```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 11/22/2020 07:26:27 PM
// Design Name:
// Module Name: Test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Test();
  reg clkin, btnR, btnL, btnU;
  wire [15:0] led;
  wire [6:0] seg;
  wire [3:0] an;
  wire dp;
  wire [7:0] D7Seg3,D7Seg2,D7Seg1,D7Seg0;

  Top_Level
   UUT (
      .clkin(clkin),
      .btnR(btnR),
      .btnL(btnL),
      .btnU(btnU),
      .seg(seg),
      .dp(dp),
      .led(led),
      .an(an)
      );
  show_7segDisplay  showit (.seg(seg),.dp(dp),.an(an),
               .D7Seg0(D7Seg0),.D7Seg1(D7Seg1),.D7Seg2(D7Seg2),.D7Seg3(D7Seg3));


// Run this simulation for 3ms. If correct TX_ERROR should be 0 at the end.
// UTC should be high and then go low at 2,705us and go low at 2,706.3us.

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


    initial    // Clock process for clkin
    begin
      btnU = 1'b0;

    #OFFSET
        clkin = 1'b1;
    forever
    begin
      #(PERIOD-(PERIOD*DUTY_CYCLE)) clkin = ~clkin;
    end
    end

    initial
    begin
    // Going Left to Right
    #100;
    btnL=1'b1;
    btnR=1'b0;
    #200;
    btnL=1'b1;
    btnR=1'b1;
    #300;
    btnL=1'b0;
    btnR=1'b1;
    #400;
    btnL=1'b0;
    btnR=1'b0;
    // Going Right to Left
    #500;
    btnL=1'b0;
    btnR=1'b1;
    #600;
    btnL=1'b1;
    btnR=1'b1;
    #700;
    btnL=1'b1;
    btnR=1'b0;
    #800;
    btnL=1'b0;
```

```
btnR=1'b0;
// Enter Left and retreating Left
#900;
btnL=1'b1;
btnR=1'b0;
#1000;
btnL=1'b1;
btnR=1'b1;
#1100;
btnL=1'b1;
btnR=1'b0;
#1200;
btnL=1'b0;
btnR=1'b0;
// Enter Left, Wander, and going Right
#1300;
btnL=1'b1;
btnR=1'b0;
#1400;
btnL=1'b1;
btnR=1'b1;
#1500;
btnL=1'b0;
btnR=1'b1;
#1600;
btnL=1'b1;
btnR=1'b1;
#1700;
btnL=1'b0;
btnR=1'b1;
#1800;
btnL=1'b0;
btnR=1'b0;
// Enter Right and retreating Right
#1900;
btnL=1'b0;
btnR=1'b1;
#2000;
btnL=1'b1;
btnR=1'b1;
#2100;
btnL=1'b1;
btnR=1'b0;
#2200;
btnL=1'b1;
btnR=1'b1;
```

```
    #2300;
    btnL=1'b0;
    btnR=1'b1;
    #2400;
    btnL=1'b0;
    btnR=1'b0;
    // Enter Right, Wander Leave Left
    #2500;
    btnL=1'b0;
    btnR=1'b1;
    #2600;
    btnL=1'b1;
    btnR=1'b1;
    #2700;
    btnL=1'b1;
    btnR=1'b0;
    #2800;
    btnL=1'b1;
    btnR=1'b1;
    #2900;
    btnL=1'b1;
    btnR=1'b0;
    #3000;
    btnL=1'b0;
    btnR=1'b0;
    // Wander for 15 Sec
    #3100;
    btnL=1'b0;
    btnR=1'b1;
    #3200;
    btnL=1'b1;
    btnR=1'b1;
    #3300;
    btnL=1'b1;
    btnR=1'b0;
    #3400;
    btnL=1'b1;
    btnR=1'b1;
    #3500;
    btnL=1'b1;
    btnR=1'b1;
    #3600;
    btnL=1'b1;
    btnR=1'b0;
    #3700;
    btnL=1'b1;
    btnR=1'b1;
    #3800;
    btnL=1'b1;
    btnR=1'b0;
    #3900;
    btnL=1'b1;
    btnR=1'b1;
    #4000;
    btnL=1'b1;
    btnR=1'b0;
    #4100;
    btnL=1'b1;
    btnR=1'b1;
    #4200;
    btnL=1'b1;
    btnR=1'b0;
    #4300;
    btnL=1'b1;
    btnR=1'b1;
    #4400;
    btnL=1'b1;
    btnR=1'b0;
    #4500;
    btnL=1'b1;
    btnR=1'b0;
    #4600;
    end

endmodule


module show_7segDisplay (
 input [6:0] seg,
 input dp,
 input [3:0] an,
 output reg [7:0] D7Seg0, D7Seg1, D7Seg2,D7Seg3);

 reg [7:0] val;

 wire AN0, AN1, AN2, AN3;
 assign AN0=an[0];
 assign AN1=an[1];
 assign AN2=an[2];
 assign AN3=an[3];

  always @(AN0 or val)
```

```
begin
  if (AN0 == 0) D7Seg0 <= val;
  else if (AN0 == 1) D7Seg0 <= " ";
  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
```