PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013) 100-ft Ring Road,Bengaluru-560 085,Karnataka,In

Electronics and Communication Engineering

DIGITAL SYSTEM DESIGN:(UE21EC342AB1)

MINI PROJECT: Pseudo Random Binary Sequence (PRBS) Generator using Linear feedback shift register (LFSR).

DONE BY:

GURUPRASAD REDDY : - PES1UG21EC107

G S S S DHANVI : - PES1UG21EC104

HEMANT CHANDRASEKHAR WADE : - PES1UG21EC112

GOKUL MVSS : - PES1UG21EC105

Course Instructor : -Dr.Rashmi Seethur

CONTENT

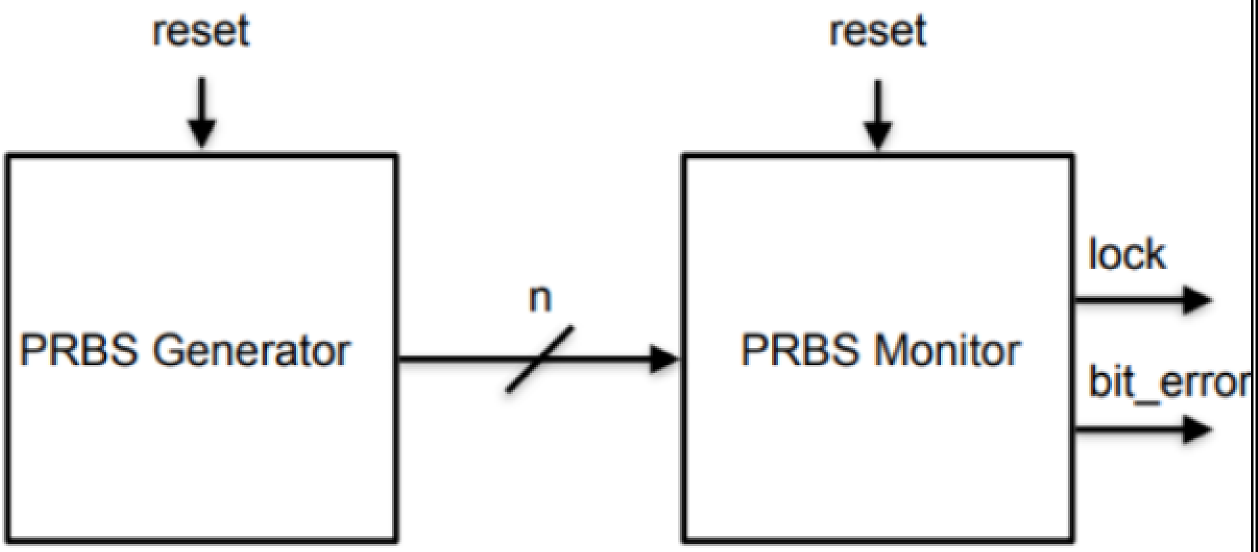
1.PROBLEM STATEMENT AND BLACKBOX STRUCTURE.
2. Introduction
3. Flowchart
4. Design Code
    a) PRBS.V
    b) PRBS_Gen.v
    c) PRBS_Monitor.v
5. Test bench
6. Simulation Result
7. Code coverage
8.Synthesis schematic
9.CONSTRAINS FILE
10.SYNTHESIS REPORT
    a) Power Report
    b) Area Report
    c) Timing Report (with and without constraints)
11.Netlist
12.RUN.tcl
13.Conclusion
14.Bibliography

1) PROBLEM STATEMENT AND BLACKBOX STRUCTURE:

Our team has been assigned with , PROJECT ID – 6 : Pseudo Random Binary Sequence (PRBS) Generator using Linear feedback shift register(LFSR).

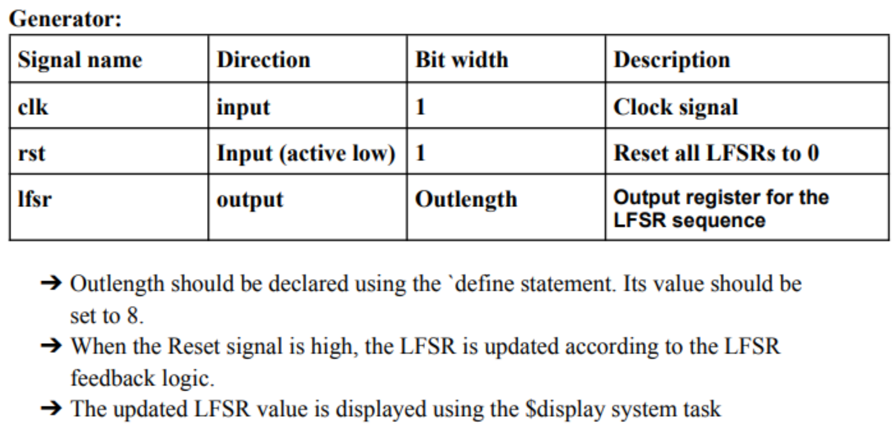
The Specifications allocated to us are shown below:

- Black Box Structure:



- Other specifications:

    - PRBS Monitor generates LFSR given the pattern generated by PRBS Generator.



Generator:

Signal name	Direction	Bit width	Description
clk	input	1	Clock signal
rst	Input (active low)	1	Reset all LFSRs to 0
lfsr	output	Outlength	Output register for the LFSR sequence

- ➔ Outlength should be declared using the `define statement. Its value should be set to 8.
- ➔ When the Reset signal is high, the LFSR is updated according to the LFSR feedback logic.
- ➔ The updated LFSR value is displayed using the $display system task

Design Ports for Monitor:

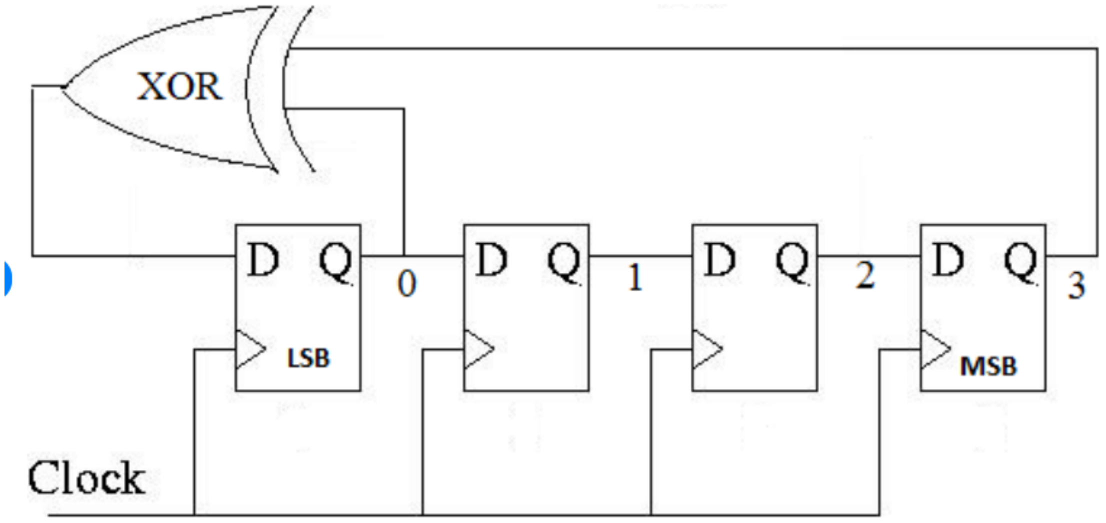
Signal name	Direction	Bit width	Description
clk	input	1	Clock signal
rst	input	1	Reset signal
lfsr	output	8	input representing the PRBS sequence from the external source.
lock	output	1	indicate if the monitored PRBS matches the received PRBS.
bit_error	output	1	indicate if a bit error has been detected.

