Lab 5

Verilog code- top module

```
module top(
    input clk,
    input [3:0] a,
    input [3:0] b,
    output [7:0] cathode,
    output [3:0] anode
    );

    wire [7:0] num;
    wire [3:0] thousands;
    wire [3:0] hundreds;
    wire [3:0] tens;
    wire [3:0] ones;

    mul_4x4 fun1(.x(a),.y(b),.pro(num));
    bin2bcd
fun2(.number(num),.thousands(thousands),.hundreds(hundreds),.tens(tens),.ones(ones));
    sevenseg
fun3(.clk(clk),.ones(ones),.tens(tens),.hundreds(hundreds),.thousands(thousands),.anode(anod
e),.cathode(cathode));

endmodule
```

Verilog code- seven segment display

```
`timescale 1ns / 1ps
module sevenseg(
    input clk,
//  input clr,              // clear pin used to reset the LED display
    input [3:0] ones,
    input [3:0] tens,
    input [3:0] hundreds,
    input [3:0] thousands,
```

```
//  input [3:0] num,
    output [7:0] cathode,
    output [3:0] anode
    );
     reg [6:0]sseg_temp;
     reg [3:0]an_temp = 4'b1110 ;
     reg [17:0] count; //the 18 bit counter which allows us to multiplex at 1000Hz

     always @ (posedge clk)
      begin
        count <= count + 1;
      end

            //code for display multiple digits (do not initialize an_temp in line 41 and comment out
lines 44 to 60)
     always @ (*)
      begin
       case(count[17:16]) //using only the 2 MSB's of the counter

        2'b00 :  //When the 2 MSB's are 00 enable the fourth display
         begin
          case(ones)
               4'd0 : sseg_temp = 7'b0000001; //to display 0
               4'd1 : sseg_temp = 7'b1001111; //to display 1
               4'd2 : sseg_temp = 7'b0010010; //to display 2
               4'd3 : sseg_temp = 7'b0000110; //to display 3
               4'd4 : sseg_temp = 7'b1001100; //to display 4
               4'd5 : sseg_temp = 7'b0100100; //to display 5
               4'd6 : sseg_temp = 7'b0100000; //to display 6
               4'd7 : sseg_temp = 7'b0001111; //to display 7
               4'd8 : sseg_temp = 7'b0000000; //to display 8
               4'd9 : sseg_temp = 7'b0000100; //to display 9
               default : sseg_temp = 7'b1111110; //dash
           endcase

          //sseg = ones;
          an_temp = 4'b1110;
         end

        2'b01:  //When the 2 MSB's are 01 enable the third display
         begin
            case(tens)
              4'd0 : sseg_temp = 7'b0000001; //to display 0
```

```
        4'd1 : sseg_temp = 7'b1001111; //to display 1
        4'd2 : sseg_temp = 7'b0010010; //to display 2
        4'd3 : sseg_temp = 7'b0000110; //to display 3
        4'd4 : sseg_temp = 7'b1001100; //to display 4
        4'd5 : sseg_temp = 7'b0100100; //to display 5
        4'd6 : sseg_temp = 7'b0100000; //to display 6
        4'd7 : sseg_temp = 7'b0001111; //to display 7
        4'd8 : sseg_temp = 7'b0000000; //to display 8
        4'd9 : sseg_temp = 7'b0000100; //to display 9
        default : sseg_temp = 7'b1111110; //dash
      endcase

  //sseg = tens;
  an_temp = 4'b1101;
 end

2'b10:  //When the 2 MSB's are 10 enable the second display
 begin
  case(hundreds)
        4'd0 : sseg_temp = 7'b0000001; //to display 0
        4'd1 : sseg_temp = 7'b1001111; //to display 1
        4'd2 : sseg_temp = 7'b0010010; //to display 2
        4'd3 : sseg_temp = 7'b0000110; //to display 3
        4'd4 : sseg_temp = 7'b1001100; //to display 4
        4'd5 : sseg_temp = 7'b0100100; //to display 5
        4'd6 : sseg_temp = 7'b0100000; //to display 6
        4'd7 : sseg_temp = 7'b0001111; //to display 7
        4'd8 : sseg_temp = 7'b0000000; //to display 8
        4'd9 : sseg_temp = 7'b0000100; //to display 9
        default : sseg_temp = 7'b1111110; //dash
   endcase

  //sseg = hundreds;
  an_temp = 4'b1011;
 end

2'b11:  //When the 2 MSB's are 11 enable the first display
 begin
  case(thousands)
        4'd0 : sseg_temp = 7'b0000001; //to display 0
        4'd1 : sseg_temp = 7'b1001111; //to display 1
        4'd2 : sseg_temp = 7'b0010010; //to display 2
        4'd3 : sseg_temp = 7'b0000110; //to display 3
```

```
            4'd4 : sseg_temp = 7'b1001100; //to display 4
            4'd5 : sseg_temp = 7'b0100100; //to display 5
            4'd6 : sseg_temp = 7'b0100000; //to display 6
            4'd7 : sseg_temp = 7'b0001111; //to display 7
            4'd8 : sseg_temp = 7'b0000000; //to display 8
            4'd9 : sseg_temp = 7'b0000100; //to display 9
            default : sseg_temp = 7'b1111110; //dash
          endcase

          //sseg = thousands;
          an_temp = 4'b0111;
         end
       endcase
      end

    assign anode = an_temp;
    assign cathode = {sseg_temp, 1'b1};
 endmodule
```

