VLSI DESIGN II

Project Title: Binary multiplication with PN Sequences

Project Done Under: PROF.DR.D.MISRA

Project By

Shuchen Liu: sl639@njit.edu

Sanket Shankar Kulkarni:sk2339@njit.edu

Yue Xu: yx329@njit.edu

Table of Contents

Acknowledgement

We wish to express our deep sense of gratitude to our Professor Dr. D Misra, New Jersey Institute of Technology for duly steering the course and for his guidance and encouragement throughout the project.

We would like to extend our sincere thanks to our Professor for clarifying each doubt of ours regarding the subject no matter how small the issues were.

Sanket Shankar Kulkarni
sk2339@njit.edu
314-06-022

Shuchen Liu
sl639@njit.edu
314-09-094

Yue Xu
yx329@njit.edu
313-76-760

ABSTRACT

Multiplier is one of the most important arithmetic unit in Microprocessors and DSPs and also a major source of power dissipation. Reducing the power dissipation of multipliers is a key to satisfy the overall power budget of various digital circuits and systems. Pseudo noise (PN) sequences are sequences of maximum period generated by a linear recurrence. PN sequences have many useful properties and have been used in a wide range of applications. In this correspondence we show how a nth-order PN Sequence can be used to multiply two n/2-bit numbers to n-bit accuracy. The main part of circuit being linear feedback shift register(LFSR). The designs are done using Mentor Graphics tool, Model Sim, Mentor Leonardo, Synopsys Primetime.

Keywords: PN Sequences, LFSR, Feedback

Task Completed by Team Members

SR.NO	TASK	Name
1.	VHDL Coding of circuit 1. Shift Register 2. Linear Feedback Shift Register 3. Linear Feedback Shift Register(Part2) Combining two different LFSR 4. Binary Fraction 5. Merging of code for Complete circuit	 Sanket Kulkarni Sanket Kulkarni Shuchen Liu Shuchen Liu Yu Xue All
2.	Simulation of Circuit Using Model Sim	Yu Xue
3	Synthesis of Circuit Using Mentor Leonardo 1. Area Report 2. Delay Report 3. Critical Path	Shuchen Liu
4	Extraction of Circuit from Verilog file using Mentor IC Schematic	Shuchen Liu Yu Xue
5	Layout of circuits with successful LVS	Sanket Kulkarni
6	Timing Analysis using Primetime	Sanket Kulkarni
7	Execution and debugging	All

Table 1: Task Assignment

1.INTRODUCTION

1.1 PN Sequence

Pseudo-Noise (PN) sequences whose terms depend in a simple manner on their predecessors are of great importance for a variety of other applications. Such sequences are easily generated by recursive procedures and hence PN sequences have an advantageous feature from the computational viewpoint, and they tend to have useful structural properties. Due to only these structural properties, PN sequences have enormous applications like Direct Sequence Spread Spectrum (DSSS), Built-in Self-Test (BIST), Decryption–Encryption System (DES) error detection and many more.

In these applications the systems use the basic hardware named Linear Feedback Shift Register (LFSR) to generate Pseudo-Noise (PN) sequence . LFSR is made up of two parts. These parts are a shift register and a feedback function. The shift register is a chain sequence of n-bits of D – type of Flip-Flops (FFs). Each time a new bit is needed to load the first bit (D-FF1) of the chains of D – FFs. The all others of the bits in the shift register are shifted one bit to the right. The feedback function is simply the Exclusive-OR (EOR) operation logic of certain bits of the register. The list of those bits which are involved in EOR operation logic is called a feedback taps (C0 C1 C2 …. Ci …. Cn-1 Cn). The new left most bit's state (first bit of D – flip-flop, D-FF1) is computed as a function of the existing feedback taps of LFSR. The output of the feedback shift register is one bit at each clock, often the most significant bit a clock before. The period p of a shift register is the length of the output sequence before it starts repeating. 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. PRBS are used in telecommunication, encryption, simulation, correlation technique and time-of-flight spectroscopy.

$PN_5 = [11111000110111010100001001011 00],$
$PN_4 = [11110001101110101000010010110 01],$
$PN_3 = [11100011011101010000100101100 11],$
$PN_2 = [11000110111010100001001011001 11],$ and
$PN_1 = [10001101110101000010010110011 11]$ respectively.

Fig.1.1 PN sequence for 5 bit LFSR

1.2 Linear Feedback Shift Register

In computing, a linear-feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is exclusive-or (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value. The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well-chosen feedback function can produce a sequence of bits that appears random and has a very long cycle. Applications of LFSRs include generating pseudo-random numbers, pseudo-noise sequences, fast digital counters, and whitening sequences. Both hardware and software implementations of LFSRs are common. The mathematics of a cyclic redundancy check, used to provide a quick check against transmission errors, are closely related to those of an LFSR.