- o PRBS Monitor generates LFSR given the pattern generated by PRBS Generator.
- o Lock is set to 1 when Gen and Monitor have the same pattern value.
- o Bit_error is set to high when Gen and Monitor sync for 3 consecutive cycles.

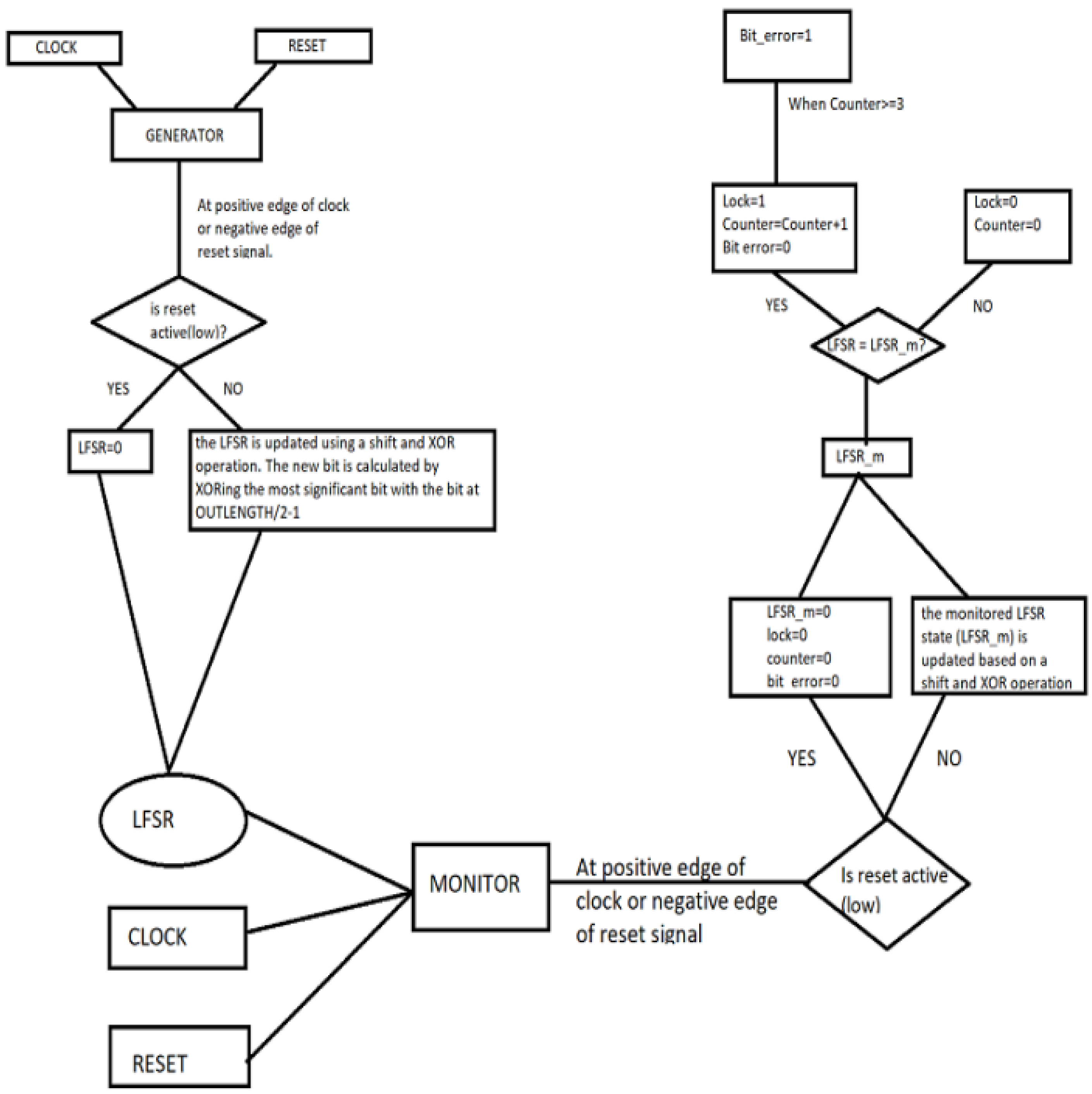
2) INTRODUCTION:

A pseudorandom binary sequence (PRBS), is a binary sequence that, while generated with a deterministic algorithm, is difficult to predict and exhibits statistical behavior similar to a truly random sequence, or In simple words, sequence of binary values that appears random but is generated by a deterministic algorithm. The most common example is the maximum length sequence generated by a (maximal) linear feedback shift register (LFSR).

LFSRs (linear feedback shift registers) provide a simple means for generating non sequential lists of numbers quickly on microcontrollers. Generating the pseudo-random numbers only requires a right-shift operation and an XOR operation. A basic LFSR does not produce very good random numbers. A better sequence of numbers can be improved by picking a larger LFSR and using the lower bits for the random number. We use a subset of the LFSR for a random number to increase the permutations of the numbers and improve the random properties of the LFSR output. Shifting the LFSR more than once before getting a random number also improves its statistical properties. Shifting the LFSR by a factor of its period will reduce the total period length by that factor.



FLOW CHART


CLOCK
RESET
GENERATOR
At positive edge of clock or negative edge of reset signal.
is reset active(low)?
YES
NO
LFSR=0
the LFSR is updated using a shift and XOR operation. The new bit is calculated by XORing the most significant bit with the bit at OUTLENGTH/2-1
LFSR
CLOCK
RESET
MONITOR
At positive edge of clock or negative edge of reset signal
Is reset active (low)
YES
NO
LFSR_m=0
lock=0
counter=0
bit_error=0
the monitored LFSR state (LFSR_m) is updated based on a shift and XOR operation
LFSR_m
LFSR = LFSR_m?
YES
NO
Lock=1
Counter=Counter+1
Bit error=0
Lock=0
Counter=0
When Counter>=3
Bit_error=1

3) Design Code:

- The design code consists of 3 parts:
  PRBS.V, PRBS_GEN.V and PRBS_MONITOR.V

a) PRBS.V

Code:

```
`define OUTLENGTH 8


module PRBS (input Clock,Reset,output lock, bit_error);

wire [`OUTLENGTH-1:0] LFSR_w;

PRBS_Gen Gen(
 Clock_G,
 Reset,
 LFSR_w
 );

PRBS_Monitor Mon(
 Clock_G,
 Reset,
 LFSR_w,
 lock,
 bit_error
 );
Endmodule
```

b) PRBS_Gen.V

Code:

```
`define OUTLENGTH 8

module PRBS_Gen (
 input Clock,
 input Reset,
 output reg [`OUTLENGTH-1:0] LFSR
);


always @(posedge Clock or negedge Reset) begin
 if(~Reset) begin
  LFSR <= 0;
 end else begin
  LFSR <= { LFSR[`OUTLENGTH-1:0], ~(LFSR[`OUTLENGTH-1] ^
LFSR[`OUTLENGTH/2-1]) } ;
  $display("Monitor %h",LFSR);
 end
end
endmodule
```

c) PRBS_Monitor.v:

CODE:

```
`define OUTLENGTH 8
module PRBS_Monitor (
 input Clock,
 input Reset,
 input [`OUTLENGTH-1:0] LFSR,
 output reg lock,
 output reg bit_error
);

reg [`OUTLENGTH-1:0] LFSR_m;
reg [`OUTLENGTH-1:0] counter;
always @(posedge Clock or negedge Reset) begin
 if(~Reset) begin
  LFSR_m <= 0;
  lock <= 0;
  counter <= 0;
  bit_error <= 0;
 end else begin
  // shift <= LFSR;
  LFSR_m <= { LFSR[`OUTLENGTH-1:0], ~(LFSR_m[`OUTLENGTH/2-1]
^ LFSR_m[`OUTLENGTH/2-2] ^ LFSR_m[`OUTLENGTH/2-3]) } ;
  $display("Monitor %h",LFSR_m);
  if (LFSR_m == LFSR) begin
   lock <=1;
   counter <= counter + 1;
   bit_error <= 0;
  end
 else begin
   lock <=0;
   counter <= 0;
   if (counter >= 3)
    bit_error <=1;
  end


end
 end
endmodule
```

4) TESTBENCH

PRBS_tb.v:

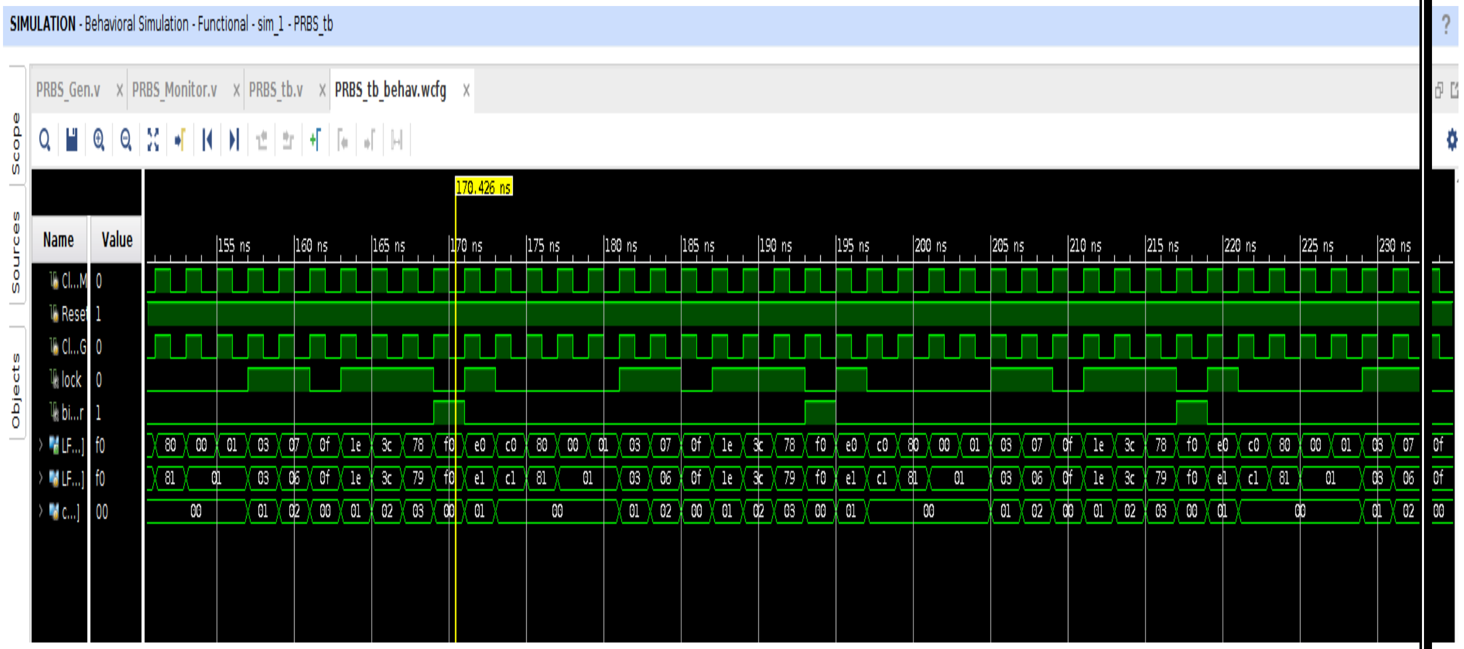
CODE:

```
`define OUTLENGTH 8
 module PRBS_tb ();
 reg Clock_G;
 reg Clock_M;
 reg Reset;
 wire lock;
 wire bit_error;
initial begin
// $monitor("Reset %b, Output %b",Reset,LFSR);
    Clock_G = 0;
    Clock_M = 0;
    Reset = 0;
    #10
    Reset = 1;
    #1000
 Reset=0;
    #1000000
    $finish;
  end
  always #1 Clock_M = ~Clock_M;
  always #1 Clock_G = ~Clock_G;
  PRBS DUT(
   Clock_G,
    Clock_M,
   Reset,
   lock,
   bit_error
```

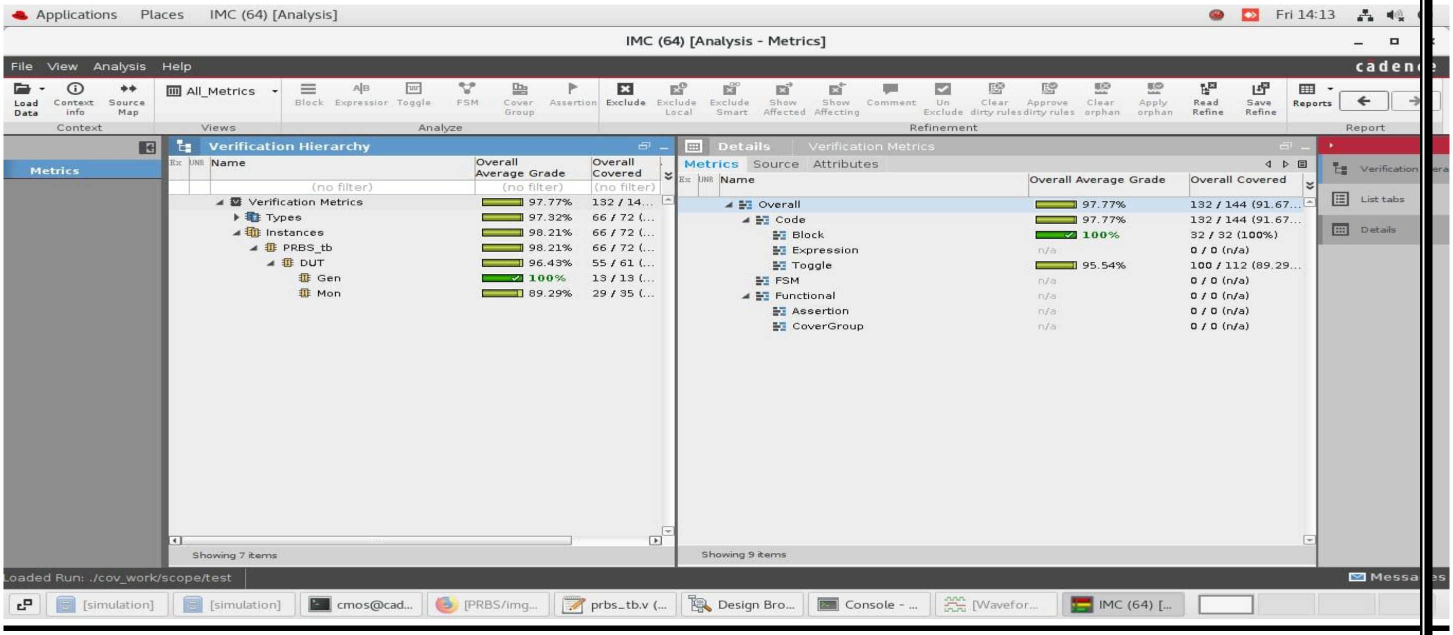
);endmodule

OUTPUTS:

6) SIMULATION:



7) Code Coverage:



Name	Overall Average Grade	Overall Covered
(no filter)	(no filter)	(no filter)
Verification Metrics	97.77%	132 / 14...
Types	97.32%	66 / 72 (...
Instances	98.21%	66 / 72 (...
PRBS_tb	98.21%	66 / 72 (...
DUT	96.43%	55 / 61 (...
Gen	100%	13 / 13 (...
Mon	89.29%	29 / 35 (...

