Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

ASSIGNMENT-2

DIGITAL SYSTEM DESIGN

SUBMITTING TO: SHREEJITH SHANKER

SUBMITTED BY: SHRUTI KATHURIA

Aim:

The purpose of this session is to bring together elements from all your Labs, but particularly Lab F-G, and give you experience with developing and testing a larger sequential design and targeting it to the FPGA board. This assignment is worth 25% of 3C7.

Sequence Detector

You need to design a system that comprises an LFSR, a sequence detecting Finite State Machine (FSM), and a counter, in order to count the number of times a certain codeword (10, 12 bits long) is issued in the stream of bits generated by the LFSR in a full cycle of that LFSR. Each student will be given a different sequence/codeword, found in a separate handout `LFSR_code_setup.pdf`.

DATA USED IN THE ASSIGNMENT

TCD ID	NAME		BOARD NUMBER	LFSR_WIDTH		SEED VALUE	PATTERN
21355061	KATHURIA	SHRUTI	Basys3108	24	xor	100111111001000000000000	11001010000

WORKING OF THE DESIGN:

At first, we made different modules

•Clock:

This helps us to control the speed of the BASYS 3 board. The output of the clock is send to the LFSR module from which we obtain a 24 bit number and max tick indicating that there are 2*24-1 cycles.

•State Machine Module:

In this we drew the Moore diagram of the state 11001010000 and then minimized the state table. Inside the state machine I gave 4-bit long variables to store my state as 11 bits were available and that can be stored in 4 bits. Used a basic switch case for determining the present and next state.

•Counter:

Now this 1 bit found was provided to the counter which also worked on the clock edge, if the found was high the bcd counter increased its count by 1 until it is either cleared by reset or max tick or crosses the value of 9999.

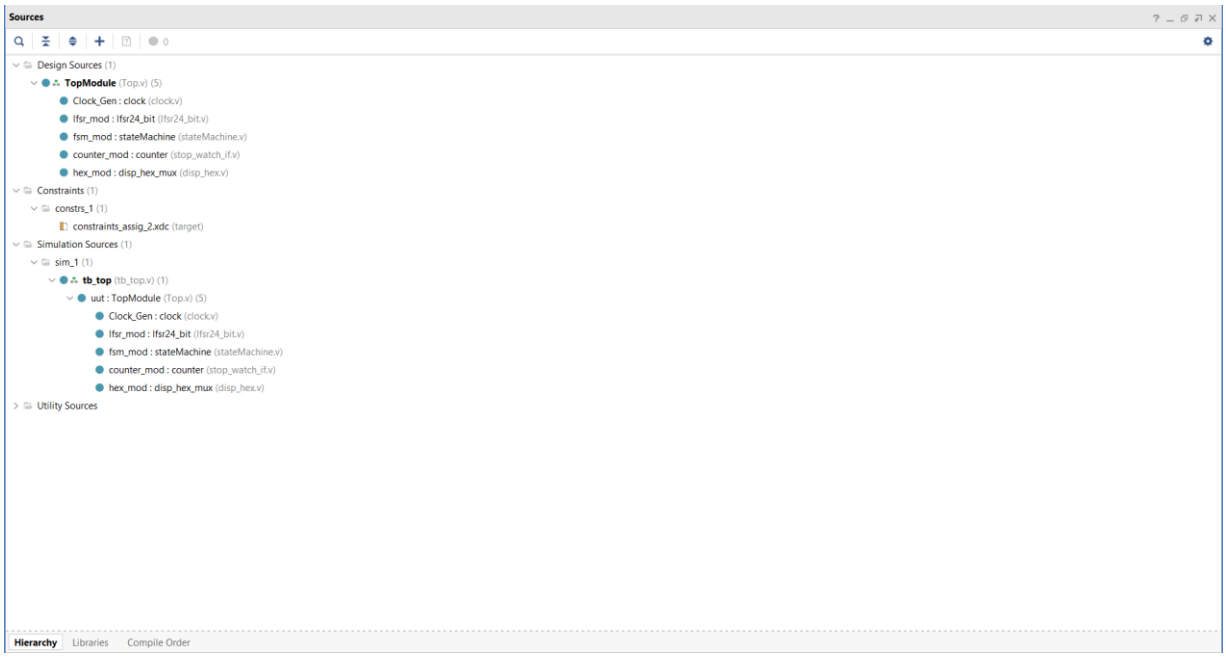
•Seven Segment Display:

We had to display the count on the seven-segment display of thebasys-3 board. Using the provide code for displaying hex from Lab G. The only thing that had to be taken care of was that the clock provide must be the boards one as we need high speed to execute the multiplexing on the board.

While Designing the Finite State Machine the overlapping is considered.

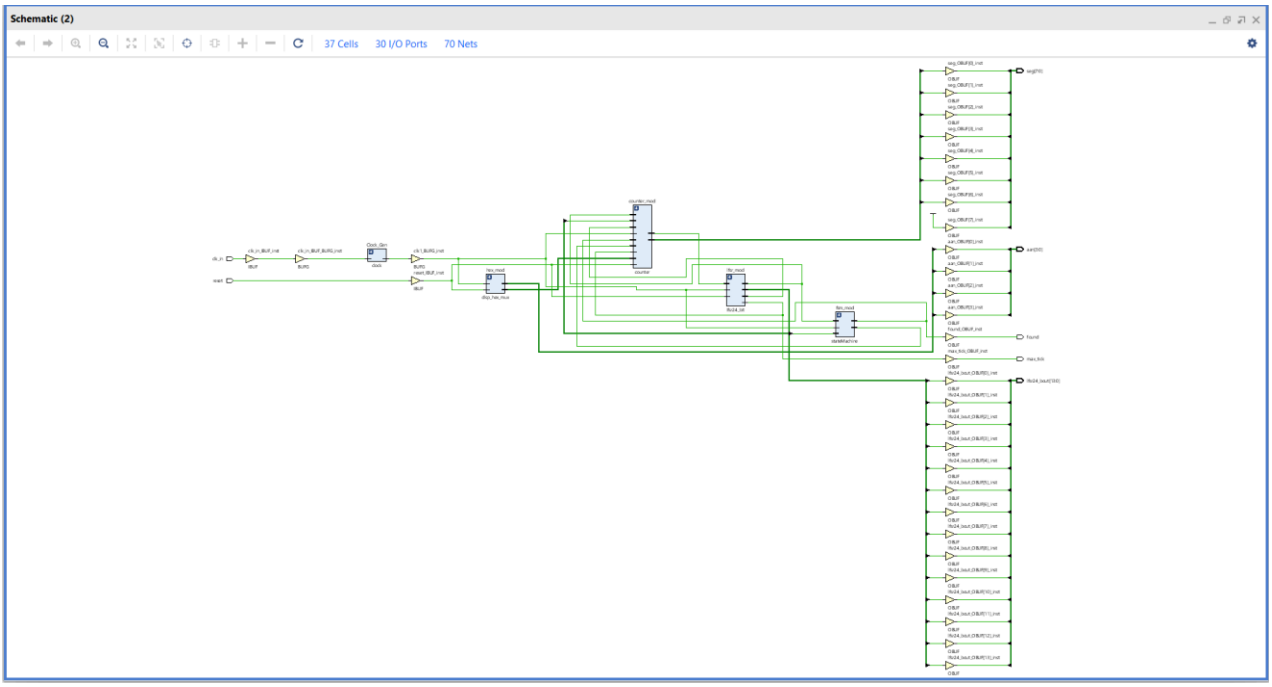
LFSR:

The seed value is given according to which the LFSR code is written and the tap values are taken from the XILINUX table.We should get the seed value when the reset is high in LFSR.