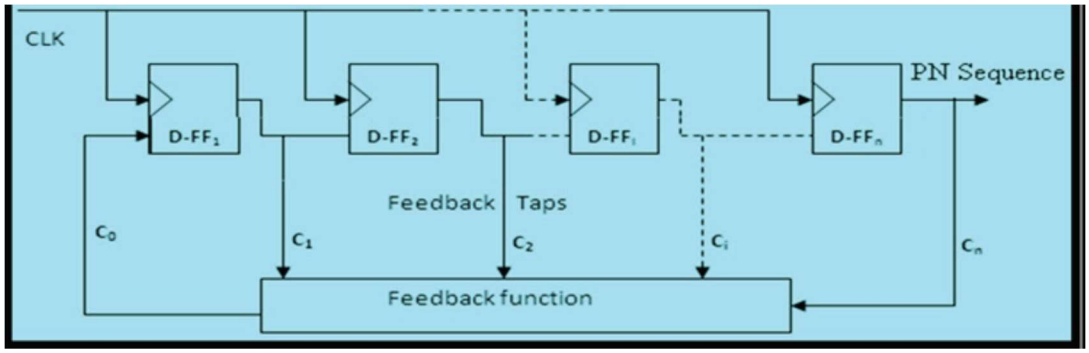


Fig.1.2 An n-bit LFSR structure

The bit positions that affect the next state are called the taps. In the diagram the taps are [16,14,13,11]. The rightmost bit of the LFSR is called the output bit. The taps are XOR'd sequentially with the output bit and then fed back into the leftmost bit. The sequence of bits in the rightmost position is called the output stream.

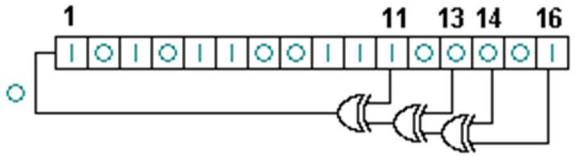


Fig.1.3 General representation

2.Properties of PN Sequence

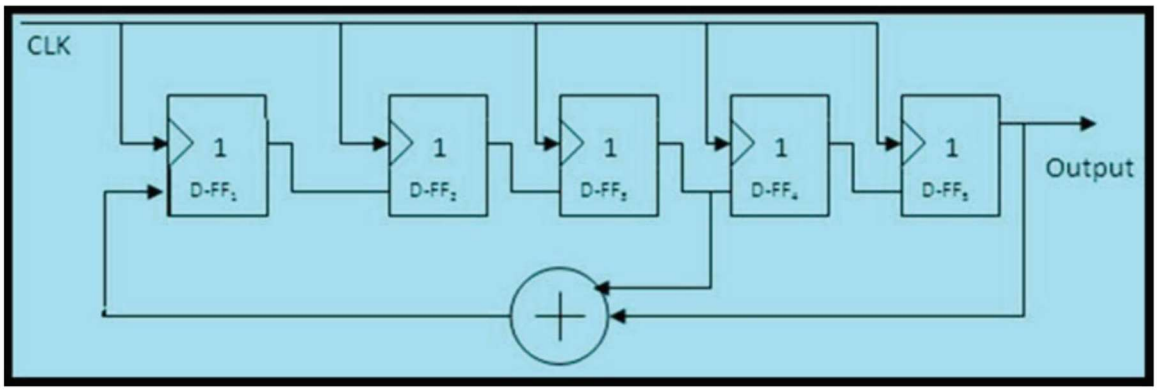


Fig.2.1 A 5 bit LFSR Circuit

The patterns produced by the LFSR shown in Figure 4, assuming that the pattern of 11111 was used as an initial loading and feedback taps are taken from 3rd and 5th bits of the FFs.

The generated PN sequences outputted from FF5, FF4, FF3, FF2, and FF1 are:

PN5 = [11111000110111010100000100101100],

PN4 = [11110001101110101000001001011001],

PN3 = [11100011011101010000100101100011],

PN2 = [11000110111010100001001011001111],

PN1 = [10001101110101000010010110011111].

Each PN sequences has the same period p = 31. Also, it is notable all the PN sequences are observing the same properties for example total number of 1s and total number of 0s, groups of 1s and 0s and like many other properties.

Property 1: P1

In every period (p = 2^n–1) of PN sequence generated by an n-bit LFSR, the sequence will contain the total number of 1s equal to 2^(n–1).

Property 2: P2

In every period (p = 2n–1) of PN sequence generated by an n-bit LFSR, the sequence will contain the total number of 0s equal to

2^(n-1)-1.

Clock	FF_1	FF_2	FF_3	FF_4	FF_5	Comment
1	1	1	1	1	1	Initial Loading
2	0	1	1	1	1	
3	0	0	1	1	1	
4	0	0	0	1	1	
5	1	0	0	0	1	
6	1	1	0	0	0	
7	0	1	1	0	0	
8	1	0	1	1	0	
9	1	1	0	1	1	
10	1	1	1	0	1	
11	0	1	1	1	0	
12	1	0	1	1	1	
13	0	1	0	1	1	
14	1	0	1	0	1	
15	0	1	0	1	0	
16	0	0	1	0	1	
17	0	0	0	1	0	
18	0	0	0	0	1	
19	1	0	0	0	0	
20	0	1	0	0	0	
21	0	0	1	0	0	
22	1	0	0	1	0	
23	0	1	0	0	1	
24	1	0	1	0	0	
25	1	1	0	1	0	
26	0	1	1	0	1	
27	0	0	1	1	0	
28	1	0	0	1	1	
29	1	1	0	0	1	
30	1	1	1	0	0	
31	1	1	1	1	0	
32	1	1	1	1	1	Starts repeating

Fig.2.2 output of 5 bit LFSR

Property 3: P3

In every period (p = 2^n-1) of PN sequence generated by an n-bit LFSR, the sequence has an occurrence of n number of 1s in succession.

Property 4: P4

In every period (p = 2^n–1) of PN sequence generated by an n-bit LFSR, the sequence does not have any occurrence of total number of (n) 0s in succession.

