date: 18th march, 2020

Lab 7

Lab Title: Seven Segment Display

Recorded and Submitted by: Bhesaniya Abhishu

Objective: To write VHDL and Vlog program for seven segment display. when ever the input in binary corresponding Hex value should be display in sevensegmend display.

Implementation:

- To turn on an LED in the display, you must put a '0' on the corresponding output, a '1' will turn off the segment

VHDL Procedure:

- 1st step is to make table for binary to hex conversion. like according to your binary input what should you feed to your seven segment display.
- then define inputs and output.
- use signal for both input and output.

```
Signal A : std_logic_vector (3 down to 0);
Signal HEX: std_logic_vector (6 down to 0);

A <= Binary (3 down to 0);
XEX <= "1111111";
```

we will use case and when statment for Binary to Hex conversion.

After writting we will compile the program and will do pin assignment and then we will test on FBGA Board.

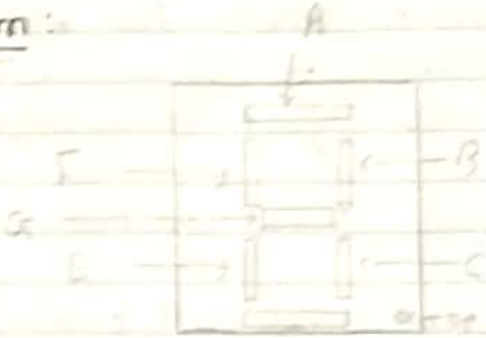
Vlog Procedure:

- 1st step is same as VHDL.
- we will define input and output.
- use wire and reg for input and output.

```
wire [3:0] A = Binary [3:0];
Reg [6:0] Hex = 7'b 1111111;
```

- else when statement for binary to Hex conversion.
- Then compile, do Pinassignment Again compile and download code on FPGA Board. and test the condition.

Diagram:



Conversion Table

Binary				HEX						
3	2	1	0	A	B	C	D	E	F	G
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0

0	1	0	0	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	0	0	0
1	0	1	1	1	1	0	0	0	0	0
1	1	0	0	0	1	1	0	0	0	1
1	1	0	1	1	0	0	0	0	0	0
1	1	1	0	0	1	1	0	0	0	0
1	1	1	1	0	1	1	1	0	0	0

Pin Assignment

Input / output Variable	Signal name	FBGA Pin no.
EN	SW[9]	PIN - AE12
Binary [0]	SW[0]	PIN - AB12
Binary [1]	SW[1]	PIN - AC12
Binary [2]	SW[2]	PIN - AF9
Binary [3]	SW[3]	PIN - AF10
LED [0]	LEDR[0]	PIN - V16
LED [1]	LEDR[1]	PIN - W16
LED [2]	LEDR[2]	PIN - V17
LED [3]	LEDR[3]	PIN - V18
HEX [0]	HEX0[0]	PIN - AE26
HEX [1]	HEX0[1]	PIN - AE27
HEX [2]	HEX0[2]	PIN - AE28
HEX [3]	HEX0[3]	PIN - AG27
HEX [4]	HEX0[4]	PIN - AF28
HEX [5]	HEX0[5]	PIN - AG28
HEX [6]	HEX0[6]	PIN - AH28
HEX1 [6]	HEX2[6]	PIN - AB23
HEX1 [5]	HEX2[5]	PIN - AE29
HEX1 [4]	HEX2[4]	PIN - AD29
HEX1 [3]	HEX2[3]	PIN - AC28

HEX1 [B]	HEX2 [B]	PIN - AD30
HEX1 [I]	HEX2 [I]	PIN - AC29
HEX1 [O]	HEX2 [O]	PIN - AC30
HEX2 [0]	HEX3 [0]	PIN - AB22
HEX2 [1]	HEX3 [1]	PIN - AB25
HEX2 [2]	HEX3 [2]	PIN - AB28
HEX2 [3]	HEX3 [3]	PIN - AC25
HEX2 [4]	HEX3 [4]	PIN - AD25
HEX2 [5]	HEX3 [5]	PIN - AC27
HEX2 [6]	HEX3 [6]	PIN - AD26

