FPGA MID-TERM LAB

NAME - ARYAMAN PATHAK
ROLL NUMBER -IMT2022513

1) Code

```
//ARYAMAN PATHAK
module  blink(
    input clk,
    output reg led
    );

reg [31:0]count;

 initial begin
//       updated
       count =0;
       led=0;


    end

always @(posedge clk) begin

if(count == 9) begin //blink
    count <= 0;            //Reset count register
    led <= ~led;           //Toggle led (in each second)
end else begin
    count <= count + 1;
    end

end

endmodule
```

2) **Testbench**

```
//ARYAMAN PATHAK
`timescale 1ns / 1ps  // Set the time scale for simulation (1ns time unit, 1ps precision)

module tb;


reg clk;     // clk should be reg, not wire, because it is driven by always block
wire led;   // reset should be reg, not wire


blink blinkled(
     .clk(clk),
    .led(led));
always begin
    #5 clk = ~clk;  // Toggle clock every 5 time units (10ns clock period)
end

initial begin
    // Initialize signals

    clk = 0;









    $finish;        // End simulation
end

endmodule
```

3)

Code->

```
//ARYAMAN PATHAK
module  blink(
    input clk,
    output reg led
    );
```

```
reg [31:0]count;

 initial begin
//       updated
       count =0;
       led=0;


    end

always @(posedge clk) begin

if(count == 99999999) begin //blink
    count <= 0;             //Reset count register
    led <= ~led;            //Toggle led (in each second)
end else begin
    count <= count + 1;
    end

end

endmodule
```

4)

Code

```
//ARYAMAN PATHAK

module  blink(
    input clk,
    input rst,
    output reg led1,
    output reg led2,
    output reg led3


    );

reg [31:0]count;
reg[1:0] led_number=0;
```

```
 initial begin
//       updated
       count =0;
       led1=0;
       led2=0;
       led3=0;


    end

always @(posedge clk or posedge rst) begin
 if(rst==1)
 begin
 count<=0;
 led_number<=0;
 led1<=0;
 led2<=0;
 led3<=0;
 end
 else begin
if(count == 99999999) begin //blink
    count <= 0;             //Reset count register
    if(led_number==0)
    begin
       led1 <= ~led1;      //Toggle led1
       led_number <= 1;
       end
     if(led_number==1)
    begin
       led2 <= ~led2;      //Toggle led1
       led_number <= 2;
       end
     if(led_number==2)
    begin
       led3 <= ~led3;      //Toggle led1
       led_number <= 0;
       end


end else begin
    count <= count + 1;
    end
end
```

```
end

endmodule
```

Constraints.xdc

```
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports led1]
set_property IOSTANDARD LVCMOS33 [get_ports led2]
set_property IOSTANDARD LVCMOS33 [get_ports led3]
set_property IOSTANDARD LVCMOS33 [get_ports rst]
set_property PACKAGE_PIN W5 [get_ports clk]

set_property PACKAGE_PIN U19 [get_ports led1]
set_property PACKAGE_PIN E19 [get_ports led2]
set_property PACKAGE_PIN U16 [get_ports led3]
set_property PACKAGE_PIN R2 [get_ports rst]
```

5)

blinking .v

```
//ARYAMAN PATHAK
module  blink(
    input clk,
    input rst,
    output reg led1,
    output reg led2,
    output reg led3,
    output reg[15:0] led_counter

    );

reg [31:0]count;
reg[1:0] led_number=0;
 initial begin
//       updated
       count =0;
       led1=0;
       led2=0;
       led3=0;
       led_counter=0;
```

```
    end

always @(posedge clk or posedge rst) begin

 if(rst==1)
 begin
 count<=0;
 led_number<=0;
 led1<=0;
 led2<=0;
 led3<=0;
 led_counter<=0;
 end
 else
if(count == 99999999) begin //blink
    count <= 0;
    led_counter=led_counter+1;            //Reset count register
    if(led_number==0)
    begin
        led1 <= ~led1;      //Toggle led1
        led_number <= 1;
        end
      if(led_number==1)
    begin
        led2 <= ~led2;      //Toggle led1
        led_number <= 2;
        end
      if(led_number==2)
    begin
        led3 <= ~led3;      //Toggle led1
        led_number <= 0;
        end

end else begin
    count <= count + 1;
    end

end

endmodule
```