Name	Overall Average Grade	Overall Covered
Overall	97.77%	132 / 144 (91.67...
Code	97.77%	132 / 144 (91.67...
Block	100%	32 / 32 (100%)
Expression	n/a	0 / 0 (n/a)
Toggle	95.54%	100 / 112 (89.29...
FSM	n/a	0 / 0 (n/a)
Functional	n/a	0 / 0 (n/a)
Assertion	n/a	0 / 0 (n/a)
CoverGroup	n/a	0 / 0 (n/a)

SYNTHESIS:

Tool Used: GENUS (By Cadence)

Commands:

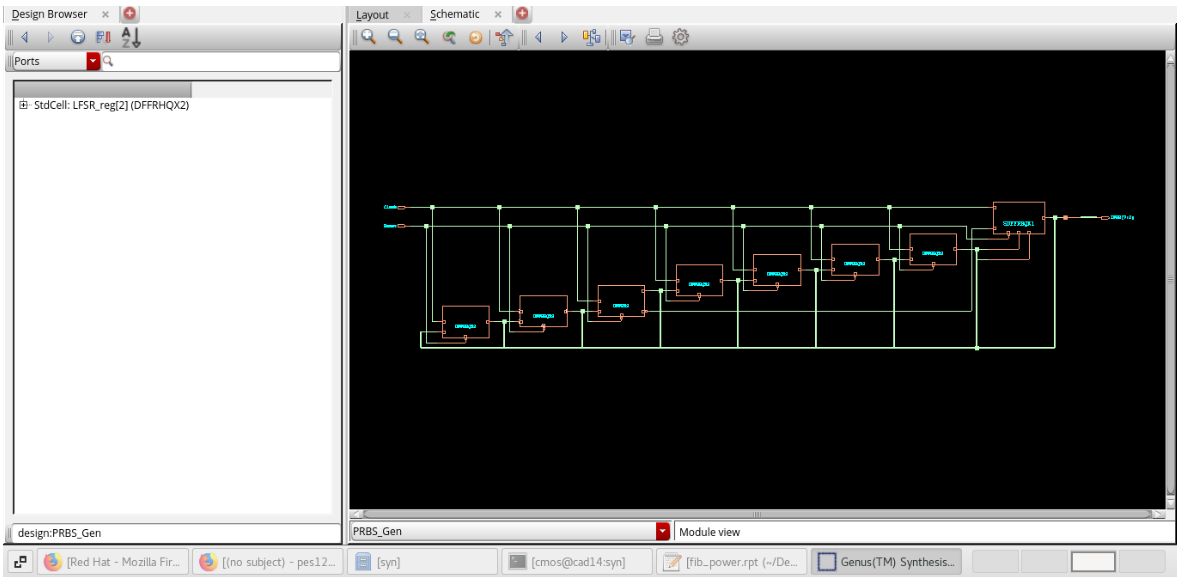
• csh
• source /home/install/cshrc
• genus -gui // This command invokes the genus tool
• read_libs
• read_hdl
• elaborate
• syn_generic
• syn_map
• syn_opt -incremental
• gui_show
• write_netlist > netlist.v
• write_sdc > filename_tool.sdc
• report_power > name_power.rpt
• report_area > name_area.rpt
• read_sdc

• report_timing > name_timing.rpt

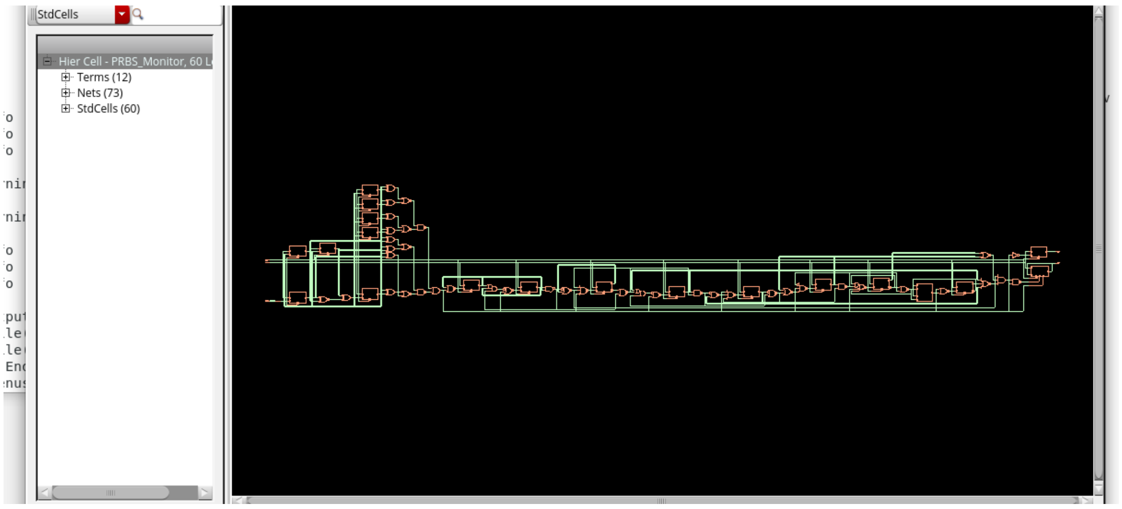
// All these commands are also directly available in the run.tcl file.
So, another method for direct simulation is
• source run.tcl
// This commands directly runs all the commands listed above in a single file

8) SYNTHESIS SCHEMATIC:

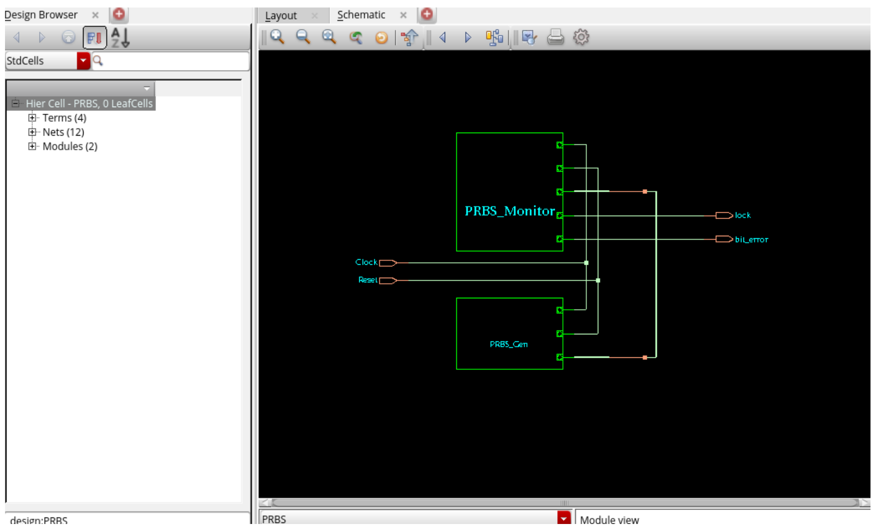
prbs_gen



prbs_monitor



Prbs

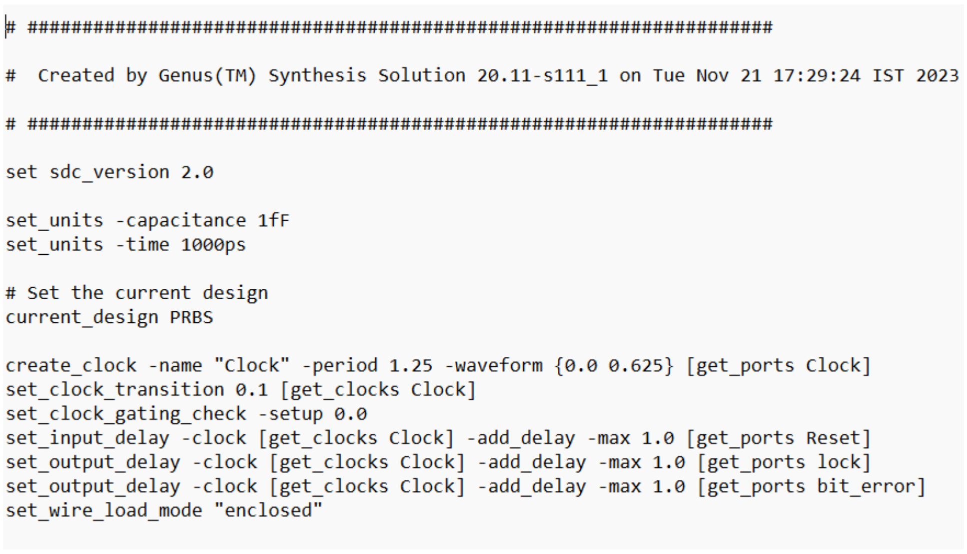


9)CONSTRAINS FILE:

input constrain file

```
create_clock -name Clock -period 1.25 -waveform {0 0.625} [get_ports "Clock"]
set_clock_transition -rise 0.1 [get_clocks "Clock"]
set_clock_transition -fall 0.1 [get_clocks "Clock"]
set_input_delay -max 0.06 -clock Clock [all_inputs]
set_output_delay -max 0.06 -clock Clock [all_outputs]
```

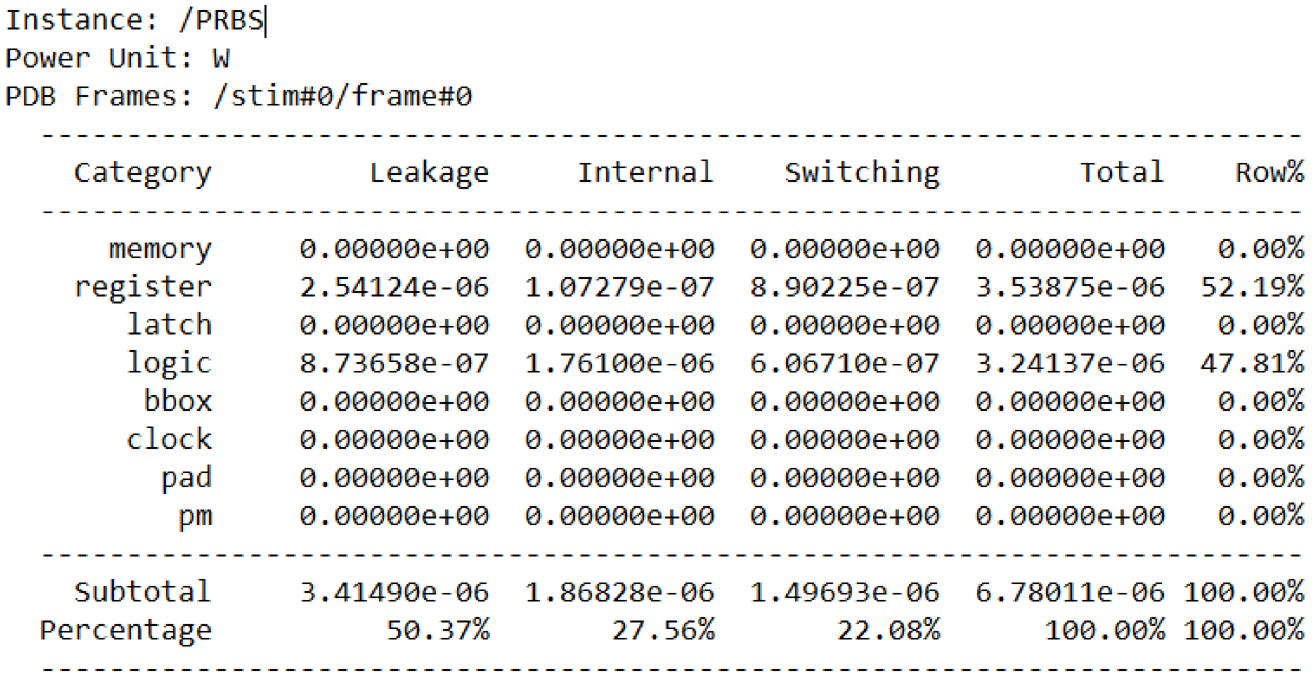
output constrain:



```
# ####################################################################

#  Created by Genus(TM) Synthesis Solution 20.11-s111_1 on Tue Nov 21 17:29:24 IST 2023

# ####################################################################

set sdc_version 2.0

set_units -capacitance 1fF
set_units -time 1000ps

# Set the current design
current_design PRBS

create_clock -name "Clock" -period 1.25 -waveform {0.0 0.625} [get_ports Clock]
set_clock_transition 0.1 [get_clocks Clock]
set_clock_gating_check -setup 0.0
set_input_delay -clock [get_clocks Clock] -add_delay -max 1.0 [get_ports Reset]
set_output_delay -clock [get_clocks Clock] -add_delay -max 1.0 [get_ports lock]
set_output_delay -clock [get_clocks Clock] -add_delay -max 1.0 [get_ports bit_error]
set_wire_load_mode "enclosed"
```

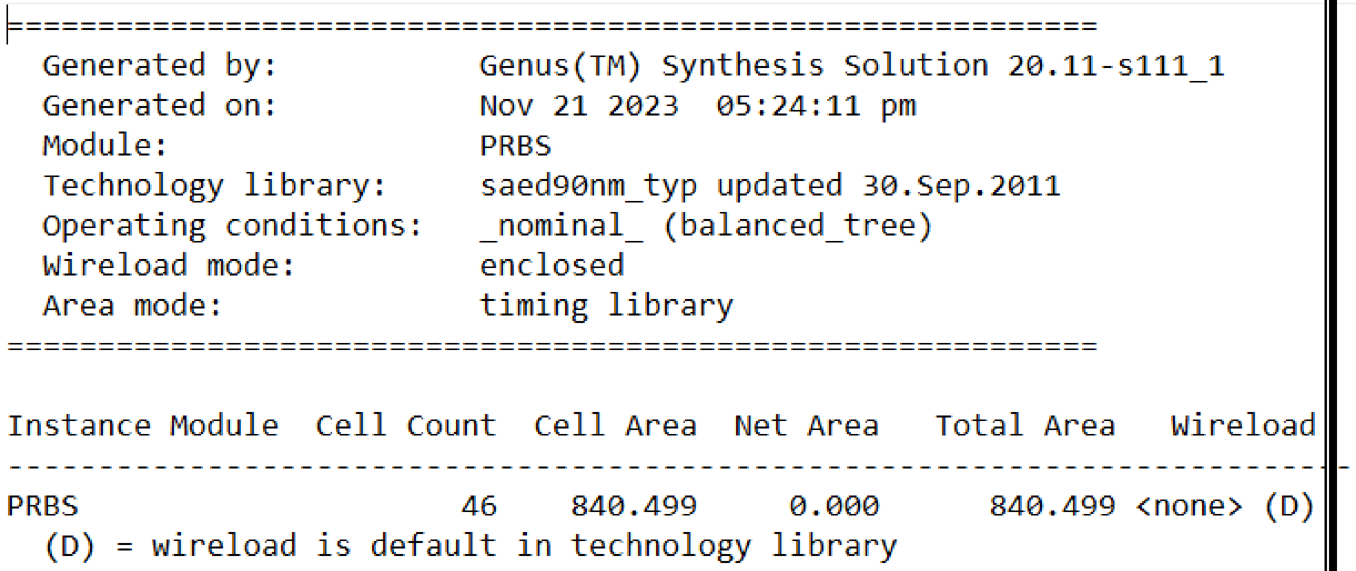
10) SYNTHESIS REPORT:

A) POWER REPORT :



```
Instance: /PRBS
Power Unit: W
PDB Frames: /stim#0/frame#0
  -------------------------------------------------------------------------
    Category         Leakage     Internal    Switching        Total    Row%
  -------------------------------------------------------------------------
      memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
    register     2.54124e-06  1.07279e-07  8.90225e-07  3.53875e-06  52.19%
       latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
       logic     8.73658e-07  1.76100e-06  6.06710e-07  3.24137e-06  47.81%
        bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
       clock     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
         pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
          pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  -------------------------------------------------------------------------
    Subtotal     3.41490e-06  1.86828e-06  1.49693e-06  6.78011e-06 100.00%
  Percentage          50.37%       27.56%       22.08%      100.00% 100.00%
  -------------------------------------------------------------------------
```

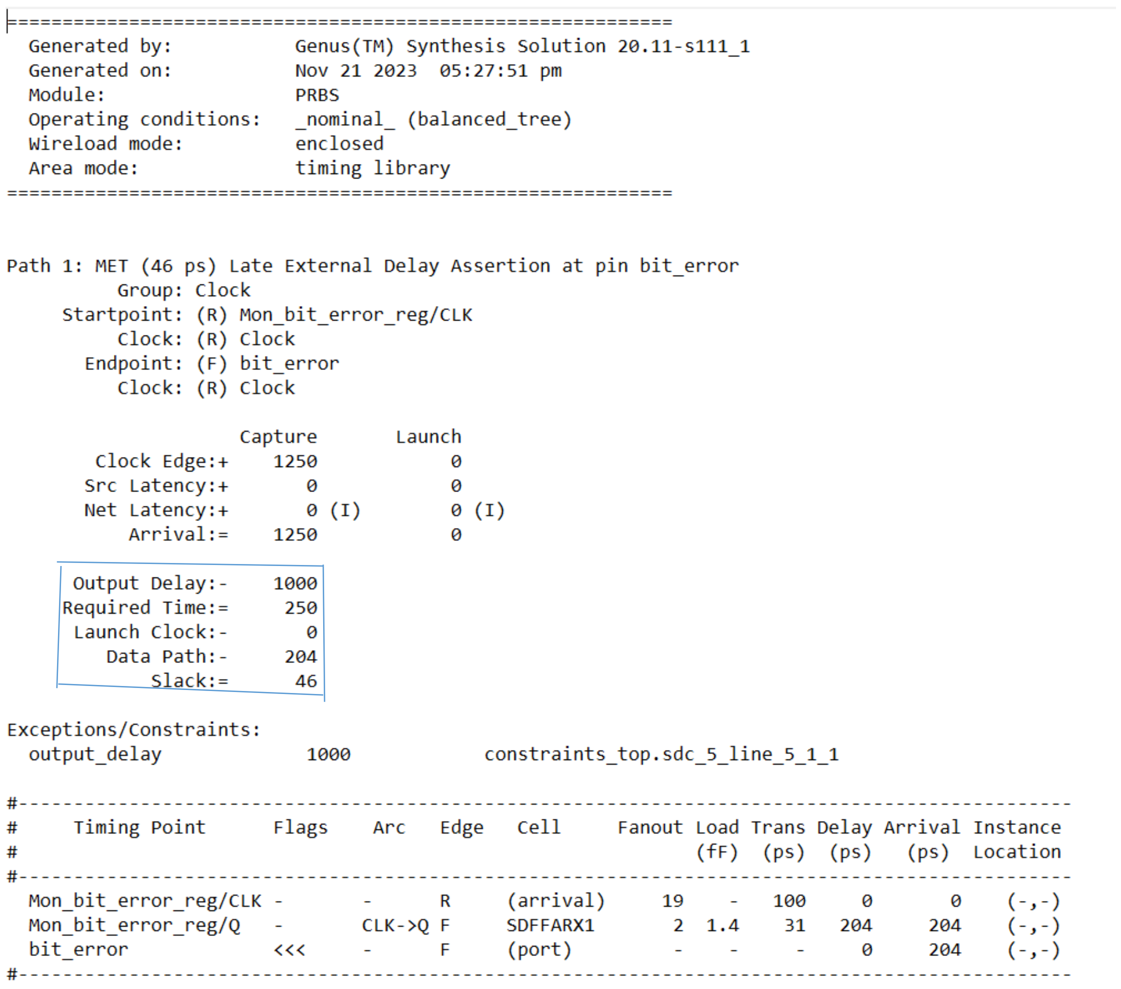
B)AREA REPORT:



```
============================================================
  Generated by:          Genus(TM) Synthesis Solution 20.11-s111_1
  Generated on:          Nov 21 2023  05:24:11 pm
  Module:                PRBS
  Technology library:    saed90nm_typ updated 30.Sep.2011
  Operating conditions:  _nominal_ (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

Instance Module  Cell Count  Cell Area  Net Area   Total Area   Wireload
-------------------------------------------------------------------------
PRBS                     46    840.499     0.000      840.499 <none> (D)
  (D) = wireload is default in technology library
```

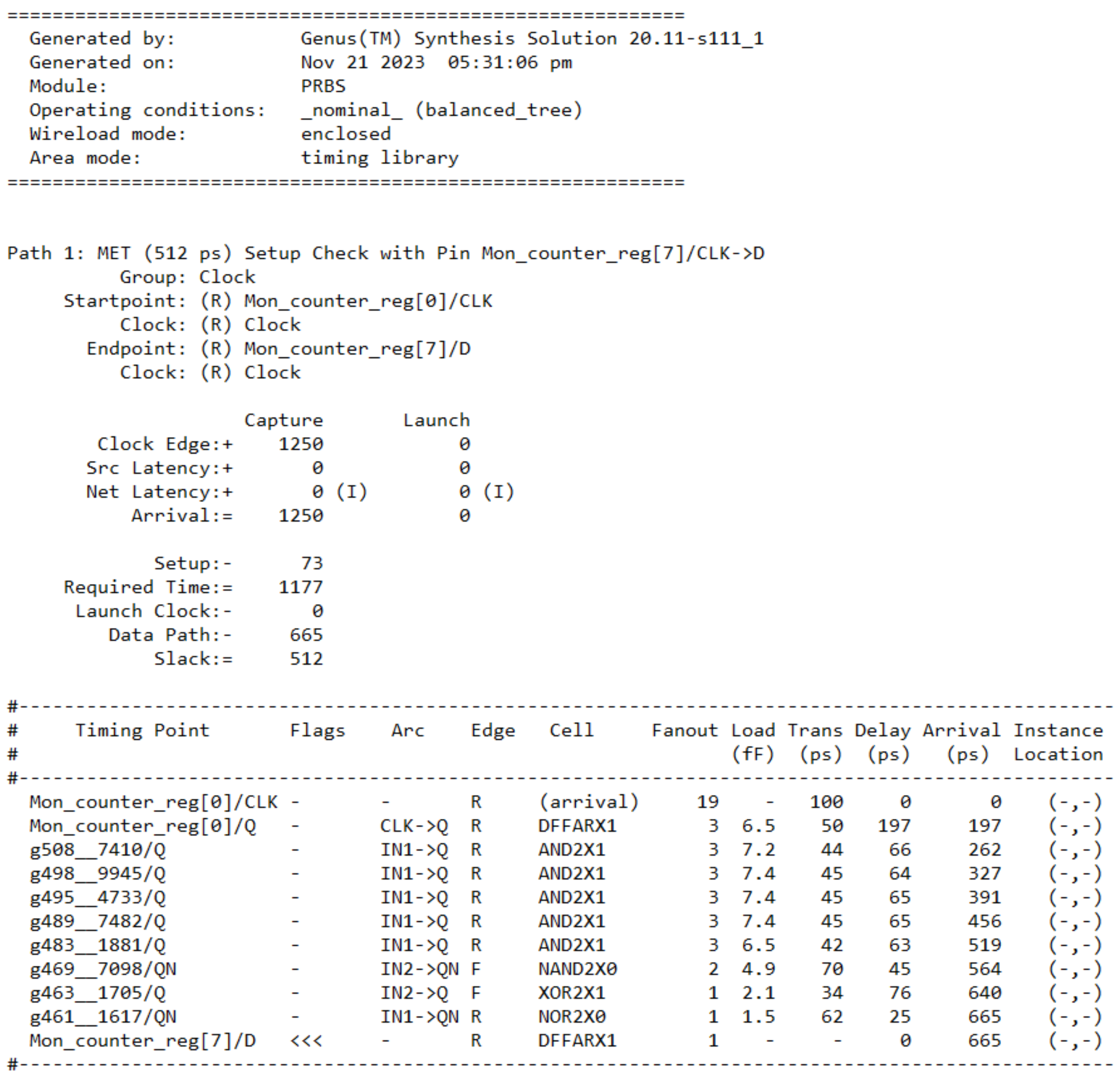
C) TIMIMG REPORT:

a) WITHOUT CONSTRAINS:



```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 20.11-s111_1
  Generated on:           Nov 21 2023  05:27:51 pm
  Module:                 PRBS
  Operating conditions:   _nominal_ (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (46 ps) Late External Delay Assertion at pin bit_error
          Group: Clock
     Startpoint: (R) Mon_bit_error_reg/CLK
          Clock: (R) Clock
       Endpoint: (F) bit_error
          Clock: (R) Clock

                     Capture       Launch
        Clock Edge:+    1250            0
       Src Latency:+       0            0
       Net Latency:+       0 (I)        0 (I)
           Arrival:=    1250            0

      Output Delay:-    1000
     Required Time:=     250
      Launch Clock:-       0
         Data Path:-     204
             Slack:=      46

Exceptions/Constraints:
  output_delay             1000            constraints_top.sdc_5_line_5_1_1

#------------------------------------------------------------------------------------------------
#     Timing Point       Flags     Arc    Edge    Cell       Fanout  Load Trans Delay Arrival Instance
#                                                                    (fF)  (ps)  (ps)   (ps)  Location
#------------------------------------------------------------------------------------------------
  Mon_bit_error_reg/CLK  -         -       R     (arrival)      19     -   100     0       0    (-,-)
  Mon_bit_error_reg/Q    -         CLK->Q  F     SDFFARX1        2   1.4    31   204     204    (-,-)
  bit_error              <<<       -       F     (port)          -     -     -     0     204    (-,-)
#------------------------------------------------------------------------------------------------
```

b) WITH CONSTRAINS



```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 20.11-s111_1
  Generated on:           Nov 21 2023  05:31:06 pm
  Module:                 PRBS
  Operating conditions:   _nominal_ (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (512 ps) Setup Check with Pin Mon_counter_reg[7]/CLK->D
          Group: Clock
     Startpoint: (R) Mon_counter_reg[0]/CLK
          Clock: (R) Clock
       Endpoint: (R) Mon_counter_reg[7]/D
          Clock: (R) Clock

                    Capture       Launch
        Clock Edge:+    1250            0
       Src Latency:+       0            0
       Net Latency:+       0 (I)        0 (I)
           Arrival:=    1250            0

             Setup:-      73
     Required Time:=    1177
      Launch Clock:-       0
         Data Path:-     665
             Slack:=     512

#--------------------------------------------------------------------------------------------------
#      Timing Point       Flags    Arc     Edge   Cell        Fanout Load Trans Delay Arrival Instance
#                                                                    (fF)  (ps)  (ps)   (ps)  Location
#--------------------------------------------------------------------------------------------------
  Mon_counter_reg[0]/CLK -        -        R     (arrival)      19    -   100     0       0    (-,-)
  Mon_counter_reg[0]/Q   -        CLK->Q   R     DFFARX1         3  6.5    50   197     197    (-,-)
  g508__7410/Q           -        IN1->Q   R     AND2X1          3  7.2    44    66     262    (-,-)
  g498__9945/Q           -        IN1->Q   R     AND2X1          3  7.4    45    64     327    (-,-)
  g495__4733/Q           -        IN1->Q   R     AND2X1          3  7.4    45    65     391    (-,-)
  g489__7482/Q           -        IN1->Q   R     AND2X1          3  7.4    45    65     456    (-,-)
  g483__1881/Q           -        IN1->Q   R     AND2X1          3  6.5    42    63     519    (-,-)
  g469__7098/QN          -        IN2->QN  F     NAND2X0         2  4.9    70    45     564    (-,-)
  g463__1705/Q           -        IN2->Q   F     XOR2X1          1  2.1    34    76     640    (-,-)
  g461__1617/QN          -        IN1->QN  R     NOR2X0          1  1.5    62    25     665    (-,-)
  Mon_counter_reg[7]/D   <<<      -        R     DFFARX1         1    -     -     0     665    (-,-)
#--------------------------------------------------------------------------------------------------
```

