Course Name: Hardware/Software Design of Embedded Systems Laboratory
Course Number and Section: 14:332:493:02

Experiment: Lab 1 - Introduction to FPGA's and VHDL
Lab Instructor: Kodra, Kliti

Date Performed: September 16th, 2016
Date Submitted: September 26th, 2016

Submitted by: Gregory Leonberg 159-00-5392

Purpose:

The purpose of this lab is to become familiar with the DE2-115 development board and the Quartus IDE. To accomplish this, a few small designs will be created, using the VHDL language, and then implemented on the board for testing.

The first design is a simple circuit that connects the output of switches 0 -17 to the input of the red LEDs, using two 18 bit vectors and a concurrent signal assignment.

The second design is a simple combinatorial logic circuit that implements the Boolean logic:
Z = !(!(X or !Y) and (!x and y))
Where Z is the output fed to LEDR(0), X is an input from SW(1) and Y is an input from SW(2).

The third design uses the four rightmost switches as a 4 bit binary input and then decodes the input to a hexadecimal value as a 7 bit vector, which is then displayed identically across all 8 seven segment displays on the board.

Theory(ies) of Operation:

For the first circuit, the lights should simply illuminate when the corresponding switch is set to the On position.

For the second design, the light should be illuminated for all X and Y input combinations with the exception of X = 0 and Y = 1.

For the third design, inputs 0000-1111 should display as 0-F in hexadecimal.

Truth Tables:

For Circuit 2:

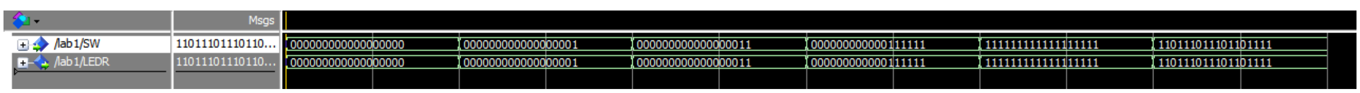
X [SW(1)]	Y [SW(2)]	Nor Gate	And Gate	Result
0	0	1	0	1
0	1	1	1	0
1	0	1	0	1
1	1	0	0	1

Circuit 1:

Design:

```
-- Import logic primitives
library ieee;
use ieee.std_logic_1164.all;

-- Simple module that connects the SW switches to the LEDR lights
entity lab1 is

port ( SW: in STD_LOGIC_VECTOR(17 downto 0); -- Initialize switches as an input
            LEDR: out STD_LOGIC_VECTOR(17 downto 0) ); -- Initialize red LEDs as an output

end lab1;


-- Define characteristics of the entity lab1
architecture Behavior of lab1 is
begin

        LEDR <= SW; -- Assign each switch to one red LED

end Behavior;
```

Test:

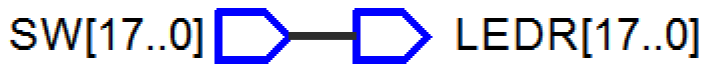


Implementation:

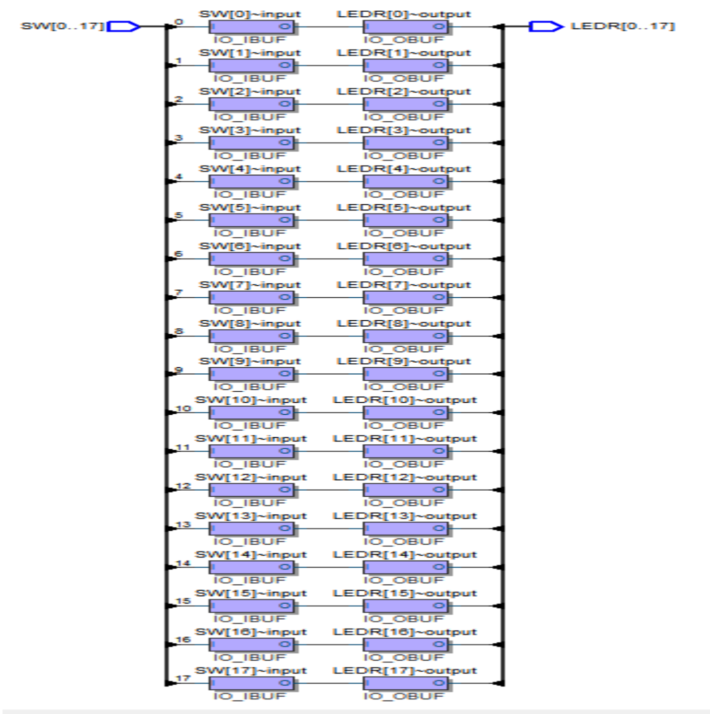
Flow Summary:

Flow Status	Successful - Sun Sep 25 15:03:45 2016
Quartus Prime Version	16.0.0 Build 211 04/27/2016 SJ Lite Edition
Revision Name	lab1
Top-level Entity Name	lab1
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	0 / 114,480 (0 %)
Total combinational functions	0 / 114,480 (0 %)
Dedicated logic registers	0 / 114,480 (0 %)
Total registers	0
Total pins	36 / 529 (7 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

RTL Viewer:



Technology Viewer:



Pin-out File Review:

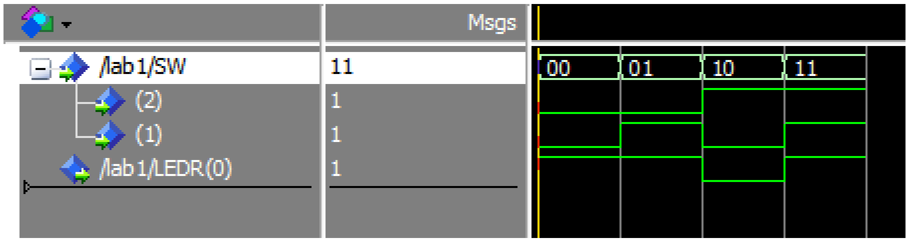
After checking the file’s contents, all ports of the design are assigned to pins.

Circuit 2:

Design:

```
-- Import logic primitives
library ieee;
use ieee.std_logic_1164.all;

-- Simple boolean logic ckt

-- Z = !( !(X or !Y) and (!x and y) )
-- where X and Y are inputs from SW(1) and SW(2)
-- and Z is an LEDR output light LEDR(0)

----------------
-- | X | Y | Z |
----------------
-- | 0 | 0 | 1 |
----------------
-- | 0 | 1 | 0 |
----------------
-- | 1 | 0 | 1 |
----------------
-- | 1 | 1 | 1 |
----------------

entity part1 is

        port (SW: in std_logic_vector(2 downto 1); -- Switches as input
              LEDR: out std_logic_vector(0 downto 0) ); -- red LEDs as output

end part1;


-- Implementation of boolean logic
architecture logic of part1 is
begin

        process (SW) is begin

                LEDR(0) <= not( (SW(2) and not SW(1)) and not(not SW(2) or SW(1) ) ); -- Logic as shown in figure 20

        end process;

end logic;
```

Test:

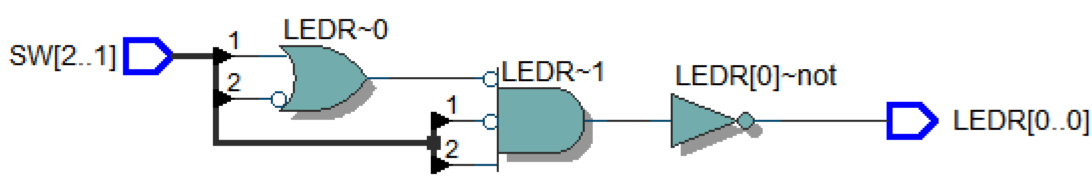


Implementation:

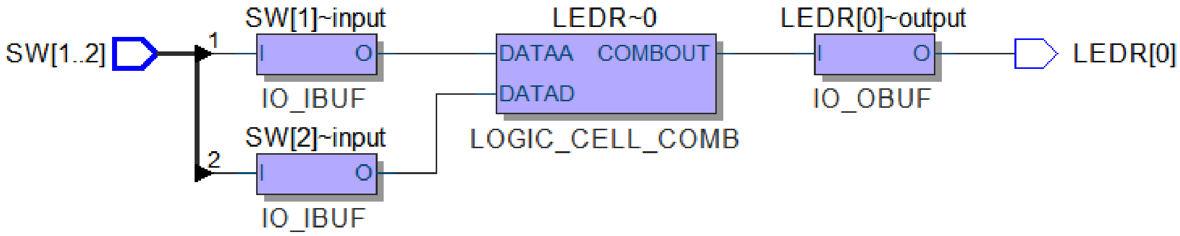
Flow Summary:

Flow Status	Successful - Sun Sep 25 15:40:12 2016
Quartus Prime Version	16.0.0 Build 211 04/27/2016 SJ Lite Edition
Revision Name	lab1
Top-level Entity Name	lab1
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	1 / 114,480 (< 1 %)
Total combinational functions	1 / 114,480 (< 1 %)
Dedicated logic registers	0 / 114,480 (0 %)
Total registers	0
Total pins	3 / 529 (< 1 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

RTL Viewer:



Technology Viewer:



Pin-out File Review:

After checking the file’s contents, all ports of the design are assigned to pins.