Property 5: P5

In every period (p = 2^n–1) of PN sequence generated by an n-bit LFSR, the sequence does not have any occurrence of total number of (n–1) 1s in succession.

Property 6: P6

In every period (p = 2^n–1) of PN sequence generated by an n-bit LFSR, the sequence has an occurrence of total number of (n–1) 0s in succession.

Property 7: P7

In every period (p = 2^n–1) of PN sequence generated by an n-bit LFSR, the sequence will contain 2^(k – 1) runs of (n – k – 1) 1s, as well as 0s, for $1<k\leq n-2$.

Number of Runs	Succession of	Comment
1	4 – 1s	→ P3
0	4 – 0s	→ P4
0	3 – 1s	→ P5
1	3 – 0s	→ P6
1	2 – 1s	→ P7 / Theorem 1
1	2 – 0s	→ P7 / Theorem 1
2	1 – 1s	→ P7 / Theorem 1
2	1 – 0s	→ P7 / Theorem 1
	Number of 1s = 8	→ P1
	Number of 0s = 7	→ P2

Fig.2.2 An n-bit LFSR structure

Example 1. Let us consider a 4-bit LFSR initially loaded with 0011 and has feedback taps from 3rd and 4th FFs. The generated PN sequence has its length p = 24-1 = 15. The sequence generated by this structure of LFSR is,

PN4 = [001101011110001].

The **Properties P1 – P6** and **P7** can be verified by analyzing the generated sequence PN4.

3.PN Sequence Multiplier

3.1 PN Sequence Multiplier Concept

PN Sequence gives the result of how nth-order PN sequence can be used to multiply two n/2-bitnumbers to n-bit accuracy. Since PN sequences have the span property, every n-bit segment (except all 0's) appears once and only once in each period. However, every n/2-bit segment appears 2"/* times [l] (except all-zero segments, which occur 2"12 - 1 times). When a PN sequence A, and its shifted version BI are placed side by side, either i) each n/2-bit segment in A, is adjacent to every possible n/2-bit segment in B1 only once or ii) these neighboring pairs occur an even number of times.

the following result.

Theorem: Let A, be a PN sequence and B1 its shifted version. If any n/2-bit segment of AI is adjacent to any n/2-bit segment of B, only once, then all other n/2-bit segments are adjacent only once in AI and B.

Proof: Assume that the n/2-bit segment al = aka k+l . . .at +, /, - I is adjacent to bl = bk bk + I . . . bk + n / 2 - I only once. Now al (which occurs 2"12 times) is followed by either ak+, * * * a k + n / 2 - 1 0 or . . . a k +,, ~ ,1- ~in equal proportion (i.e., times each). Similarly, bl is followed by either one of two possible segments an equal number of times. Thus, if two occurrences of a, have the same neighbor, say bZ, the remaining AL's will also have the same neighbors in pairs. But this contradicts the initial assumption. Therefore, all the neighbors of al are distinct. This is true for any n/2-bit segment, and so the result follows. The above theorem defines a search procedure for phases of type