7segment_led.v

```
//ARYAMAN PATHAK

// fpga4student.com: FPGA projects, Verilog projects, VHDL projects
// FPGA tutorial: seven-segment LED display controller on Basys  3 FPGA
module Seven_segment_LED_Display_Controller(
   input clock_100Mhz, // 100 Mhz clock source on Basys 3 FPGA
   input reset, // reset
   output reg [3:0] Anode_Activate, // anode signals of the 7-segment LED display
   output reg [6:0] LED_out,// cathode patterns of the 7-segment LED display
   output wire led1,
   output wire led2,
   output wire led3
   );

   reg [26:0] one_second_counter; // counter for generating 1 second clock enable
   wire one_second_enable;// one second enable for counting numbers
   wire [15:0] displayed_number; // counting number to be displayed
   reg [3:0] LED_BCD;
   reg [19:0] refresh_counter; // 20-bit for creating 10.5ms refresh period or 380Hz
refresh rate
            // the first 2 MSB bits for creating 4 LED-activating signals with 2.6ms digit
period
   wire [1:0] LED_activating_counter;
                // count     0    ->  1  ->  2  ->  3
             // activates    LED1    LED2   LED3   LED4
            // and repeat



   blink blinking(
   clock_100Mhz, reset,
    led1,
   led2,
   led3,
    displayed_number

   );
   always @(posedge clock_100Mhz or posedge reset)
```

```
    begin
        if(reset==1)
            one_second_counter <= 0;
        else begin
            if(one_second_counter>=99999999)
                one_second_counter <= 0;
            else
                one_second_counter <= one_second_counter + 1;
        end
    end
    assign one_second_enable = (one_second_counter==99999999)?1:0;
//    always @(posedge clock_100Mhz or posedge reset)
//    begin
//        if(reset==1)
////            displayed_number <= 0;
//        else if(one_second_enable==1)
////            displayed_number <= displayed_number + 1;
//    end
    always @(posedge clock_100Mhz or posedge reset)
    begin
        if(reset==1)
            refresh_counter <= 0;
        else
            refresh_counter <= refresh_counter + 1;
    end
    assign LED_activating_counter = refresh_counter[19:18];
    // anode activating signals for 4 LEDs, digit period of 2.6ms
    // decoder to generate anode signals
    always @(*)
    begin
        case(LED_activating_counter)
        2'b00: begin
            Anode_Activate = 4'b0111;
            // activate LED1 and Deactivate LED2, LED3, LED4
            LED_BCD = displayed_number/1000;
            // the first digit of the 16-bit number
              end
        2'b01: begin
            Anode_Activate = 4'b1011;
            // activate LED2 and Deactivate LED1, LED3, LED4
            LED_BCD = (displayed_number % 1000)/100;
            // the second digit of the 16-bit number
              end
        2'b10: begin
```

```
            Anode_Activate = 4'b1101;
            // activate LED3 and Deactivate LED2, LED1, LED4
            LED_BCD = ((displayed_number % 1000)%100)/10;
            // the third digit of the 16-bit number
               end
         2'b11: begin
            Anode_Activate = 4'b1110;
            // activate LED4 and Deactivate LED2, LED3, LED1
            LED_BCD = ((displayed_number % 1000)%100)%10;
            // the fourth digit of the 16-bit number
              end
         endcase
      end
      // Cathode patterns of the 7-segment LED display
      always @(*)
      begin
         case(LED_BCD)
         4'b0000: LED_out = 7'b0000001; // "0"
         4'b0001: LED_out = 7'b1001111; // "1"
         4'b0010: LED_out = 7'b0010010; // "2"
         4'b0011: LED_out = 7'b0000110; // "3"
         4'b0100: LED_out = 7'b1001100; // "4"
         4'b0101: LED_out = 7'b0100100; // "5"
         4'b0110: LED_out = 7'b0100000; // "6"
         4'b0111: LED_out = 7'b0001111; // "7"
         4'b1000: LED_out = 7'b0000000; // "8"
         4'b1001: LED_out = 7'b0000100; // "9"
         default: LED_out = 7'b0000001; // "0"
         endcase
      end
   endmodule
```

Constraint.xdc

```
set_property IOSTANDARD LVCMOS33 [get_ports led1]
set_property IOSTANDARD LVCMOS33 [get_ports led2]
set_property IOSTANDARD LVCMOS33 [get_ports led3]


# Clock signal
```

```
set_property PACKAGE_PIN W5 [get_ports clock_100Mhz]
 set_property IOSTANDARD LVCMOS33 [get_ports clock_100Mhz]
set_property PACKAGE_PIN R2 [get_ports reset]
 set_property IOSTANDARD LVCMOS33 [get_ports reset]
#seven-segment LED display
set_property PACKAGE_PIN W7 [get_ports {LED_out[6]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED_out[6]}]
set_property PACKAGE_PIN W6 [get_ports {LED_out[5]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED_out[5]}]
set_property PACKAGE_PIN U8 [get_ports {LED_out[4]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED_out[4]}]
set_property PACKAGE_PIN V8 [get_ports {LED_out[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED_out[3]}]
set_property PACKAGE_PIN U5 [get_ports {LED_out[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED_out[2]}]
set_property PACKAGE_PIN V5 [get_ports {LED_out[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED_out[1]}]
set_property PACKAGE_PIN U7 [get_ports {LED_out[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {LED_out[0]}]
set_property PACKAGE_PIN U2 [get_ports {Anode_Activate[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Anode_Activate[0]}]
set_property PACKAGE_PIN U4 [get_ports {Anode_Activate[1]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Anode_Activate[1]}]
set_property PACKAGE_PIN V4 [get_ports {Anode_Activate[2]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Anode_Activate[2]}]
set_property PACKAGE_PIN W4 [get_ports {Anode_Activate[3]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {Anode_Activate[3]}]


set_property PACKAGE_PIN U19 [get_ports led1]
set_property PACKAGE_PIN E19 [get_ports led2]
set_property PACKAGE_PIN U16 [get_ports led3]
```