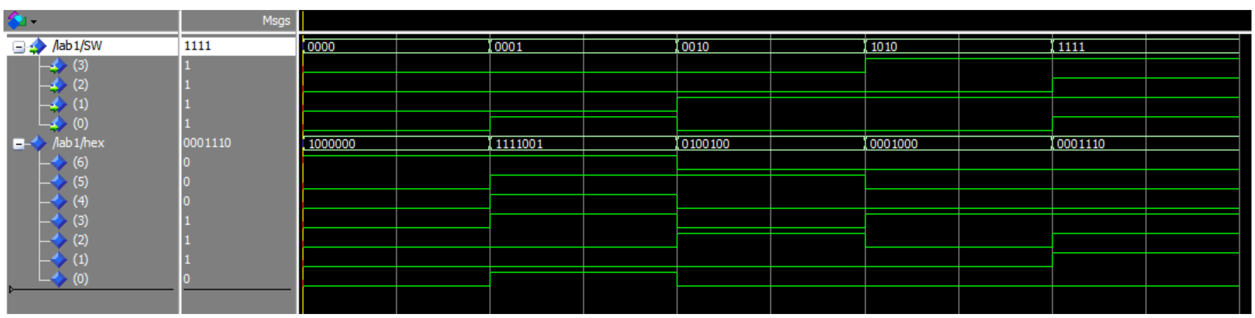
Circuit 3:

Design:

```
-- Import logic primitives
library ieee;
use ieee.std_logic_1164.all;

-- Decodes 4 bit input into hex, drives it across all hex displays on board
entity part2 is

port  ( SW: in std_logic_vector(3 downto 0); -- 4 switches as binary input to decode
        HEX0: out std_logic_vector(6 downto 0);
        HEX1: out std_logic_vector(6 downto 0);
        HEX2: out std_logic_vector(6 downto 0);
        HEX3: out std_logic_vector(6 downto 0);
        HEX4: out std_logic_vector(6 downto 0);
        HEX5: out std_logic_vector(6 downto 0);
        HEX6: out std_logic_vector(6 downto 0);
        HEX7: out std_logic_vector(6 downto 0) );

end part2;

architecture decode of part2 is
signal hex : std_logic_vector(6 downto 0); -- intermediate signal
begin

        process (SW, hex) is
        begin

                case SW is

                        when "0000" => hex <= "1000000"; -- 0
                        when "0001" => hex <= "1111001"; -- 1
                        when "0010" => hex <= "0100100"; -- 2
                        when "0011" => hex <= "0110000"; -- 3
                        when "0100" => hex <= "0011001"; -- 4
                        when "0101" => hex <= "0010010"; -- 5
                        when "0110" => hex <= "0000010"; -- 6
                        when "0111" => hex <= "1111000"; -- 7
                        when "1000" => hex <= "0000000"; -- 8
                        when "1001" => hex <= "0011000"; -- 9
                        when "1010" => hex <= "0001000"; -- A
```

```vhdl
                when "1011" => hex <= "0000011"; -- b
                when "1100" => hex <= "0100111"; -- c
                when "1101" => hex <= "0100001"; -- d
                when "1110" => hex <= "0000110"; -- E
                when "1111" => hex <= "0001110"; -- F
                when others => hex <= "1111111"; -- null

            end case;

            -- Drive signal to hex displays (Active low)

            HEX0 <= hex;
            HEX1 <= hex;
            HEX2 <= hex;
            HEX3 <= hex;
            HEX4 <= hex;
            HEX5 <= hex;
            HEX6 <= hex;
            HEX7 <= hex;

        end process;

end decode;
```