Hierarchy

SCHEMATIC DIAGRAM:



TESTBENCH:

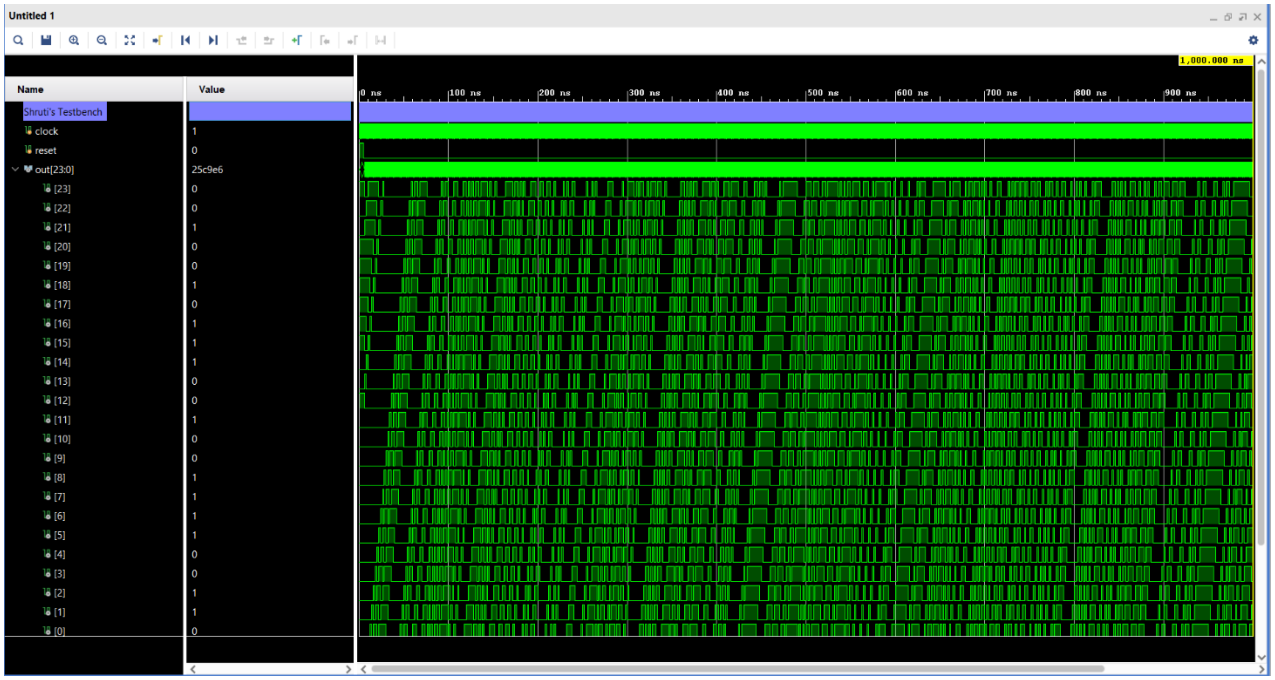
Reset is set to high

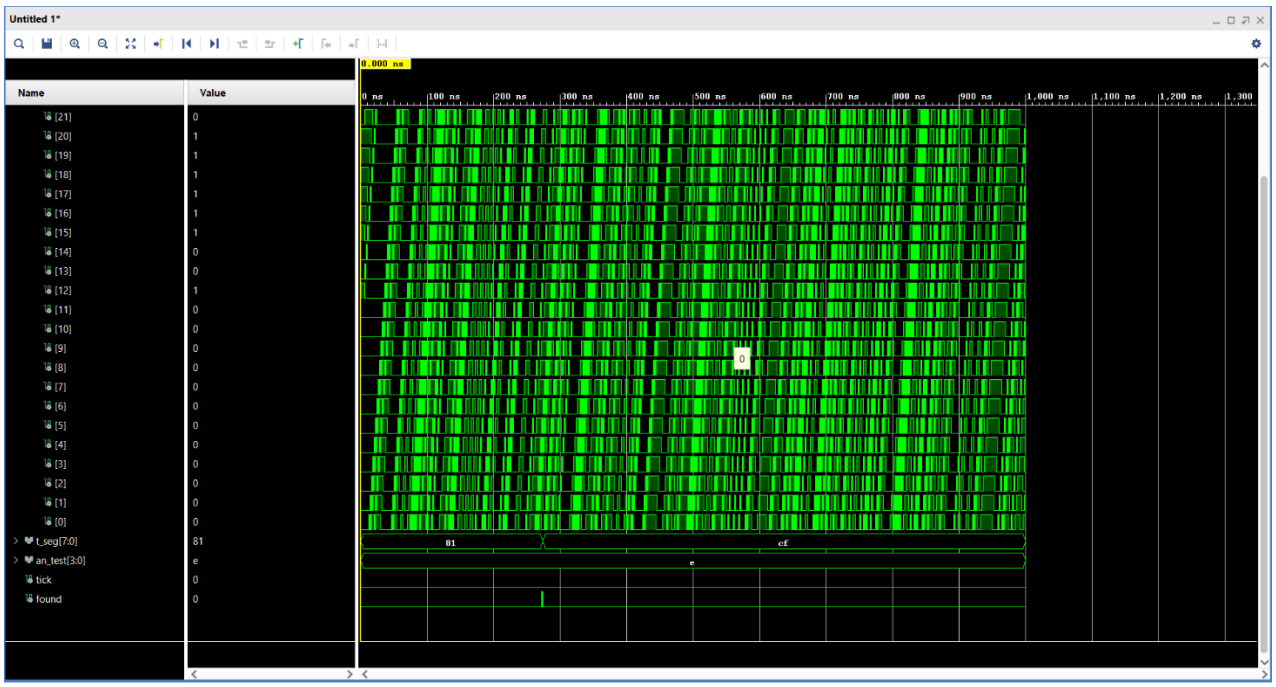
tb_top.v

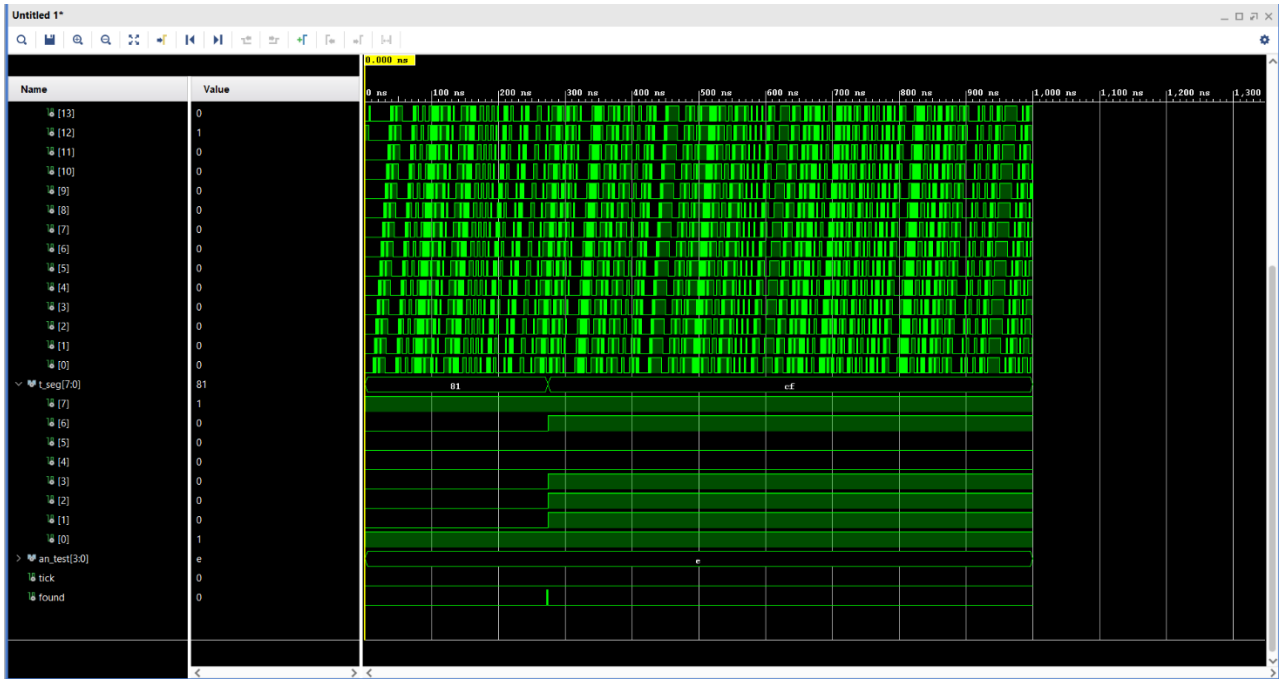
C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sim_1/new/tb_top.v

```
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////


module tb_top;
reg clock;
reg reset;
wire [23:0] out;
wire [7:0] t_seg;
wire [3:0] an_test;
wire tick;
wire found;


TopModule uut (.clk_in(clock), .reset(reset),.lfsr24_out(out),.aan(an_test),.seg(t_seg),.max_tick(tick),.found(found));

initial
begin
    clock = 1'b1;
    reset=1'b1;
    #5;
    reset =1'b0;
end

always

begin
    #1 clock = ~clock; // negate to oscillate
end
endmodule
```

When we use this testbench the output is:


Untitled 1
Name
Value
Shruti's Testbench
clock
reset
out[23:0]
25c9e6
1,000.000 ns


Untitled 1*
Name
Value
t_seg[7:0]
an_test[3:0]
tick
found
0.000 ns



We can see whenever the found is high, on the next positive clock edge the counter increase the count.

