2.)

module m( D0, D1, S, Y);

input wire D0, D1, S;

output reg Y;

always @(D0 or D1 or S)

begin

if(S)

Y= D1;

else

Y=D0;

end

endmodule

3.)

`timescale 1ns / 1ps module

full\_adder( A, B, Cin, S, Cout);

input wire A, B, Cin;

output reg S, Cout;

always @(A or B or Cin)

begin

S = A ^ B ^ Cin;

Cout = A&B | (A^B) & Cin;

end

endmodule

4.)

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity decoder is

Port ( s : in STD\_LOGIC\_VECTOR (2 downto 0);

y : out STD\_LOGIC\_VECTOR (7 downto 0));

end decoder;

architecture Behavioral of decoder is

begin

with sel select

y<="00000001" when "000",

"00000010" when "001",

"00000100" when "010",

"00001000" when "011",

"00010000" when "100",

"00100000" when "101",

"01000000" when "110",

"10000000" when "111",

"00000000" when others;

end Behavioral;

5.)

**Logic module:**

module segment7(

bcd,

seg

);

input [3:0] bcd;

output [6:0] seg;

reg [6:0] seg;

always @(bcd)

begin

case (bcd)

0 : seg = 7'b0000001;

1 : seg = 7'b1001111;

2 : seg = 7'b0010010;

3 : seg = 7'b0000110;

4 : seg = 7'b1001100;

5 : seg = 7'b0100100;

6 : seg = 7'b0100000;

7 : seg = 7'b0001111;

8 : seg = 7'b0000000;

9 : seg = 7'b0000100;

default : seg = 7'b1111111;

endcase

end

endmodule

**testbench:**

module tb\_segment7;

reg [3:0] bcd;

wire [6:0] seg;

integer i;

segment7 uut (

.bcd(bcd),

.seg(seg)

);

initial begin

for(i = 0;i < 16;i = i+1)

begin

bcd = i;

#10;

end

end

endmodule