<u>Test:</u>

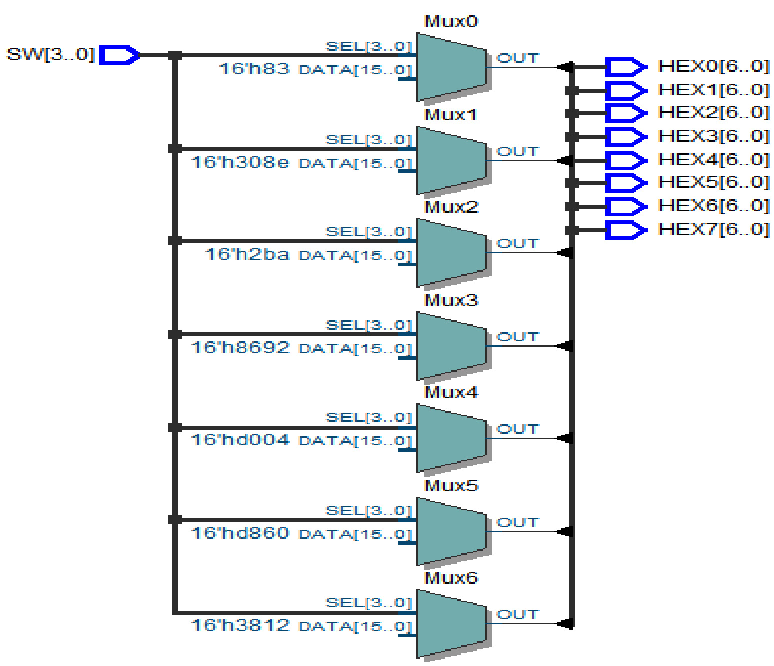


Implementation:

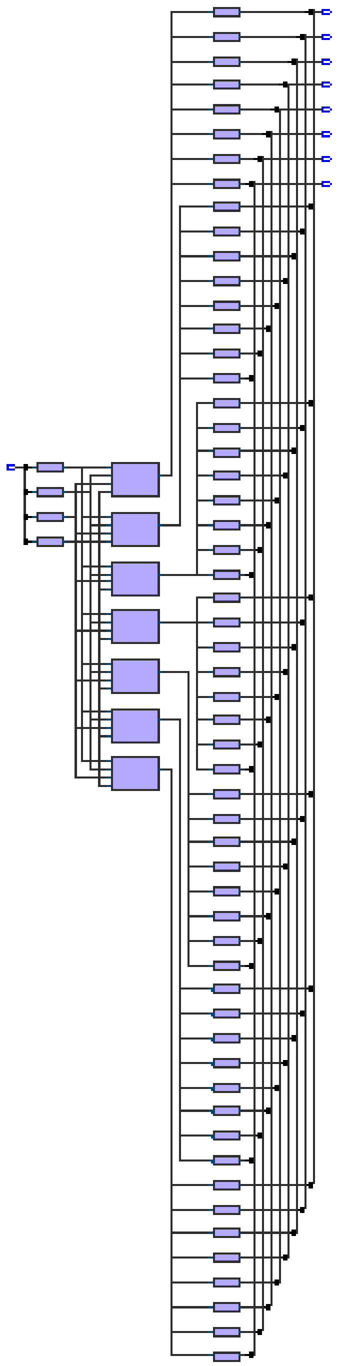
Flow Summary:

Flow Status	Successful - Sun Sep 25 15:59:08 2016
Quartus Prime Version	16.0.0 Build 211 04/27/2016 SJ Lite Edition
Revision Name	lab1
Top-level Entity Name	lab1
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	7 / 114,480 (< 1 %)
Total combinational functions	7 / 114,480 (< 1 %)
Dedicated logic registers	0 / 114,480 (0 %)
Total registers	0
Total pins	60 / 529 (11 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

RTL Viewer:



Technology Viewer:



Pin-out File Review:

After checking the file's contents, all ports of the design are assigned to pins.

Discussion:

Each of the parts of this lab uses an insignificant amount of resources on the FPGA, with the exception of IO pins (due to IO requiring physical space on the board). The first part used only pins, since it was simple mapping of the switches to LEDs with no combinatorial logic involved. The second part used a single logic element to implement a NAND gate, a NOR gate, and an AND gate. The third part used 7 logic elements because it needed a multiplexer for each hex output bit.

Since modern FPGAs have such a large amount of logic elements, larger (and even multiple) designs can be loaded onto the chip for testing. Since modern ASICs are growing larger and larger due to the shrinking size of transistors, having larger FPGAs allows designs to be prototyped and tested with hardware rather than pure simulation.

While performing this lab, I discovered from a fellow student that the seven segment display on the DE2-115 was active low, meaning I had to invert all of my output bits. At first, I tried using bitwise not gates to accomplish this, but the RTL viewer showed me the folly of this strategy: an extra 56 (7 * 8) not gates were added to the design, which I considered a wanton waste of hardware. Therefore, I went back and fixed the issue at the multiplexer level by changing the values in my case statement.

After performing this lab, I believe I understand all of the concepts covered.