FINDING THE PATTERN ACCORDING TO MOORE:

As there are 11 bits we take 12 bits to make the state diagram. Among which state A and L are same so it gets minimised.

Current State	Next State Input 0	Next State Input 1	Output Input 0	Output Input 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	E	B	0	0
E	A	F	0	0
F	G	A	0	0
G	A	H	0	0
H	I	C	0	0
I	J	B	0	0
J	K	B	0	0
K	L	B	1	0

B	B,C										
C	A,D B,C	A,D									
D	A,E	A,E C,B	D,E C,B								
E	B,F	C,F	A,D C,F	A,E B,F							
F	A,G A,B	A,G A,C	D,G A,C	E,G A,B	A,G A,F						
G	B,H	A,H	A,D C,H	A,E B,H	F,H	A,G A,H					
H	A,I B,C	A,I	D,I	E,I B,C	A,I C,F	G,I A,C	A,I C,H				
I	A,J	A,J C,B	D,J C,B	E,J	A,J B,F	G,J A,B	A,G H,B	I,J C,B			
J	A,K	A,K C,B	D,K C,B	E,K	A,K B,F	G,K A,B	A,K B,H	I,K C,B	J,K		
K	~~A,L~~	~~A,L C,B~~	~~D,L C,B~~	~~E,L~~	~~A,L F,B~~	~~G,L A,B~~	~~A,L B,H~~	~~I,K C,B~~	~~J,L~~	~~K,L~~	
L	✓	C,B	D,A C,B	A,E	F,B	A,G A,B	H,B	I,A C,B	A,J	A,K	~~A,L~~
	A	B	C	D	E	F	G	H	I	J	K