3.2 Block Diagram

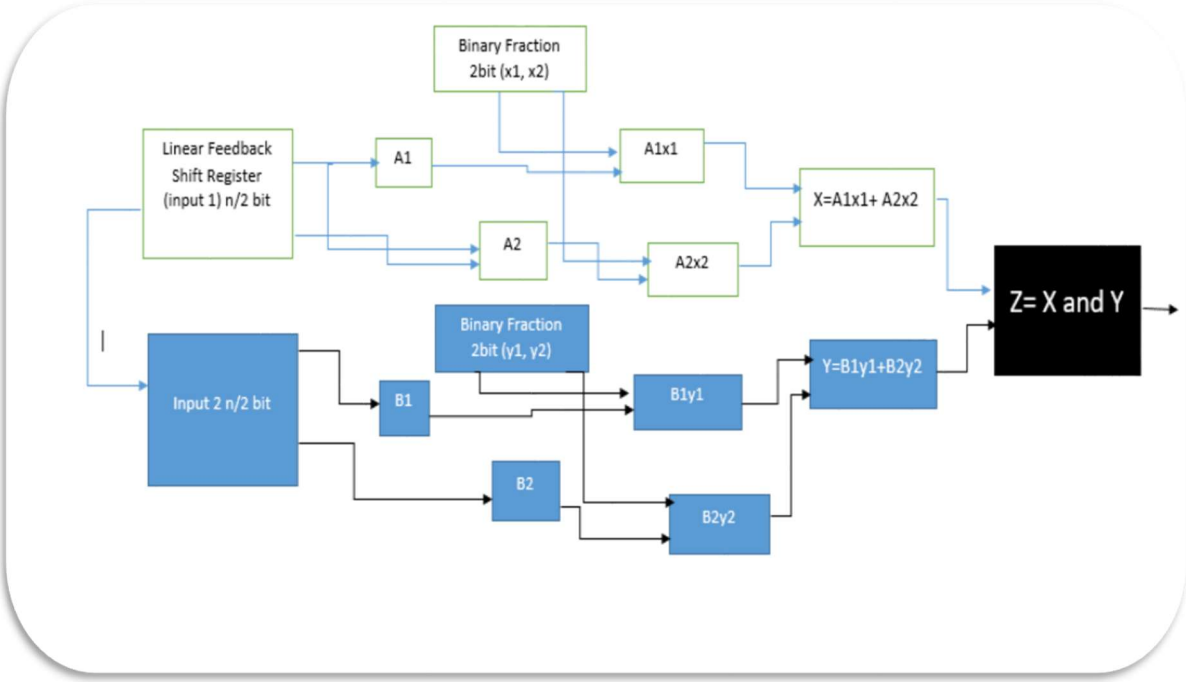


Fig.3.1 Block Diagram

3.3 Circuit Diagram

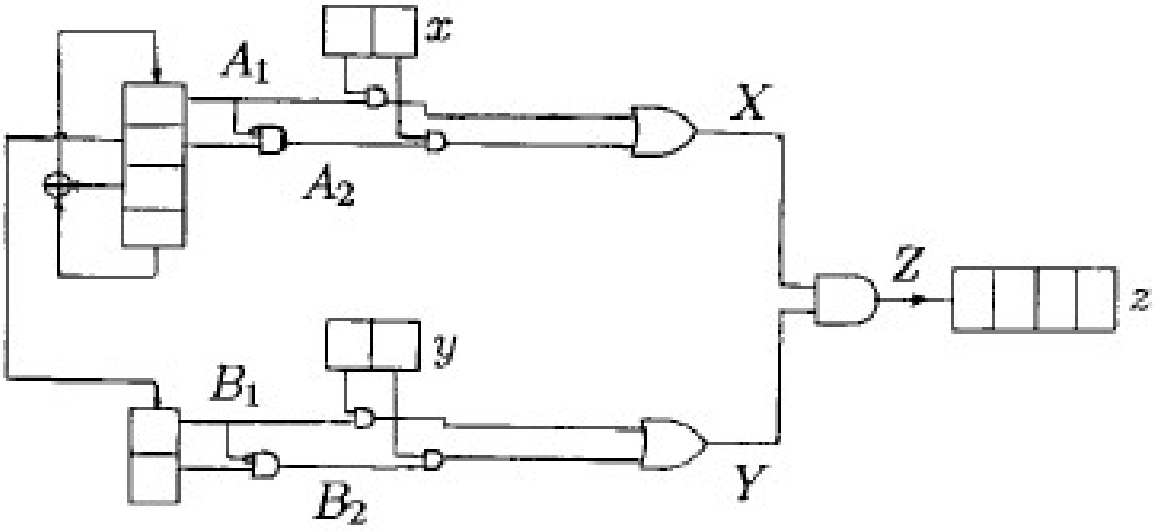


Fig.3.2 Circuit

3.4 Logical Equations

Z=X*Y

=(x1*A1+x2*A2) * (y1*B1+y2*B2)

=x1*y1*A1*B1+x1*y2*A1*B2+x2*y1*A2*B1 +x2*y2*A2*B2

3.5 Truth Table

A_1	A_2	B_1	B_2	A_1B_1	A_1B_2	A_2B_1	A_2B_2	X	Y	Z
0	1	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	1	1	1	1
1	0	1	0	1	0	0	0	1	1	1
1	0	0	0	0	0	0	0	1	0	0
1	0	0	1	0	1	0	0	1	1	1
1	0	1	0	1	0	0	0	1	1	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	0
0	1	1	0	0	0	1	0	1	1	1
1	0	0	1	0	1	0	0	1	1	1
1	0	1	0	1	0	0	0	1	1	1
0	0	1	0	0	0	0	0	0	1	0
0	1	1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	0	0	1	1	1
8	4	8	4	4	2	2	1	12	12	9

Table 2: Example x=0.11 (binary) & y= 0.11(binary)

4.Design Approach

4.1 VHDL Codes

The VHDL Codes are written and compiled in Model Sim. The VHDL codes are hardware descriptive language. The successful compilation of VHDL results in executing of simulations.

4.2 Waveform Simulation

The Vhdl codes can be further simulated in waveform to check the accuracy of the code. This check helps us to verify the results obtained are as desired or Not.

4.3 Circuit Synthesis

The Vhdl codes can be read in Mentor Leonardo to produce the circuit corresponding to the Vhdl code. The Mentor Leonardo needs the library input and code input. It converts the code in Verilog. This Verilog file is further used in Mentor IC Schematic Editor to obtain the circuit.

4.4 Making Layouts

The Mentor graphics IC layout editor is used for creating layouts according to the schematic. Different types of approaches are available for creating schematic, in this project the auto layout is followed for the successful execution of the layouts.

4.5 Timing Analysis

The Timing analysis is performed using tools called Synopsys Primetime. Timing analysis provides the details of critical path and Slack of the circuit with respect to clock.

5.Linear Feedback Shift Register

The first building block of the circuit is Linear Feedback Shift Register (LFSR). The Complete flow below explains the process of LFSR

5.1 VHDL CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lfsr_n is
        generic (constant N: integer :=4 );
        port (
                clk                     :in  std_logic;
                reset                   :in  std_logic;
                lfsr_out                :out std_logic_vector (1 downto 0)
        );
end entity;

architecture behavioral of lfsr_n is

signal lfsr_tmp :std_logic_vector (0 to N-1);
begin
        process (clk,reset)
                variable feedback       :std_logic;
        begin
                  if (reset = '1') then
                lfsr_tmp <= ('1','0','0','1');
        elsif (rising_edge(clk)) then
           feedback:=lfsr_tmp(2) xor lfsr_tmp(3);
                lfsr_tmp <= feedback & lfsr_tmp(0 to N-2);

        end if;

        end process;

        lfsr_out(0)<= lfsr_tmp(3);
        lfsr_out(1)<= lfsr_tmp(2);