Verilog code- full adder

```
module full_adder(
input a,
input b,
input c,
output sum,
output carry
);

assign sum=a^b^c;
assign carry=(a & b) | (c&(a^b));

endmodule
```

Verilog code- 4x4 multiplier

```
module mul_4x4(
```

```
    input [3:0] x,
    input [3:0] y,
    output [7:0] pro
    );

    wire c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12;
    wire sum1,sum2,sum3,sum4,sum5,sum6;

    assign pro[0]=x[0] & y[0];

    full_adder fun1(.a(y[0]&x[1]),.b(y[1]&x[0]),.c(1'b0),.sum(pro[1]),.carry(c1));

    full_adder fun2(.a(y[0]&x[2]),.b(y[1]&x[1]),.c(c1),.sum(sum1),.carry(c2));
    full_adder fun3(.a(y[2]&x[0]),.b(sum1),.c(1'b0),.sum(pro[2]),.carry(c3));

    full_adder fun4(.a(y[0]&x[3]),.b(y[1]&x[2]),.c(c2),.sum(sum2),.carry(c4));
    full_adder fun5(.a(c3),.b(y[2]&x[1]),.c(sum2),.sum(sum3),.carry(c5));
    full_adder fun6(.a(y[3]&x[0]),.b(1'b0),.c(sum3),.sum(pro[3]),.carry(c6));

    full_adder fun7(.a(y[1]&x[3]),.b(1'b0),.c(c4),.sum(sum4),.carry(c7));
    full_adder fun8(.a(sum4),.b(y[2]&x[2]),.c(c5),.sum(sum5),.carry(c8));
    full_adder fun9(.a(y[3]&x[1]),.b(sum5),.c(c6),.sum(pro[4]),.carry(c9));

    full_adder fun10(.a(y[2]&x[3]),.b(c8),.c(c7),.sum(sum6),.carry(c10));
    full_adder fun11(.a(y[3]&x[2]),.b(sum6),.c(c9),.sum(pro[5]),.carry(c11));

    full_adder fun12(.a(y[3]&x[3]),.b(c11),.c(c10),.sum(pro[6]),.carry(c12));

    assign pro[7]=c12;


endmodule
```

Verilog code- binary to bcd

```
`timescale 1ns / 1ps
//below Binary to BCD is taken from the following link:
http://www.deathbylogic.com/2013/12/binary-to-binary-coded-decimal-bcd-converter/

module bin2bcd(number, thousands, hundreds, tens, ones);
```