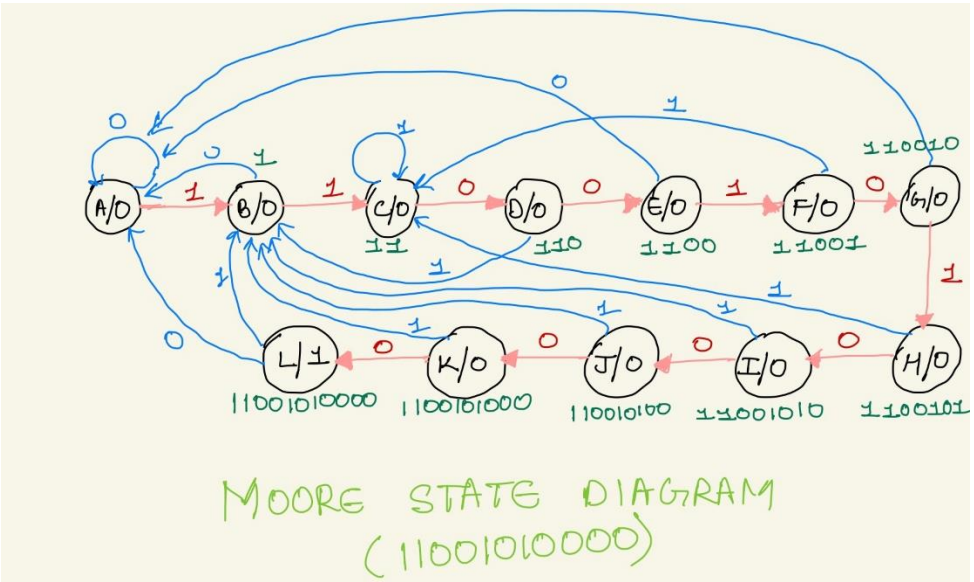
A,J L

B	~~B,C~~										
C	~~A,D B,C~~	~~A,D~~									
D	~~A,E~~	~~A,E C,B~~	~~D,E C,B~~								
E	~~B,F~~	~~C,F~~	~~A,D C,F~~	~~A,E B,F~~							
F	~~A,G A,B~~	~~A,G A,C~~	~~D,G A,C~~	~~E,G A,B~~	~~A,G A,F~~						
G	~~B,H~~	~~A,H~~	~~A,D C,H~~	~~A,E B,H~~	~~F,H~~	~~A,G A,H~~					
H	~~A,I B,C~~	~~A,I~~	~~D,I~~	~~E,I B,C~~	~~A,I C,F~~	~~G,I A,C~~	~~A,I C,H~~				
I	~~A,J~~	~~A,J C,B~~	~~D,J C,B~~	~~E,J~~	~~A,J B,F~~	~~G,J A,B~~	~~A,J H,B~~	~~I,J C,B~~			
J	~~A,K~~	~~A,K C,B~~	~~D,K C,B~~	~~E,K~~	~~A,K B,F~~	~~G,K A,B~~	~~A,K B,H~~	~~I,K C,B~~	~~J,K~~		
K	~~A,L~~	~~A,L C,B~~	~~D,L C,B~~	~~E,L~~	~~A,L F,B~~	~~G,L A,B~~	~~A,L B,H~~	~~I,K C,B~~	~~J,L~~	~~K,L~~	
L	✓	~~C,B~~	~~D,A C,B~~	~~A,E~~	~~F,B~~	~~A,G A,B~~	~~H,B~~	~~I,A C,B~~	~~A,J~~	~~A,K~~	~~A,L~~
	A	B	C	D	E	F	G	H	I	J	K

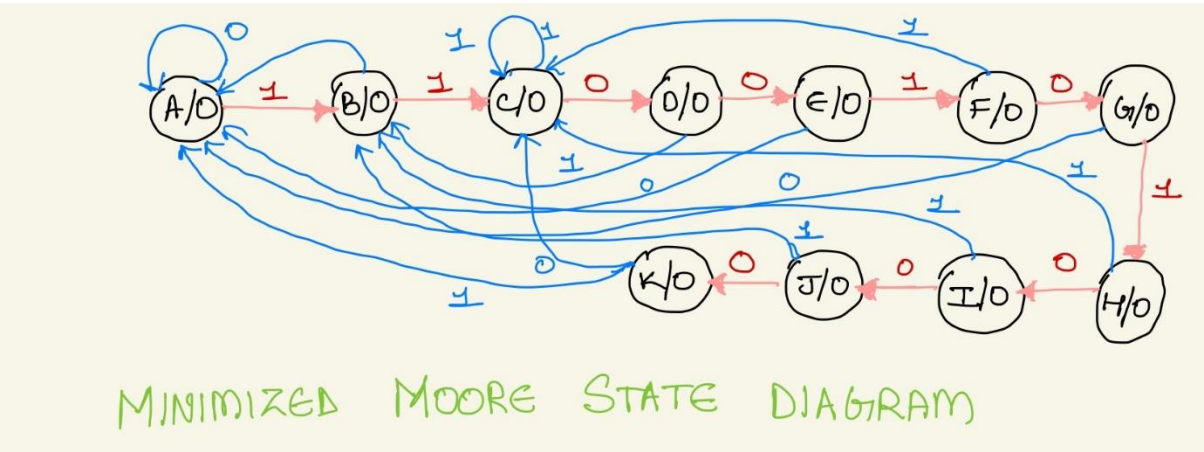
A,J L

MINIMISING TABLE

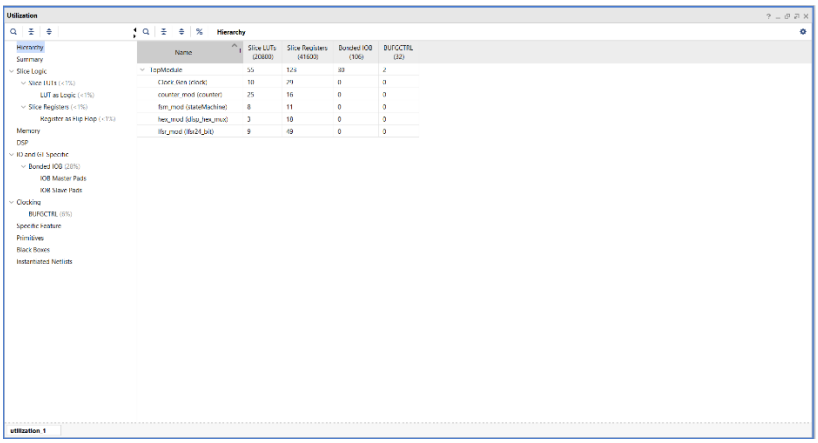
FINITE STATE DIAGRAM



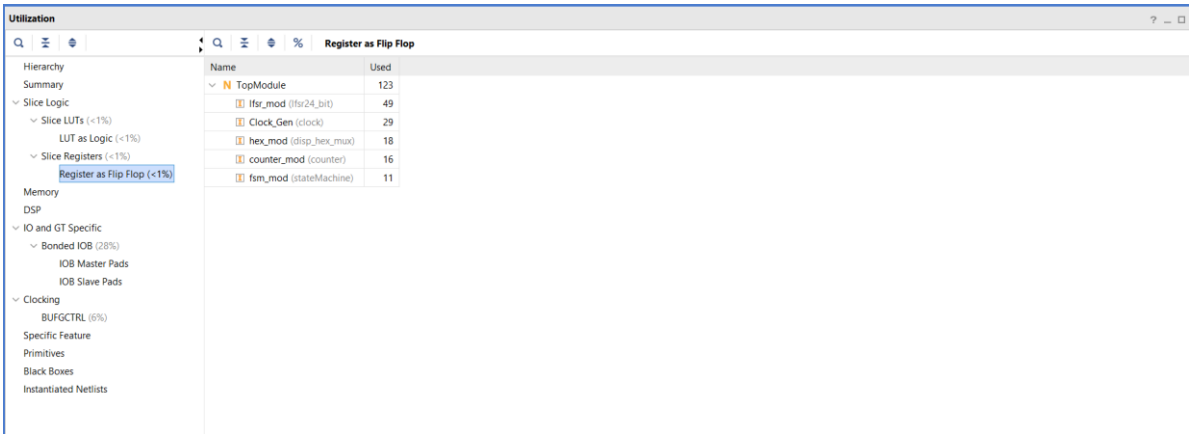
MINIMIZED STATE DIAGRAM:



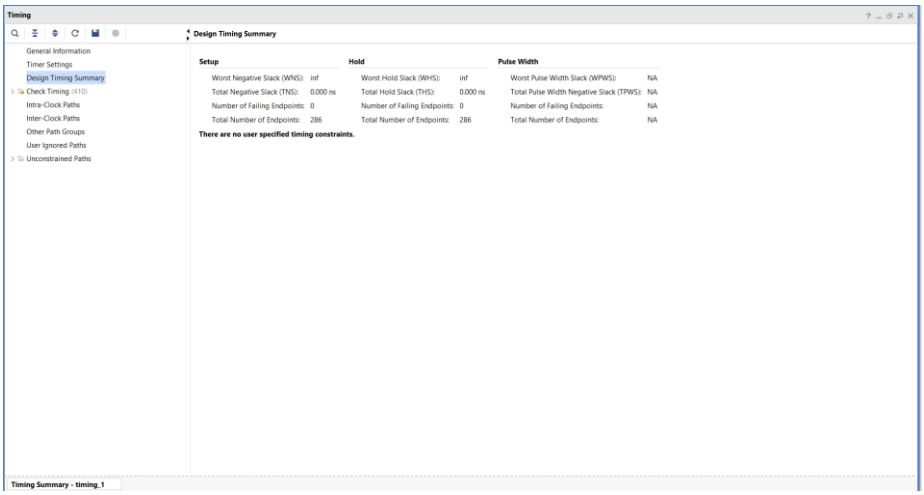
UTILIZATION REPORT:



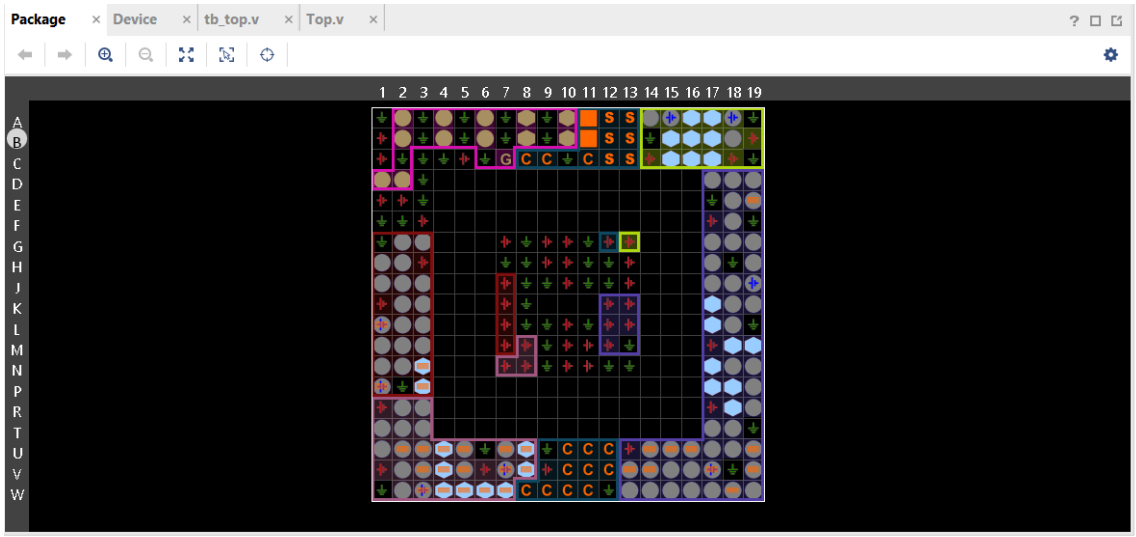
UTILIZATION REPORT ACCORDING TO REGISTER FLIP FLOPS:



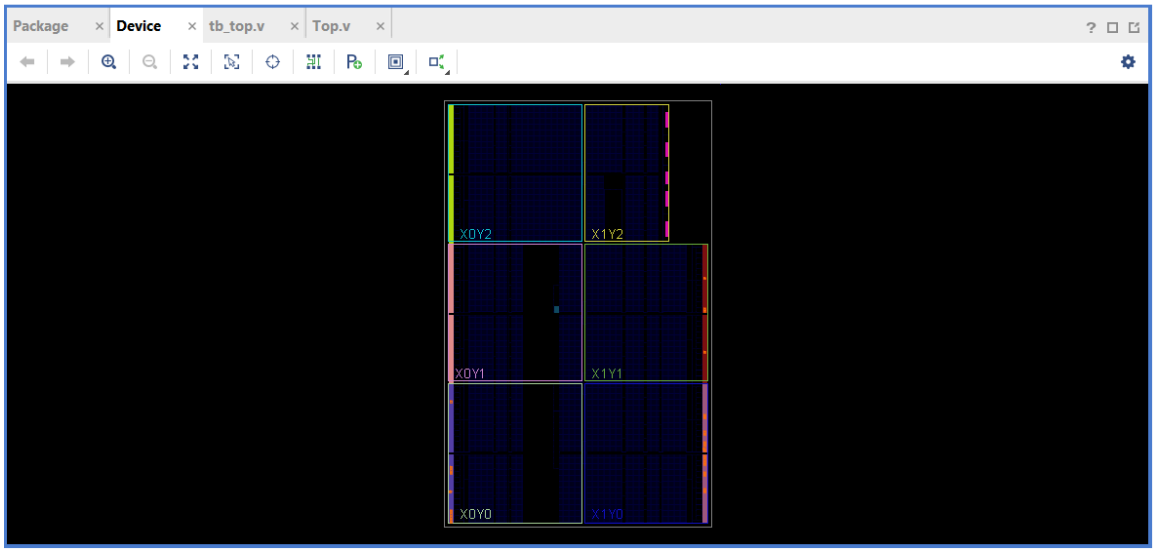
TIMING REPORT:



PACKAGE:



DEVICE:



CODES:

1)CLOCK.V

clock.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/new/clock.v

```
// Create Date: 27.11.2021 08:50:39
// Design Name:
// Module Name: clock
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
`timescale 1ns / 1ps
module clock(clk_in,clk_out
 );
input clk_in; // input clock on FPGA
output reg clk_out; // output clock after dividing the input clock by divisor

    parameter DIV = 32'd50000;
    reg [27:0] counter = 32'd0;

    always @(posedge clk_in)
      begin
          counter <= counter + 1;
          if(counter > DIV)
            counter <= 32'b0;
          clk_out <= (counter < DIV/2) ? 1'b1: 1'b0;
      end




endmodule
```