end architecture;
```

5.2 Compile Results

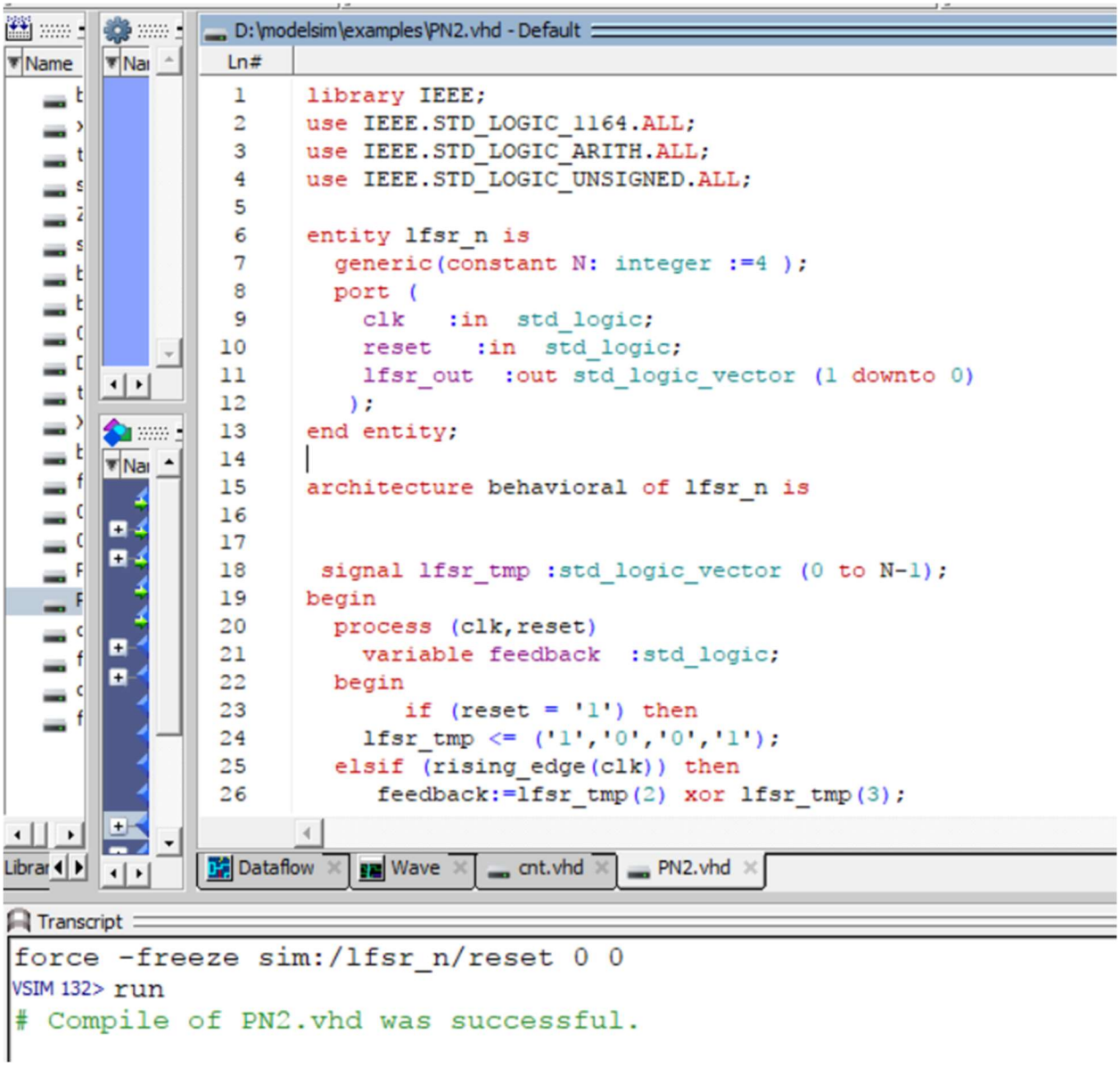


Fig5.1 Image Showing Compile Results for LFSR Code

5.3 Simulation of LFSR Code

The Simulation below explains the working of LFSR.

1) Initially reset=1; Output is set to 1001
2) Next reset=0; The execution of LFSR is carried for 15 more cycles to show the shift obtained in every step and feedback bit producing new results every cycle.
3) The output of the last flip flop is used as PN sequence. The LFSR can also be represented as PN Sequence Code generator.

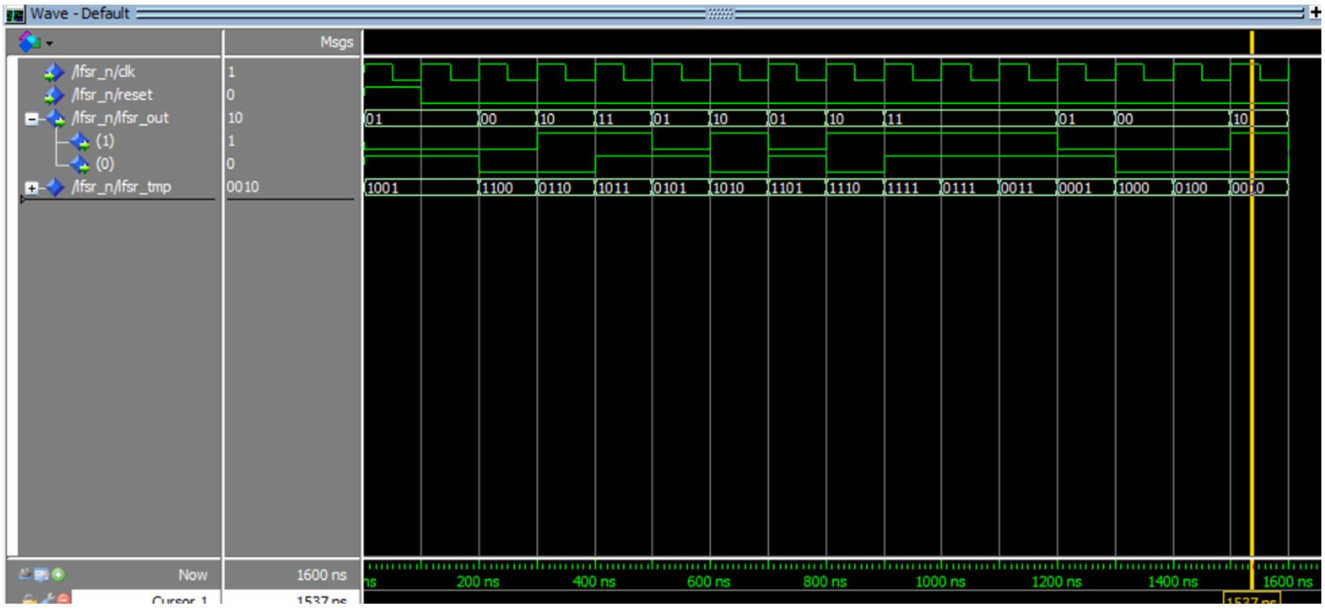


Fig5.2 Simulation Results of LFSR Code

5.4 Synthesis of LFSR

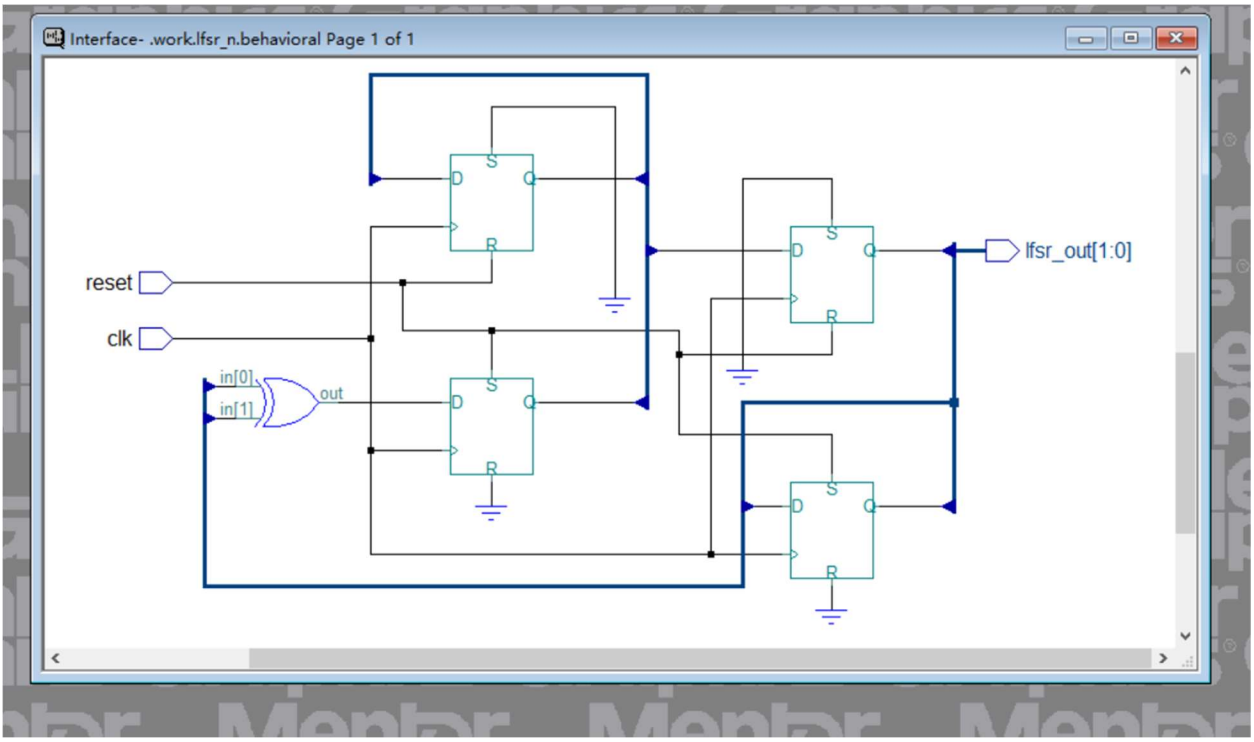


Fig.5.3 Circuit obtained from LFSR Code

6. 4 Bit PN Sequence Multiplier

The Linear Feedback Shift Register obtained previously is modified in to the 4-bit multiplier which uses PN sequences to produce the partial result and final product. The final product obtained has 4-bit accuracy.

6.1 General Circuit of 4-bit PN Sequence Multiplier

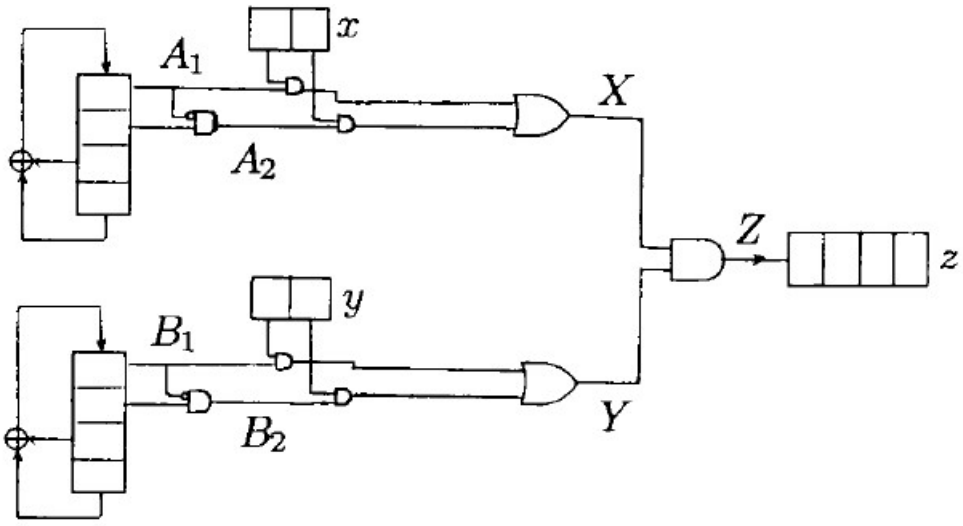


Fig6.1 General Circuit Diagram followed for VHDL code

6.2 VHDL Codes

a) VHDL Code with Counter