VHDL - Program

```
library ieee;
use ieee.std_logic_1164.all;

Entity abhesamiyulab7 is
Port ( EN : in std_logic;
Binary : in std_logic_vector (3 downto 0);
LED : out std_logic_vector (3 downto 0);
HEX : out std_logic_vector (6 downto 0);
HEX1 : out std_logic_vector (6 downto 0);
HEX2 : out std_logic_vector (6 downto 0) );
End abhesamiyulab7;

architecture Hexdecoder of abhesamiyulab7 is

Signal A : std_logic_vector (3 downto 0);

begin
    A <= Binary (3 downto 0);

Process (A, EN)
begin
  if (EN = '1')
then
```

```
case A is
when "0000" => HEX <= "0000001";
when "0001" => HEX <= "1001111";
when "0010" => HEX <= "0010010";
when "0100" => HEX <= "0000110";
when "0101" => HEX <= "1001100";
when "0110" => HEX <= "0100100";
when "0111" => HEX <= "0001111";
when "1000" => HEX <= "0000000";
when "1001" => HEX <= "0000100";
when "1010" => HEX <= "0001000";
when "1011" => HEX <= "1100000";
when "1100" => HEX <= "0110001";
when "1101" => HEX <= "1000010";
when "1110" => HEX <= "0110000";
when "1111" => HEX <= "0111000";
when others => HEX <= "1111111";
end case;
Case A is
when "0000" => HEX1 <= "0000001";
when "0001" => HEX1 <= "1001111";
when "0010" => HEX1 <= "0010010";
when "0011" => HEX1 <= "0000110";
when "0100" => HEX1 <= "1001100";
when "0101" => HEX1 <= "0100100";
when "0110" => HEX1 <= "0000000";
when "0111" => HEX1 <= "0001111";
when "1000" => HEX1 <= "0000000";
when "1001" => HEX1 <= "0000100";
when "1010" => HEX1 <= "0000001";
when "1011" => HEX1 <= "1001111";
when "1100" => HEX1 <= "0010010";
when "1101" => HEX1 <= "0000110";
when "1110" => HEX1 <= "1001100";
when "1111" => HEX1 <= "0100100";
when others => HEX1 <= "1111111";
when
end case;
```

```
Case A is
   when "0000" => HEX2 <= "1111111";
   when "0001" => HEX2 <= "1111111";
   when "0010" => HEX2 <= "1111111";
   when "0011" => HEX2 <= "1111111";
   when "0100" => HEX2 <= "1111111";
   when "0101" => HEX2 <= "1111111";
   when "0110" => HEX2 <= "1111111";
   when "0111" => HEX2 <= "1111111";
   when "1000" => HEX2 <= "1111111";
   when "1001" => HEX2 <= "1111111";
   when "1010" => HEX2 <= "1001111";
   when "1011" => HEX2 <= "1001111";
   when "1100" => HEX2 <= "1001111";
   when "1101" => HEX2 <= "1001111";
   when "1110" => HEX2 <= "1001111";
   when "1111" => HEX2 <= "1001111";
   when others => HEX2 <= "1111111";
   when others =>
   end case;
   LED <= A;
 else
    LED <= "0000";
    HEX <= "1111111";
    HEX1 <= "1111111";
    HEX2 <= "1111111";
 End if;
End process;
end HEX decoder;
```

Vlog Program:

```
module abhesaniya1ab77 (EN, Binary, HEX, HEX1, HEX2, LED
);

input [3:0] Binary;
input EN;
output [3:0] LED;
```

```
output [6:0]Hex;
output [6:0]Hex1;
output [6:0]Hex2;

reg [6:0] Hex;
reg [6:0] Hex1;
reg [6:0] Hex2;
reg [3:0] LED;
wire [3:0] A = Binary[3:0];

always @(EN or A)
begin
    if(EN==1)
    begin
    Hex = (A == 4'b0000) ? 7'b0000001 :
          (A == 4'b0001) ? 7'b1001111 :
          (A == 4'b0010) ? 7'b0010010 :
          (A == 4'b0011) ? 7'b0000110 :
          (A == 4'b0100) ? 7'b1001100 :
          (A == 4'b0101) ? 7'b0100100 :
          (A == 4'b0110) ? 7'b0100000 :
          (A == 4'b0111) ? 7'b0001111 :
          (A == 4'b1000) ? 7'b0000000 :
          (A == 4'b1001) ? 7'b0000100 :
          (A == 4'b1010) ? 7'b0001000 :
          (A == 4'b1011) ? 7'b1100000 :
          (A == 4'b1100) ? 7'b0110001 :
          (A == 4'b1101) ? 7'b1000010 :
          (A == 4'b1110) ? 7'b0110000 :
          (A == 4'b1111) ? 7'b0111000 :
          7'b1111111;

Hex1 = (A == 4'b0000) ? 7'b0000001 :
       (A == 4'b0001) ? 7'b1001111 :
       (A == 4'b0010) ? 7'b0010010 :
       (A == 4'b0011) ? 7'b0000110 :
       (A == 4'b0100) ? 7'b1001100 :
       (A == 4'b0101) ? 7'b0100100 :
```

```
(A == 4'b0110) ? 7'b 0100000 :
(A == 4'b0111) ? 7'b 0001111 :
(A == 4'b1000) ? 7'b 0000000 :
(A == 4'b1001) ? 7'b 0000100 :
(A == 4'b1010) ? 7'b 0000001 :
(A == 4'b1011) ? 7'b 1001111 :
(A == 4'b1100) ? 7'b 0010010 :
(A == 4'b1101) ? 7'b 0000110 :
(A == 4'b1110) ? 7'b 1001100 :
(A == 4'b1111) ? 7'b 0100100 :
7'b1111111 ;

Hex2 : (A == 4'b0000) ? 7'b 1111111 :
       (A == 4'b0001) ? 7'b 1111111 ;
       (A == 4'b0010) ? 7'b 1111111 ;
       (A == 4'b0011) ? 7'b 1111111 ;
       (A == 4'b0100) ? 7'b 1111111 ;
       (A == 4'b0101) ? 7'b 1111111 ;
       (A == 4'b0110) ? 7'b 1111111 ;
       (A == 4'b0111) ? 7'b 1111111 ;
       (A == 4'b1000) ? 7'b 1111111 ;
       (A == 4'b1001) ? 7'b 1111111 ;
       (A == 4'b1010) ? 7'b 1001111 ;
       (A == 4'b1011) ? 7'b 1001111 ;
       (A == 4'b1100) ? 7'b 1001111 ;
       (A == 4'b1101) ? 7'b 1001111 ;
       (A == 4'b1110) ? 7'b 1001111 ;
       (A == 4'b1111) ? 7'b 1001111 ;
        7'b1111111 ;
  LED = A;
 end
else
begin
     LED = 4'b0000 ;
     Hex = 7'b 1111111 ;
     Hex1 7'b 1111111 ;
     Hex2 7'b1111111 ;
 end
```

end
end module

Discussion