

**DESIGN**

The following design documents are attached:

Filename Description

Lab3\_BlockDiagram Top level block diagram

Hex\_Display.png Hex\_display block diagram.

FullAdder.png FullAdder\_4bit block diagram

shematic Schematic Diagram after verifying the design

**VHDL Design or TRANSLATE TO VHDL**

The following HDL models are attached:

Filename Description

Lab3 Top Level: Displays sum of two elements on a seven seg display with controls

FullAdder\_4bit Component 1: computes the sum of two elements using a full adder

hex\_display Component 2: displays digits on the boards seven segs including the decimal point

**TEST IN SIMULATOR**

The following files were used for testing to verify the design:

Filename Description

Lab3\_tb Simulation File for Lab\_3

All tests in the testbench passed and on the board also. Exhaustive testing was not carried out, rather input values of A from 0 to 15 and B from 9 to 12 were used to test the design in the testbench.

The lab was to make a design that displays the output of a 4 bit full adder on a seven segment display of the nexys board. The top level design, lab 3, consist of two components: FullAdder\_4bit (from previous lab) and the hex\_display. The hex display has four inputs: decimalpoint, value, blank and test. The decimal point switch enabled the decimal point to be displayed on the seven segment, while the blank clears out the display on all the segments. On the other hand, the test forces all the channels to display digit ‘8’.

The sum from the adder is wired to the value inputs of the hex display and is passed through a encoder to display the 1-15 values of the input ‘value’.

The Full adder was designed from the truth table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **Cin** | **Cout** | **EEEE** |
| 0000 | 0000 | 0 | 0 | 0000 |
| 0001 | 0001 | 0 | 0 | 0010 |
| 0010 | 0010 | 0 | 0 | 0100 |
| 0011 | 0011 | 0 | 0 | 0110 |
| 0100 | 0100 | 0 | 0 | 1000 |
| 0101 | 0101 | 0 | 0 | 1010 |
| 0110 | 0110 | 0 | 0 | 1100 |
| 0111 | 0111 | 0 | 0 | 1110 |
| 1000 | 1000 | 0 | 1 | 0000 |
| 1001 | 1001 | 0 | 1 | 0010 |
| 1010 | 1010 | 0 | 1 | 0100 |
| 1011 | 1011 | 0 | 1 | 0110 |
| 1100 | 1100 | 0 | 1 | 1000 |
| 1101 | 1101 | 0 | 1 | 1010 |
| 1110 | 1110 | 0 | 1 | 1100 |
| 1111 | 1111 | 0 | 1 | 1110 |

Note: not all test cases are outlined in this truth table

Cout = A and B OR A and Cin OR B and Cin

EEEE = A xor B xor Cin

The encoder uses a case-when statement to choose how ‘value’ is encoded on the seven segment display.