2)LFSR24_BIT.V

lfsr24_bit.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/new/lfsr24_bit.v

```

module lfsr24_bit(
    input wire clk,reset,
    output [23:0] lfsr_out,
    output max_tick_reg
    );

    reg [23:0] lfsr_reg;
    reg [23:0] lfsr_next;
    reg lfsr_tap;
    reg[23:0] count=24'b000000000000000000000000;
    reg tick_var=1'b0;

    always @(posedge clk)
    if(reset)
    lfsr_reg <=24'b100111111100100000000000;
    else
    lfsr_reg <= lfsr_next;

    always @(posedge clk)
    if(count== 24'b111111111111111111111111)   //2*N-1 cycles
      begin
      tick_var <= 1'b1;
      count <= 24'b000000000000000000000000;
      end
    else
      begin
      tick_var <=1'b0;
      count <= count + 1'b1;
      end

    always @*
    begin
    lfsr_tap =lfsr_reg[23]^lfsr_reg[22]^lfsr_reg[21]^lfsr_reg[16];
    lfsr_next = {lfsr_reg[22:0],lfsr_tap };
    end

    assign lfsr_out =lfsr_reg;
    assign max_tick_reg =tick_var;
  endmodule

```

3) COUNTER.V

stop_watch_if.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/imports/Provided stopwatch code/stop_watch_if.v

```
// Listing 4.18
module counter
   (
    input wire clk,
    input wire go, clr,
    output wire [3:0] d3,d2, d1, d0
   );

   // declaration
   localparam  DVSR = 10000000;
   reg [23:0] ms_reg;
   wire [23:0] ms_next;
   reg [3:0] d3_reg,d2_reg, d1_reg, d0_reg;
   reg [3:0] d3_next,d2_next, d1_next, d0_next;
   wire ms_tick;

   // body
   // register
   always @(posedge clk)
   begin
      ms_reg <= ms_next;
      d3_reg <= d3_next;
      d2_reg <= d2_next;
      d1_reg <= d1_next;
      d0_reg <= d0_next;
   end

   // next-state logic
   // 0.1 sec tick generator: mod-5000000
   assign ms_next = (clr || (ms_reg==DVSR && go)) ? 4'b0 :
                    (go) ? ms_reg + 1 :
                           ms_reg;
   assign ms_tick = (ms_reg==DVSR) ? 1'b1 : 1'b0;
   // 3-digit bcd counter
   always @*
   begin
      // default: keep the previous value
      d0_next = d0_reg;
      d1_next = d1_reg;
      d2_next = d2_reg;
      d3_next = d3_reg;
```

stop_watch_if.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/imports/Provided stopwatch code/stop_watch_if.v

```
        // default: keep the previous value
      d0_next = d0_reg;
      d1_next = d1_reg;
      d2_next = d2_reg;
      d3_next = d3_reg;
      if (clk)
        begin
          d0_next = 4'b0;
          d1_next = 4'b0;
          d2_next = 4'b0;
          d3_next = 4'b0;
        end
      else if (qv)
        if (d0_reg != 9)
          d0_next = d0_reg + 1;
        else                    // reach XX9
          begin
            d0_next = 4'b0;
            if (d1_reg != 9)
              d1_next = d1_reg + 1;
            else        // reach X99
              begin
                d1_next = 4'b0;
                if (d2_reg != 9)
                  d2_next = d2_reg + 1;
                else // reach 999
                  begin
                    d2_next = 4'b0;
                    if(d3_reg != 9)
                        d3_next = d3_reg +1;
                      else
                        d3_next = 4'b0;
                  end


              end
          end
    end

    // output logic
    assign d0 = d0_reg;
```

stop_watch_if.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/imports/Provided stopwatch code/stop_watch_if.v

```
      if (clk)
        begin
          d0_next = 4'b0;
          d1_next = 4'b0;
          d2_next = 4'b0;
          d3_next = 4'b0;
        end
      else if (qv)
        if (d0_reg != 9)
          d0_next = d0_reg + 1;
        else                    // reach XX9
          begin
            d0_next = 4'b0;
            if (d1_reg != 9)
              d1_next = d1_reg + 1;
            else        // reach X99
              begin
                d1_next = 4'b0;
                if (d2_reg != 9)
                  d2_next = d2_reg + 1;
                else // reach 999
                  begin
                    d2_next = 4'b0;
                    if(d3_reg != 9)
                        d3_next = d3_reg +1;
                      else
                        d3_next = 4'b0;
                  end


              end
          end
    end

    // output logic
    assign d0 = d0_reg;
    assign d1 = d1_reg;
    assign d2 = d2_reg;
    assign d3 = d3_reg;

endmodule
```

4)DISP_HEX.V

disp_hex.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/new/disp_hex.v

```
`timescale 1ns / 1ps
module disp_hex_mux
(
  input wire clk, reset,
  input wire [3:0] hex3, hex2, hex1, hex0, // hex digits
  input wire [3:0] dp_in, // 4 decimal points
  output reg [3:0] an, // enable 1-out-of-4 asserted low
  output reg [7:0] sseg // led segments
);
// constant declaration
// refreshing rate around 800 Hz (50 MHz/2^16)
localparam N = 18;
// internal signal declaration
reg [N-1:0] q_reg;
wire [N-1:0] q_next;
reg [3:0] hex_in;
reg dp;
// N-bit counter
// register
always @(posedge clk)
  if (reset)
    q_reg <= 0;
  else
    q_reg <= q_next;
// next-state logic
assign q_next = q_reg + 1;
// 2 MSBs of counter to control 4-to-1 multiplexing
// and to generate active-low enable signal
always @*
  case (q_reg[N-1:N-2])
    2'b00:
      begin
        an = 4'b1110;
        hex_in = hex0;
        dp = dp_in[0];
      end
    2'b01:
      begin
        an = 4'b1101;
        hex_in = hex1;
        dp = dp_in[1];
```

disp_hex.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/new/disp_hex.v

```
      2'b01:
        begin
          an = 4'b1101;
          hex_in = hex1;
          dp = dp_in[1];
        end
      2'b10:
        begin
          an = 4'b1011;
          hex_in = hex2;
          dp = dp_in[2];
        end
      2'b11:
        begin
          an = 4'b0111;
          hex_in = hex3;
          dp = dp_in[3];
        end
      default:
        begin
          an = 4'b0111;
          hex_in = hex3;
          dp = dp_in[3];
        end
  endcase
// hex to seven-segment led display

always @*
begin
  case(hex_in)
   4'h0: seg[6:0] = 7'b0000001;
   4'h1: seg[6:0] = 7'b1001111;
 4'h2: seg [6:0] = 7'b0010010;
 4'h3: seg [6:0] = 7'b0000110;
 4'h4: seg[6:0] = 7'b1001100;
 4'h5: seg[6:0] = 7'b0100100;
 4'h6: seg [6:0] = 7'b0100000;
 4'h7: seg[6:0] = 7'b0001111;
 4'h8: seg [6: 0] = 7'b0000000 ;
 4'h9: seg [6:0] = 7'b0000100;
 4'ha: seg [6:0] = 7'b0001000;
```