This Code is Executed with counter apart from above diagram to show the results of the number of 1s obtained in final circuit

Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lfsr_n is
        generic(constant N: integer :=4
        );

        port (
          x,y:in std_logic_vector(1 downto 0);
                clk                         :in  std_logic;
```

```
                reset                   :in  std_logic;
                lfsr_outB:out std_logic_vector(1 downto 0);
                lfsr_out                :out std_logic_vector (1 downto 0);
                CapX,CapY: out std_logic;
                Z:out std_logic;
                Zcnt:out integer)
                ;
end entity;

architecture behavioral of lfsr_n is
        signal lfsr_tmp                 :std_logic_vector (0 to N-1);
        signal lfsr_tmpB                :std_logic_vector (0 to N-1);

         signal Xtmp,Ytmp,Ztmp:std_logic;
         signal cnt: integer:=0;

begin
        process (clk,reset)
                variable feedback       :std_logic;
        begin
                  if (reset = '1') then
                lfsr_tmp <= (0=>'1', others=>'0');
        elsif (rising_edge(clk)) then
           feedback:=lfsr_tmp(2) xor lfsr_tmp(3);
                lfsr_tmp <= feedback & lfsr_tmp(0 to N-2);
        end if;
        end process;
        lfsr_out(0)<= lfsr_tmp(3);
        lfsr_out(1)<= lfsr_tmp(2);
        --lfsr_out(2)<= lfsr_tmp(1);
        --lfsr_out(3)<= lfsr_tmp(0);



        process (clk,reset)
                variable feedbackB      :std_logic;
        begin
                  if (reset = '1') then
                lfsr_tmpB <= ('1','0','0','1');
        elsif (rising_edge(clk)) then
           feedbackB:=lfsr_tmpB(2) xor lfsr_tmpB(3);
                lfsr_tmpB <= feedbackB & lfsr_tmpB(0 to N-2);
        end if;
        end process;
        lfsr_outB(0)<= lfsr_tmpB(3);
        lfsr_outB(1)<= lfsr_tmpB(2);
        --lfsr_outB(2)<= lfsr_tmpB(1);
```

```
        --lfsr_outB(3)<= lfsr_tmpB(0);



      Xtmp<=(lfsr_tmp(3) and x(0)) or  ((lfsr_tmp(2) and not lfsr_tmp(3)) and x(1));
      Ytmp<=(lfsr_tmpB(3) and y(0)) or  ((lfsr_tmpB(2) and not lfsr_tmpB(3))and y(1));
     CapX <=Xtmp;
      CapY<=Ytmp;
      Ztmp<=Xtmp and Ytmp;
    process(clk)
       --variable cnt: integer;
       begin

         if clk'event and clk='1' then
         if Ztmp='1'
         then cnt<=cnt+1;
         else cnt<=cnt;
         end if;
       end if;
       end process;

       Z<=Ztmp;
       Zcnt<=cnt;

end architecture;
```