case value is

-- case counting is

when "0000" => segs <=NOT "0111111"; -- 0

when "0001" => segs <=NOT "0000110"; -- 1

when "0010" => segs <=NOT "1011011"; -- 2

when "0011" => segs <=NOT "1001111"; -- 3

when "0100" => segs <=NOT "1100110"; -- 4

when "0101" => segs <=NOT "1101101"; -- 5

when "0110" => segs <=NOT "1111101"; -- 6

when "0111" => segs <=NOT "0000111"; -- 7

when "1000" => segs <=NOT "1111111"; -- 8

when "1001" => segs <=NOT "1100111"; -- 9

when "1010" => segs <=NOT "1110111"; -- A

when "1011" => segs <=NOT "1111100"; -- b

when "1100" => segs <=NOT "0111001"; -- c

when "1101" => segs <=NOT "1011110"; -- d

when "1110" => segs <=NOT "1111001"; -- E

when others => segs <=NOT "1110001"; -- F

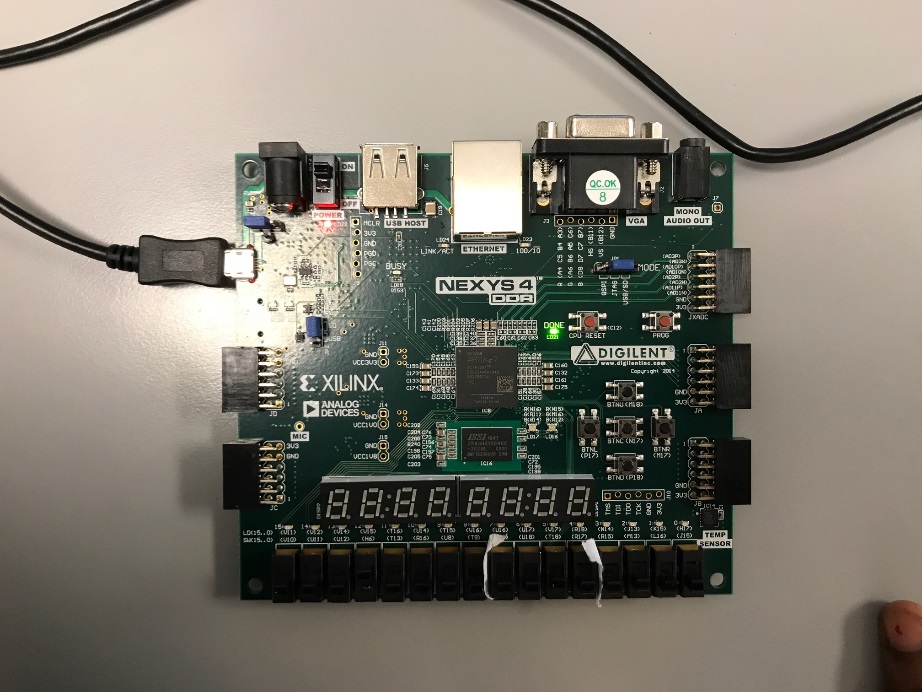
**TEST IN HARDWARE**

The project implements all the functionality specified in the lab. For demonstration of the project, visit

