```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/13/2022 09:35:17 PM
// Design Name:
// Module Name: Lab3_sim
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Lab3_sim();
wire [6:0] seg;
wire dp;
wire [3:0] an;
reg [7:0]sw;
reg btnC, btnL, clkin, btnR;
wire [7:0] D7Seg3, D7Seg2, D7Seg1, D7Seg0;


Top_level UUT(
    .sw(sw[7:0]), .btnL(btnL), .btnC(btnC), .btnR(btnR), .clkin(clkin),
.seg(seg[6:0]), .dp(dp),
    .an(an[3:0])

);

show_7segDisplay  showit (
    .seg(seg[6:0]),
    .DP(dp), .AN0(an[0]), .AN1(an[1]), .AN2(an[2]), .AN3(an[3]),
    .D7Seg0(D7Seg0), .D7Seg1(D7Seg1), .D7Seg2(D7Seg2), .D7Seg3(D7Seg3)
  );
```

```
parameter PERIOD = 10;
   parameter real DUTY_CYCLE = 0.5;
   parameter OFFSET = 2;

   initial    // Clock process for clkin
   begin
       #OFFSET
           clkin = 1'b1;
       forever
       begin
           #(PERIOD-(PERIOD*DUTY_CYCLE)) clkin = ~clkin;
       end
   end

  initial
  begin
   btnR = 1'b0;
   #50;
   sw = 8'b00000000;
   btnL = 1'b0;
   btnC = 1'b0;

   #50;
   sw = 8'b00010001;
   // shows 1 1
   #50;
   sw = 8'b00100010;
   // shows 2 2
   #50;
   sw = 8'b00110011;
   //shows 3 3
   #50;
   sw = 8'b01000100;
   //shows 4 4
   #50;
   sw = 8'b01010101;
   //shows 5 5
   #50;
   sw = 8'b01100110;
   // shows 6 6
   #50;
   sw = 8'b01110111;
   // shows 7 7
   #50;
   sw = 8'b10001000;
   // shows    8 8
```

```
    #50;
    sw = 8'b10011001;
    //shows  9 9
    #50;
    sw = 8'b10101010;
    //shows A A (10 10)
    #50;
    sw = 8'b10111011;
    //shows b b (11 11)
    #50;
    sw = 8'b11001100;
    //shows C C (12 12)
    #50;
    sw = 8'b11111101;
    // shows F D (15 13)
    #50;
    sw = 8'b01101001;
    // shows 6 9 (ha nice)
    #50;
    sw = 8'b01000010;
    // shows 4 2
    #50;
    sw = 8'b10000000;
    //shows 8 0
     #50;
   end
endmodule
module show_7segDisplay (
  input [6:0] seg,
  input DP,AN0,AN1,AN2,AN3,
  output reg [7:0] D7Seg0, D7Seg1, D7Seg2,D7Seg3);

  reg [7:0] val;

  always @(AN0 or val)
  begin
    if (AN0 == 0) D7Seg0 <= val;
    else if (AN0 == 1) D7Seg1 <= " ";
    else D7Seg0 <= 8'bX;   //  non-blocking assignment
  end

  always @(AN1 or val)
  begin
    if (AN1 == 0) D7Seg1 <= val;
    else if (AN1 == 1) D7Seg1 <= " ";
    else D7Seg1 <= 8'bX;   //  non-blocking assignment
```

```
end

always @(AN2 or val)
begin
  if (AN2 == 0) D7Seg2 <= val;
  else if (AN2 == 1) D7Seg2 <= " ";
  else D7Seg2 <= 8'bX;   //  non-blocking assignment
end

always @(AN3 or val)
begin
  if (AN3 == 0) D7Seg3 <= val;
  else if (AN3 == 1) D7Seg3 <= " ";
  else D7Seg3 <= 8'bX;   //  non-blocking assignment
end

always @(seg)
case (seg)
7'b0111111:
     val = "-";
7'b1111111:
     val = " ";
7'b1000000:
     val = "0";
7'b1111001:
     val = "1";
7'b0100100:
     val = "2";
7'b0110000:
     val = "3";
7'b0011001:
     val = "4";
7'b0010010:
     val = "5";
7'b0000010:
     val = "6";
7'b1111000:
     val = "7";
7'b0000000:
     val = "8";
7'b0011000:
     val = "9";
7'b0001000:
     val = "A";
7'b0000011:
     val = "B";
  7'b1000110:
       val = "C";
  7'b0100001:
       val = "D";
  7'b0000110:
       val = "E";
  7'b0001110:
       val = "F";
  default:
       val = 8'bX;
  endcase
endmodule
```