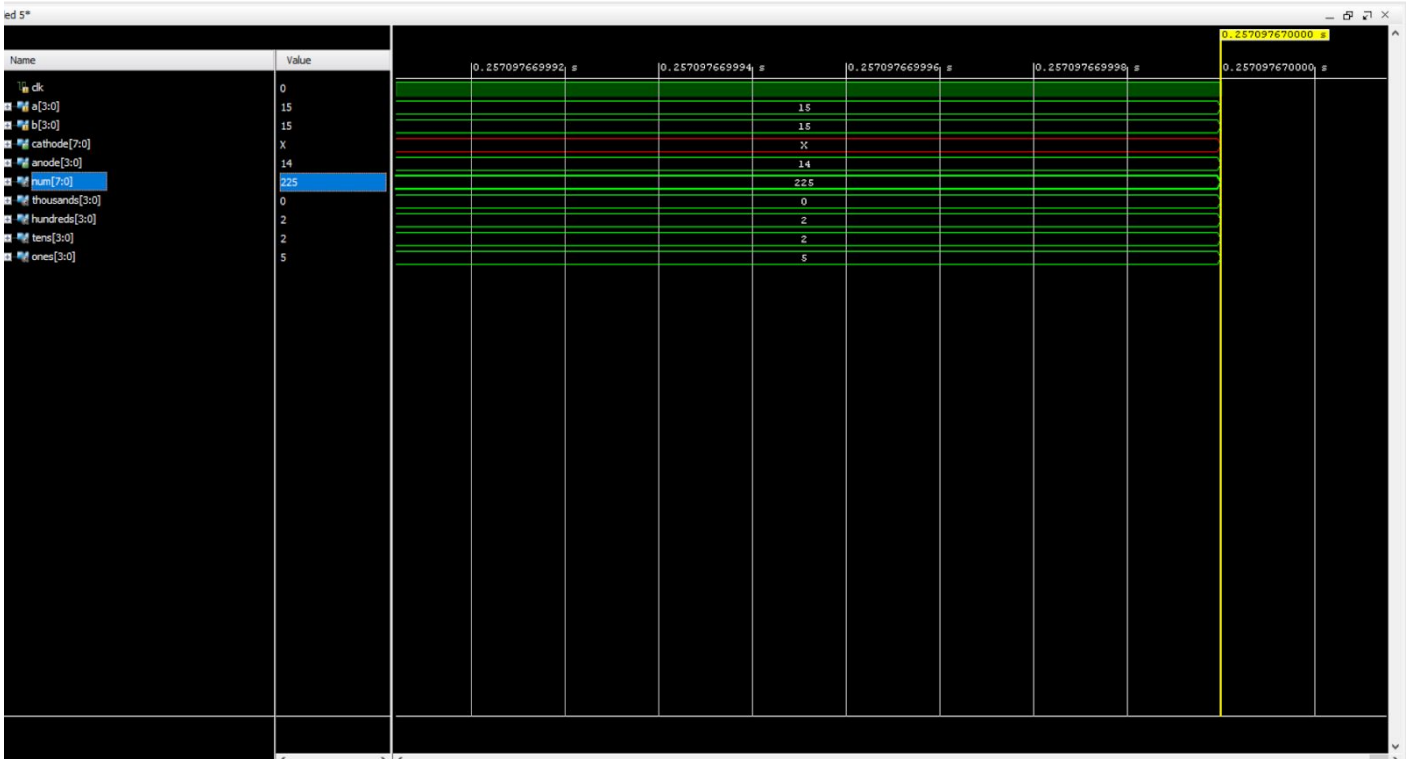
```
  // I/O Signal Definitions
  input  [7:0] number;
  output reg [3:0] thousands;
  output reg [3:0] hundreds;
  output reg [3:0] tens;
  output reg [3:0] ones;

  // Internal variable for storing bits
  reg [19:0] shift;
  integer i;

  always @(number)
  begin
    // Clear previous number and store new number in shift register
    shift[19:8] = 0;
    shift[7:0] = number;

    // Loop eight times
    for (i=0; i<8; i=i+1) begin
      if (shift[11:8] >= 5)
        shift[11:8] = shift[11:8] + 3;

      if (shift[15:12] >= 5)
        shift[15:12] = shift[15:12] + 3;

      if (shift[19:16] >= 5)
        shift[19:16] = shift[19:16] + 3;

      // Shift entire register left once
      shift = shift << 1;
    end

    // Push decimal numbers to output
    hundreds = shift[19:16];
    tens     = shift[15:12];
    ones     = shift[11:8];
    thousands= 4'b0000;

  end

endmodule
```