11) NETLIST:

```
// Generated by Cadence Genus(TM) Synthesis Solution 20.11-s111_1
// Generated on: Nov 21 2023 17:28:44 IST (Nov 21 2023 11:58:44 UTC)

// Verification Directory fv/PRBS

module PRBS(Clock, Reset, lock, bit_error);
  input Clock, Reset;
  output lock, bit_error;
  wire Clock, Reset;
  wire lock, bit_error;
  wire [7:0] Mon_counter;
  wire [7:0] LFSR_w;
  wire [7:0] Mon_LFSR_m;
  wire UNCONNECTED, UNCONNECTED0, UNCONNECTED1,
UNCONNECTED2,
       UNCONNECTED3, UNCONNECTED4, UNCONNECTED5,
UNCONNECTED6;
  wire UNCONNECTED7, UNCONNECTED8, UNCONNECTED9,
UNCONNECTED10,
       UNCONNECTED11, UNCONNECTED12, UNCONNECTED13,
UNCONNECTED14;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11, n_12, n_13, n_14, n_15;
  wire n_16, n_17, n_18, n_19, n_20, n_21, n_22, n_23;
  wire n_24, n_25, n_26, n_27, n_28, n_29;
  SDFFARX1 Mon_bit_error_reg(.RSTB (Reset), .CLK (Clock), .D
       (bit_error), .SI (n_28), .SE (n_25), .Q (bit_error), .QN
       (UNCONNECTED));
  DFFARX1 \Mon_counter_reg[5] (.RSTB (Reset), .CLK (Clock), .D (n_23),
       .Q (Mon_counter[5]), .QN (UNCONNECTED0));
  DFFARX1 \Mon_counter_reg[4] (.RSTB (Reset), .CLK (Clock), .D (n_27),
       .Q (Mon_counter[4]), .QN (UNCONNECTED1));
  DFFARX1 \Mon_counter_reg[2] (.RSTB (Reset), .CLK (Clock), .D (n_26),
       .Q (Mon_counter[2]), .QN (UNCONNECTED2));
  DFFARX1 \Mon_counter_reg[3] (.RSTB (Reset), .CLK (Clock), .D (n_29),
       .Q (Mon_counter[3]), .QN (UNCONNECTED3));
  DFFARX1 \Mon_counter_reg[1] (.RSTB (Reset), .CLK (Clock), .D (n_20),
       .Q (Mon_counter[1]), .QN (UNCONNECTED4));
  DFFARX1 \Mon_counter_reg[6] (.RSTB (Reset), .CLK (Clock), .D (n_22),
       .Q (Mon_counter[6]), .QN (UNCONNECTED5));
  DFFARX1 \Mon_counter_reg[7] (.RSTB (Reset), .CLK (Clock), .D (n_19),
       .Q (Mon_counter[7]), .QN (UNCONNECTED6));
```

```
DFFARX1 \Mon_counter_reg[0] (.RSTB (Reset), .CLK (Clock), .D (n_21),
    .Q (Mon_counter[0]), .QN (UNCONNECTED7));
NOR3X0 g479__2398(.IN1 (n_8), .IN2 (n_5), .IN3 (n_28), .QN (n_29));
NOR3X0 g480__5107(.IN1 (n_11), .IN2 (n_9), .IN3 (n_28), .QN (n_27));
NOR3X0 g482__6260(.IN1 (n_7), .IN2 (n_24), .IN3 (n_28), .QN (n_26));
NAND4X0 g474__4319(.IN1 (n_24), .IN2 (n_28), .IN3 (n_3), .IN4 (n_1),
    .QN (n_25));
NOR3X0 g468__8428(.IN1 (n_15), .IN2 (n_12), .IN3 (n_28), .QN (n_23));
NOR2X0 g464__5526(.IN1 (n_16), .IN2 (n_28), .QN (n_22));
NOR2X0 g476__6783(.IN1 (n_28), .IN2 (Mon_counter[0]), .QN (n_21));
NOR3X0 g481__3680(.IN1 (n_28), .IN2 (n_4), .IN3 (n_0), .QN (n_20));
NOR2X0 g461__1617(.IN1 (n_17), .IN2 (n_28), .QN (n_19));
DFFARX1 Mon_lock_reg(.RSTB (Reset), .CLK (Clock), .D (n_18), .Q
    (lock), .QN (UNCONNECTED8));
SDFFARX1 \Gen_LFSR_reg[0] (.RSTB (Reset), .CLK (Clock), .D (n_10),
    .SI (LFSR_w[3]), .SE (LFSR_w[7]), .Q (LFSR_w[0]), .QN
    (UNCONNECTED9));
INVX0 g485(.INP (n_18), .ZN (n_28));
XNOR2X1 g486__2802(.IN1 (Mon_LFSR_m[0]), .IN2 (LFSR_w[0]), .Q (n_18));
DFFARX1 \Gen_LFSR_reg[7] (.RSTB (Reset), .CLK (Clock), .D
    (LFSR_w[6]), .Q (LFSR_w[7]), .QN (UNCONNECTED10));
XOR2X1 g463__1705(.IN1 (Mon_counter[7]), .IN2 (n_14), .Q (n_17));
DFFARX1 \Mon_LFSR_m_reg[0] (.RSTB (Reset), .CLK (Clock), .D (n_13),
    .Q (Mon_LFSR_m[0]), .QN (UNCONNECTED11));
OAI21X1 g466__5122(.IN1 (Mon_counter[6]), .IN2 (n_15), .IN3 (n_14),
    .QN (n_16));
DFFARX1 \Gen_LFSR_reg[6] (.RSTB (Reset), .CLK (Clock), .D
    (LFSR_w[5]), .Q (LFSR_w[6]), .QN (UNCONNECTED12));
XNOR3X1 g491__8246(.IN1 (LFSR_w[1]), .IN2 (LFSR_w[2]), .IN3
    (LFSR_w[3]), .Q (n_13));
NAND2X0 g469__7098(.IN1 (Mon_counter[6]), .IN2 (n_15), .QN (n_14));
DFFARX1 \Gen_LFSR_reg[5] (.RSTB (Reset), .CLK (Clock), .D
    (LFSR_w[4]), .Q (LFSR_w[5]), .QN (UNCONNECTED13));
DFFARX1 \Gen_LFSR_reg[4] (.RSTB (Reset), .CLK (Clock), .D
    (LFSR_w[3]), .Q (LFSR_w[4]), .QN (UNCONNECTED14));
NOR2X0 g477__6131(.IN1 (n_11), .IN2 (Mon_counter[5]), .QN (n_12));
AND2X1 g483__1881(.IN1 (n_11), .IN2 (Mon_counter[5]), .Q (n_15));
NOR2X0 g488__5115(.IN1 (n_8), .IN2 (Mon_counter[4]), .QN (n_9));
AND2X1 g489__7482(.IN1 (n_8), .IN2 (Mon_counter[4]), .Q (n_11));
DFFARX1 \Gen_LFSR_reg[3] (.RSTB (Reset), .CLK (Clock), .D (n_6), .Q
    (LFSR_w[3]), .QN (n_10));
AND2X1 g495__4733(.IN1 (n_7), .IN2 (Mon_counter[3]), .Q (n_8));
NOR2X0 g494__6161(.IN1 (n_7), .IN2 (Mon_counter[3]), .QN (n_5));
DFFARX1 \Gen_LFSR_reg[2] (.RSTB (Reset), .CLK (Clock), .D (n_2), .Q
    (n_6), .QN (LFSR_w[2]));
```

```
  NOR2X0 g497__9315(.IN1 (n_4), .IN2 (Mon_counter[2]), .QN (n_24));
  AND2X1 g498__9945(.IN1 (n_4), .IN2 (Mon_counter[2]), .Q (n_7));
  NOR3X0 g503__2883(.IN1 (Mon_counter[3]), .IN2 (Mon_counter[4]), .IN3
       (Mon_counter[7]), .QN (n_3));
  NOR2X0 g505__2346(.IN1 (Mon_counter[5]), .IN2 (Mon_counter[6]), .QN
       (n_1));
  DFFARX1 \Gen_LFSR_reg[1] (.RSTB (Reset), .CLK (Clock), .D
       (LFSR_w[0]), .Q (n_2), .QN (LFSR_w[1]));
  NOR2X0 g507__1666(.IN1 (Mon_counter[0]), .IN2 (Mon_counter[1]), .QN
       (n_0));
  AND2X1 g508__7410(.IN1 (Mon_counter[0]), .IN2 (Mon_counter[1]), .Q
       (n_4));
endmodule
```

12) RUN.tcl

```
set_db init_lib_search_path /home/install/FOUNDRY/digital/45nm/LIBS/lib/max/

set_db lef_library /home/install/FOUNDRY/digital/45nm/LIBS/lef/gsclib045.fixed.lef

set_db library  slow.lib



read_hdl {./prbs.v}


elaborate

read_sdc ./constraints_top.sdc


#set_attribute syn_generic_effort medium
#set_attribute syn_map_effort  medium
#set_attribute syn_opt_effort  medium

syn_generic
syn_map
syn_opt

write_hdl > fib1_netlist.v
write_sdc  > fib1_tool.sdc

gui_show

report timing > fib1_timing.rpt
report power > fib1_power.rpt

report area > fib1_cell.rpt
report gates > fib1_gates.rpt
```

13) CONCLUSION:

We have successfully implemented the given problem statement, Pseudo random binary sequence using Linear feedback shift register, and implemented simulation and synthesis with the tools available in Cadence

14) BIBLIOGRAPHY:

- https://www.analog.com/en/design-notes/random-number-generation-using-lfsr.html
- https://core.ac.uk/download/pdf/53186864.pdf