disp_hex.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/new/disp_hex.v

```
        end
      2'b11:
        begin
          an = 4'b0111;
          hex_in = hex3;
          dp = dp_in[3];
        end
      default:
        begin
          an = 4'b0111;
          hex_in = hex3;
          dp = dp_in[3];
        end
  endcase
// hex to seven-segment led display

always @*
begin
  case(hex_in)
   4'h0: seg[6:0] = 7'b0000001;
   4'h1: seg[6:0] = 7'b1001111;
 4'h2: seg [6:0] = 7'b0010010;
 4'h3: seg [6:0] = 7'b0000110;
 4'h4: seg[6:0] = 7'b1001100;
 4'h5: seg[6:0] = 7'b0100100;
 4'h6: seg [6:0] = 7'b0100000;
 4'h7: seg[6:0] = 7'b0001111;
 4'h8: seg [6: 0] = 7'b0000000 ;
 4'h9: seg [6:0] = 7'b0000100;
 4'ha: seg [6:0] = 7'b0001000;
 4'hb: seg [6:0] = 7'b1100000;
 4'hc: seg [6:0] = 7'b0110001;
 4'hd: seg [6:0] = 7'b1000010;
 4'he: seg [6:0] = 7'b0110000;
default:
        seg [6:0] = 7'b0111000;
endcase
   seg[7] = dp;
end
endmodule
```

5) STATE MACHINE

stateMachine.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/new/stateMachine.v

```
`timescale 1ns / 1ps
`timescale 1ns / 1ps
module stateMachine
(
input wire clk, reset,
input wire seq,
output reg tick
);
// symbolic state declaration
localparam A = 4'b0000,
B = 4'b0001,
C = 4'b0010,
D = 4'b0011,
E = 4'b0100,
F = 4'b0101,
G = 4'b0110,
H = 4'b0111,
I = 4'b1000,
J = 4'b1001,
K = 4'b1010;

reg [3:0] state_reg, state_next;

always @(posedge clk, posedge reset)
   begin
    if (reset)
    state_reg <= A;
    else
    state_reg <= state_next;
end
// next-state logic and output logic
always @*
   begin
    state_next = state_reg;
// default state: the same
    tick = 1'b0; // default output: 0
   case (state_reg)
    A:
     if (seq)
     state_next = B;
     else
```

stateMachine.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/new/stateMachine.v

```
  case (state_reg)
    A:
     if (seq)
     state_next = B;
     else
     state_next = A;
    B:
     if (seq)
     state_next = C;
     else
     state_next = A;
    C:
     if (~seq)
     state_next = D;
     else
     state_next = C;
    D:
     if (~seq)
     state_next = E;
     else
     state_next = B;
    E:
     if (seq)
     state_next = F;
     else
     state_next = C;
     F:
     if (~seq)
     state_next = G;
     else
     state_next = A;
     G:
     if (seq)
     state_next = H;
     else
     state_next = C;
      H:
     if (~seq)
     state_next = I;
     else
     state_next = B;
```

stateMachine.v

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/sources_1/new/stateMachine.v

```
     state_next = G;
     else
     state_next = A;
     G:
     if (seq)
     state_next = H;
     else
     state_next = C;
      H:
     if (~seq)
     state_next = I;
     else
     state_next = B;
     I:
     if (~seq)
     state_next = J;
     else
     state_next = B;
     J:
     if (~seq)
     state_next = K;
     else
     state_next = B;
     K:
     if (~seq)
     begin
     state_next = A;
     tick=1'b1;
     end
     else
     state_next = B;

  default:
     begin
     state_next = A;
     tick=1'b1;
end
endcase
end
endmodule
```

CONSTRAINTS (XDC FILE)

constraints_assig_2.xdc

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/constrs_1/new/constraints_assig_2.xdc

```
set_property IOSTANDARD LVCMOS33 [get_ports {aan[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {aan[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {aan[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {aan[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[13]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[12]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lfsr24_bout[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
set_property PACKAGE_PIN W4 [get_ports {aan[3]}]
set_property PACKAGE_PIN V4 [get_ports {aan[2]}]
set_property PACKAGE_PIN U4 [get_ports {aan[1]}]
set_property PACKAGE_PIN U2 [get_ports {aan[0]}]
set_property PACKAGE_PIN W3 [get_ports {lfsr24_bout[13]}]
set_property PACKAGE_PIN P3 [get_ports {lfsr24_bout[12]}]
set_property PACKAGE_PIN U3 [get_ports {lfsr24_bout[11]}]
set_property PACKAGE_PIN W3 [get_ports {lfsr24_bout[10]}]
set_property PACKAGE_PIN V3 [get_ports {lfsr24_bout[9]}]
set_property PACKAGE_PIN V13 [get_ports {lfsr24_bout[8]}]
set_property PACKAGE_PIN V14 [get_ports {lfsr24_bout[7]}]
set_property PACKAGE_PIN U14 [get_ports {lfsr24_bout[6]}]
set_property PACKAGE_PIN U15 [get_ports {lfsr24_bout[5]}]
set_property PACKAGE_PIN W10 [get_ports {lfsr24_bout[4]}]
set_property PACKAGE_PIN V19 [get_ports {lfsr24_bout[3]}]
```