XDC file

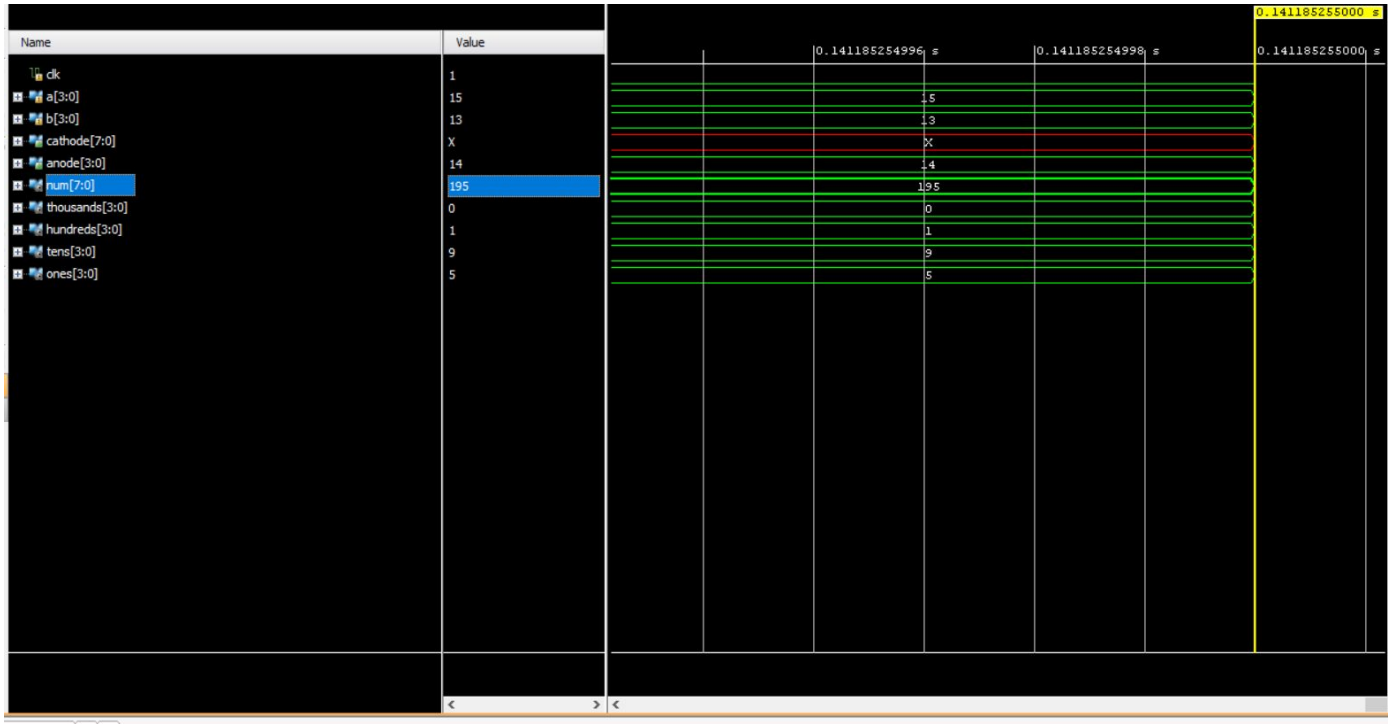
```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
        set_property IOSTANDARD LVCMOS33 [get_ports clk]

# Switches
set_property PACKAGE_PIN V17 [get_ports {a[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {a[0]}]
set_property PACKAGE_PIN V16 [get_ports {a[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {a[1]}]
set_property PACKAGE_PIN W16 [get_ports {a[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {a[2]}]
set_property PACKAGE_PIN W17 [get_ports {a[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {a[3]}]
set_property PACKAGE_PIN W15 [get_ports {b[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {b[0]}]
set_property PACKAGE_PIN V15 [get_ports {b[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {b[1]}]
set_property PACKAGE_PIN W14 [get_ports {b[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {b[2]}]
set_property PACKAGE_PIN W13 [get_ports {b[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {b[3]}]

#7 segment display
set_property PACKAGE_PIN W7 [get_ports {cathode[7]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[7]}]
set_property PACKAGE_PIN W6 [get_ports {cathode[6]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[6]}]
set_property PACKAGE_PIN U8 [get_ports {cathode[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[5]}]
set_property PACKAGE_PIN V8 [get_ports {cathode[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[4]}]
set_property PACKAGE_PIN U5 [get_ports {cathode[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[3]}]
set_property PACKAGE_PIN V5 [get_ports {cathode[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[2]}]
set_property PACKAGE_PIN U7 [get_ports {cathode[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {cathode[1]}]
set_property PACKAGE_PIN V7 [get_ports cathode[0]]
   set_property IOSTANDARD LVCMOS33 [get_ports cathode[0]]
```

```
set_property PACKAGE_PIN U2 [get_ports {anode[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {anode[0]}]
set_property PACKAGE_PIN U4 [get_ports {anode[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {anode[1]}]
set_property PACKAGE_PIN V4 [get_ports {anode[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {anode[2]}]
set_property PACKAGE_PIN W4 [get_ports {anode[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {anode[3]}]
```

Outputs



A= 15
B= 15
Product= 225 (num)



A= 15
B= 13
Product= 195 (num)