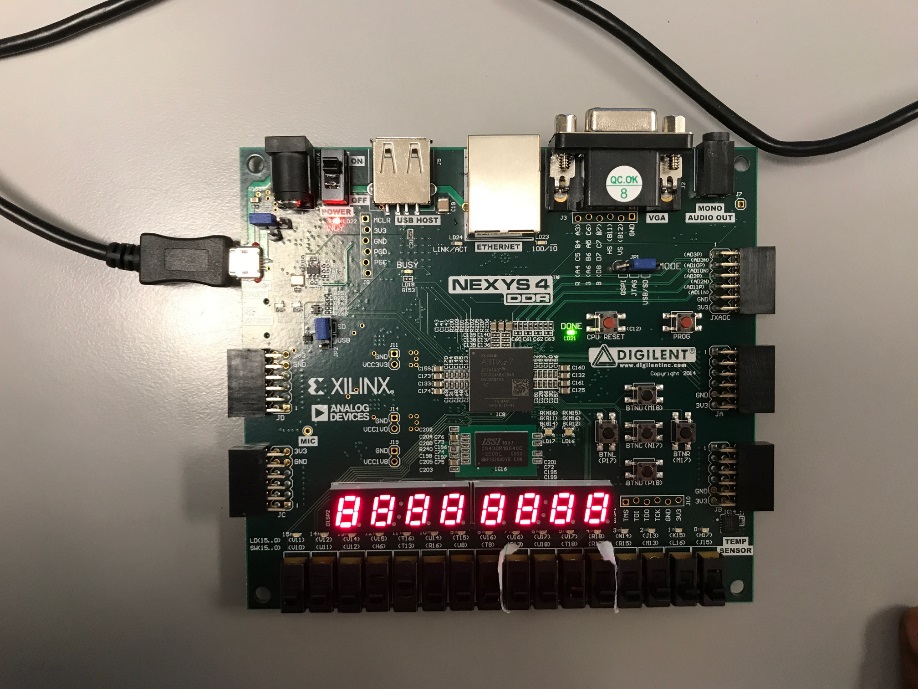
this url: <http://youtu.be/45cH-9nCD9A>

Attached are the photos from our demo

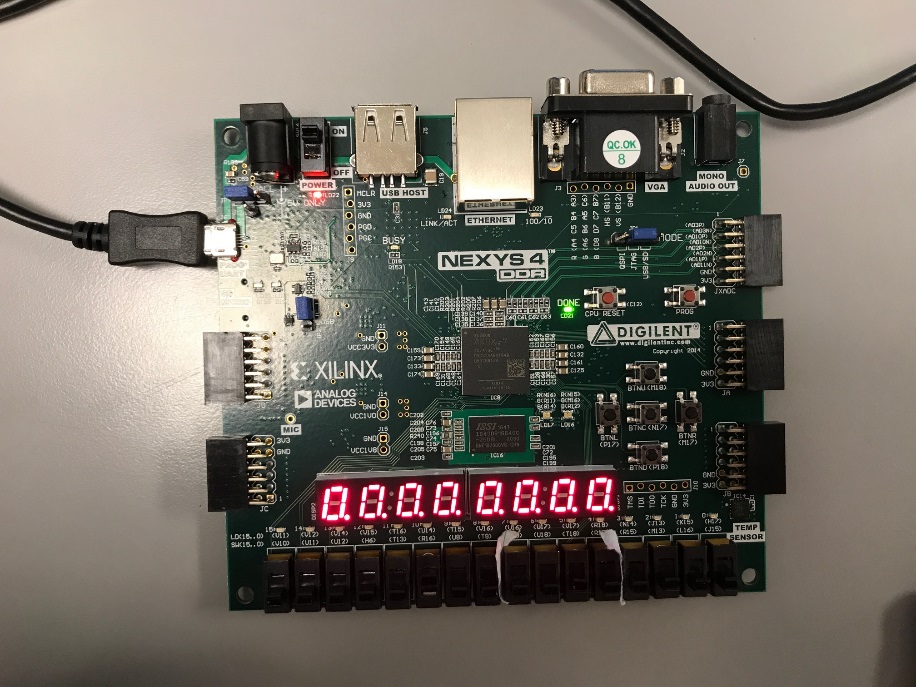
Blank:



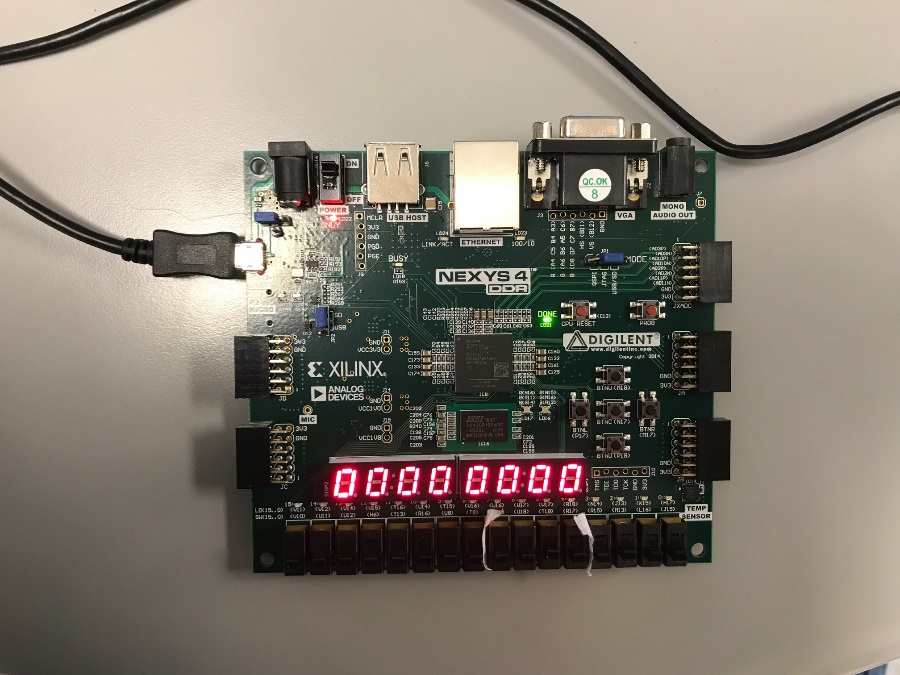
Test:



Decimal point:



When decimal point switch is off:



**QUESTIONS**

--- No extra questions this week.