constraints_assig_2.xdc

C:/Users/DELL/project_Assignment_2/project_Assignment_2.srcs/constrs_1/new/constraints_assig_2.xdc

```
set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
set_property PACKAGE_PIN W4 [get_ports {aan[3]}]
set_property PACKAGE_PIN V4 [get_ports {aan[2]}]
set_property PACKAGE_PIN U4 [get_ports {aan[1]}]
set_property PACKAGE_PIN U2 [get_ports {aan[0]}]
set_property PACKAGE_PIN N3 [get_ports {lfsr24_bout[13]}]
set_property PACKAGE_PIN P3 [get_ports {lfsr24_bout[12]}]
set_property PACKAGE_PIN U3 [get_ports {lfsr24_bout[11]}]
set_property PACKAGE_PIN W3 [get_ports {lfsr24_bout[10]}]
set_property PACKAGE_PIN V3 [get_ports {lfsr24_bout[9]}]
set_property PACKAGE_PIN V13 [get_ports {lfsr24_bout[8]}]
set_property PACKAGE_PIN V14 [get_ports {lfsr24_bout[7]}]
set_property PACKAGE_PIN U14 [get_ports {lfsr24_bout[6]}]
set_property PACKAGE_PIN U15 [get_ports {lfsr24_bout[5]}]
set_property PACKAGE_PIN W18 [get_ports {lfsr24_bout[4]}]
set_property PACKAGE_PIN V19 [get_ports {lfsr24_bout[3]}]
set_property PACKAGE_PIN U19 [get_ports {lfsr24_bout[2]}]
set_property PACKAGE_PIN E19 [get_ports {lfsr24_bout[1]}]
set_property PACKAGE_PIN U16 [get_ports {lfsr24_bout[0]}]
set_property PACKAGE_PIN V7 [get_ports {seg[7]}]
set_property PACKAGE_PIN W7 [get_ports {seg[6]}]
set_property PACKAGE_PIN W6 [get_ports {seg[5]}]
set_property PACKAGE_PIN U8 [get_ports {seg[4]}]
set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
set_property PACKAGE_PIN U5 [get_ports {seg[2]}]
set_property PACKAGE_PIN V5 [get_ports {seg[1]}]
set_property PACKAGE_PIN U7 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk_in]
set_property PACKAGE_PIN W5 [get_ports clk_in]
set_property PACKAGE_PIN V17 [get_ports reset]
set_property IOSTANDARD LVCMOS33 [get_ports reset]

set_property IOSTANDARD LVCMOS33 [get_ports found]
set_property IOSTANDARD LVCMOS33 [get_ports max_tick]
set_property PACKAGE_PIN P1 [get_ports max_tick]
set_property PACKAGE_PIN L1 [get_ports found]
```

I/O Ports

Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Te
All ports (30)														
aan (4)	OUT					☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
aan[3]	OUT				W4	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
aan[2]	OUT				V4	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
aan[1]	OUT				U4	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
aan[0]	OUT				U2	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout (14)	OUT					☑	(Multiple)	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				N3	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				P3	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U3	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				W3	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				V3	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				V13	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				V14	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U14	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U15	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				W18	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				V19	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U19	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				E19	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U16	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg (8)	OUT					☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[7]	OUT				V7	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[6]	OUT				W7	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[5]	OUT				W6	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[4]	OUT				U8	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[3]	OUT				V8	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[2]	OUT				U5	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[1]	OUT				V5	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[0]	OUT				U7	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50

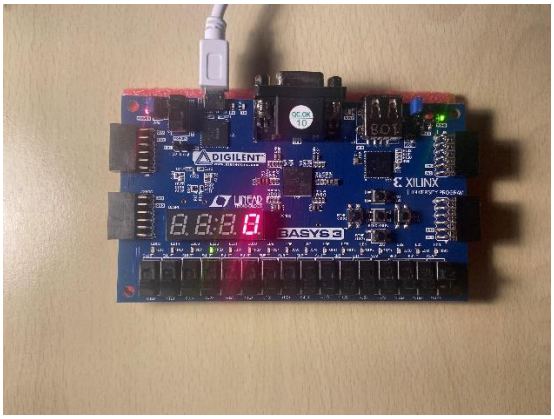
I/O Ports

Name	Direction	Board Part Pin	Board Part Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Te
aan[0]	OUT				U2	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout (14)	OUT					☑	(Multiple)	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				N3	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				P3	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U3	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				W3	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				V3	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				V13	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				V14	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U14	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U15	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				W18	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				V19	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U19	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				E19	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
lfsr24_bout[...	OUT				U16	☑	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg (8)	OUT					☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[7]	OUT				V7	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[6]	OUT				W7	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[5]	OUT				W6	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[4]	OUT				U8	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[3]	OUT				V8	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[2]	OUT				U5	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[1]	OUT				V5	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
seg[0]	OUT				U7	☑	34	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
Scalar ports (4)														
clk_in	IN				W5	☑	34	LVCMOS33*	3.300				NONE	NONE
found	OUT				L1	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
max_tick	OUT				P1	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
reset	IN				V17	☑	14	LVCMOS33*	3.300				NONE	NONE

DEMO:

Now when we generate the bitstream and the reset is high so the output on the board is as follows:

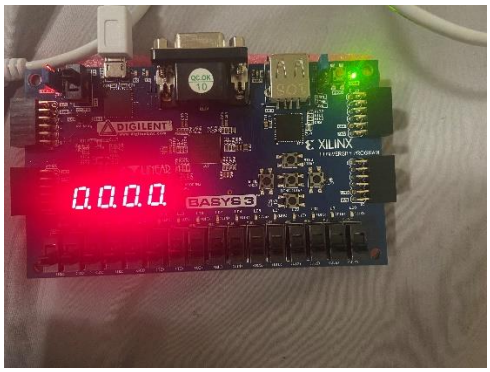
My seed value is 100111111001000000000000.I define the LFSR for 14 bits and output the seed value now comes from the left 14 bits. Soo now the 1 led lights up.

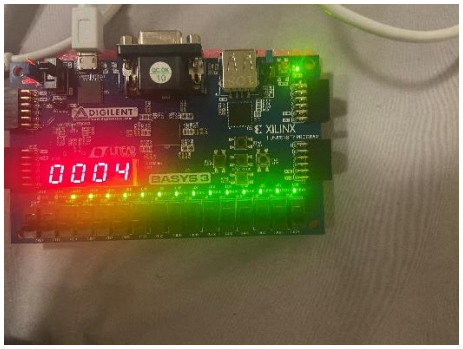


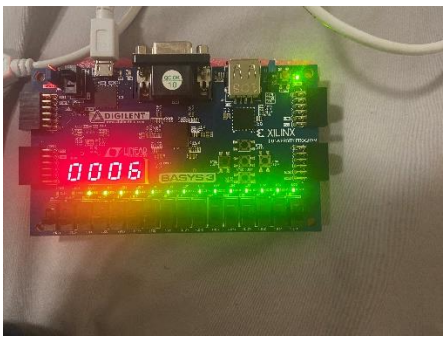
When we generate the full bitstream. The counter goes on and lfsr bits are also shown.The video link is attached for DEMO.

As soon as we run the bit file the led's start to blink with the speed initialised in the clock.v file.

The counter starts from 0 and goes on till 9999 on the BASYS 3 board . My LFSR cycle goes on till 2^24-1.While the seven segment decoder goes on the speed of the clock is too high soo the blinking of the LED's cant be seen.







The demo video is in the link below.

In the DEMO video as the max_tick becomes 1 the counter increases.

LINK: https://drive.google.com/file/d/195bqEcnWrSvQPf0e_AKP8e8wh1fiKTaG/view?usp=sharing

The clock in this project is being handled by the CLK_IN in the xdc file.

The output of the counter is the number of times the same pattern is getting generated.

-SHRUTI KATHURIA