b) VHDL Code with Counter

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lfsr_n is
  generic(constant N: integer :=4
  );

  port (
    x,y:in std_logic_vector(1 downto 0);
  clk :in  std_logic;
reset :in  std_logic;
lfsr_outB:out std_logic_vector(1 downto 0);
lfsr_out :out std_logic_vector (1 downto 0);
CapX,CapY: out std_logic;
Z:out std_logic)
;
end entity;

architecture behavioral of lfsr_n is
  signal lfsr_tmp :std_logic_vector (0 to N-1);
  signal lfsr_tmpB :std_logic_vector (0 to N-1);
```

```
  signal Xtmp,Ytmp,Ztmp:std_logic;


begin
  process (clk,reset)
variable feedback  :std_logic;
begin
    if (reset = '1') then
lfsr_tmp <= (0=>'1', others=>'0');
elsif (rising_edge(clk)) then
  feedback:=lfsr_tmp(2) xor lfsr_tmp(3);
lfsr_tmp <= feedback & lfsr_tmp(0 to N-2);
end if;
end process;
lfsr_out(0)<= lfsr_tmp(3);
lfsr_out(1)<= lfsr_tmp(2);
--lfsr_out(2)<= lfsr_tmp(1);
--lfsr_out(3)<= lfsr_tmp(0);
process (clk,reset)
variable feedbackB  :std_logic;
begin
    if (reset = '1') then
lfsr_tmpB <= ('1','0','0','1');
elsif (rising_edge(clk)) then
  feedbackB:=lfsr_tmpB(2) xor lfsr_tmpB(3);
lfsr_tmpB <= feedbackB & lfsr_tmpB(0 to N-2);
end if;
end process;
lfsr_outB(0)<= lfsr_tmpB(3);
lfsr_outB(1)<= lfsr_tmpB(2);
--lfsr_outB(2)<= lfsr_tmpB(1);
--lfsr_outB(3)<= lfsr_tmpB(0);



       Xtmp<=(lfsr_tmp(3) and x(0)) or  ((lfsr_tmp(2) and not lfsr_tmp(3)) and x(1));
       Ytmp<=(lfsr_tmpB(3) and y(0)) or  ((lfsr_tmpB(2) and not lfsr_tmpB(3))and y(1));
      CapX <=Xtmp;
       CapY<=Ytmp;
       Ztmp<=Xtmp and Ytmp;

        Z<=Ztmp;


end architecture;
```

6.3 Compilation Results

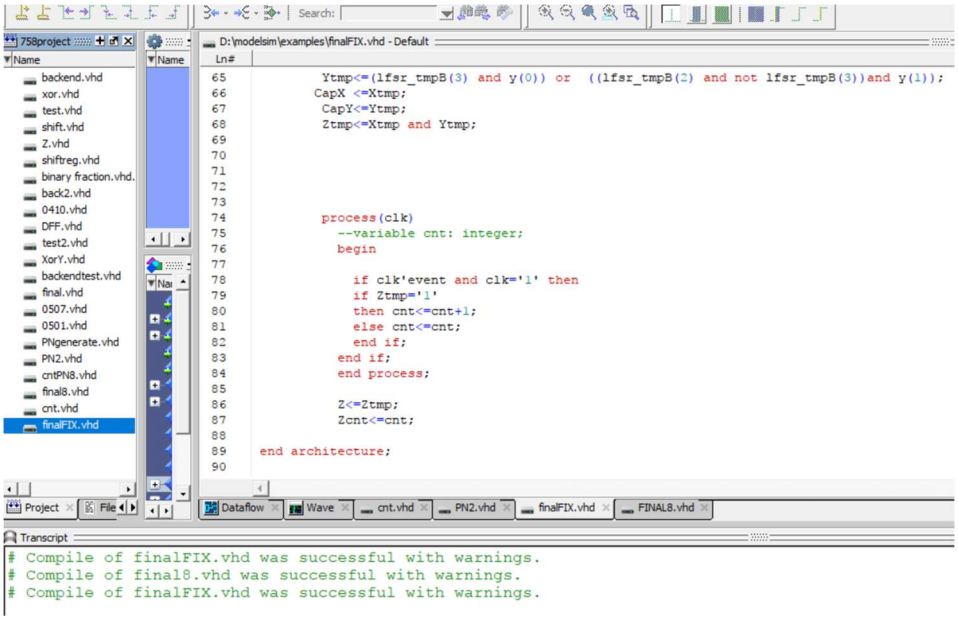


Fig.6.2 VHDL Successful compilation for program with counter

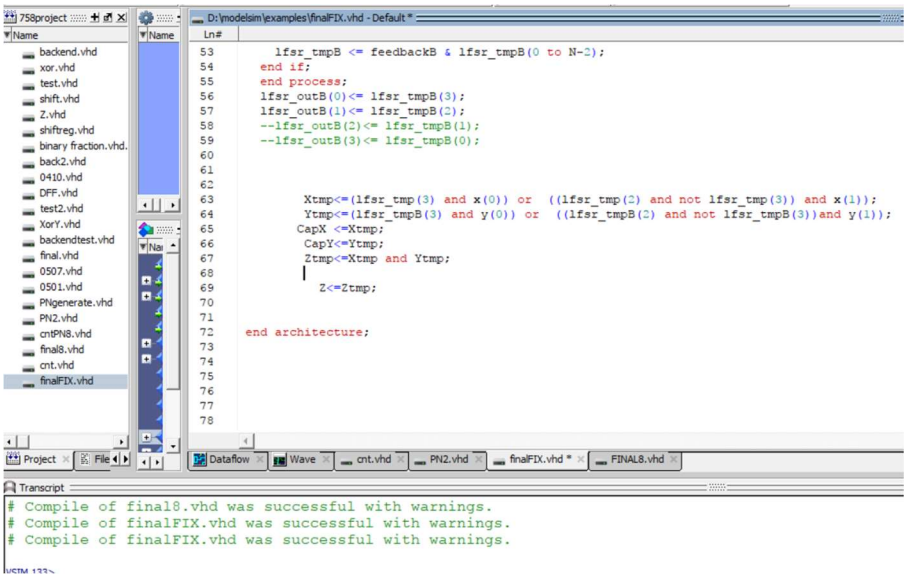


Fig.6.3 VHDL Successful compilation for program without counter

6.4 Simulation Results of 4 bit PN Sequence Multiplier

a) When input is x=0.11 and y=0.11 in binary

Z shows the graph of result after multiplication

The number of 1s in the Z is show through a count recorded through counter

Zcnt = 9

The result can be verified with table provided at end of this section.

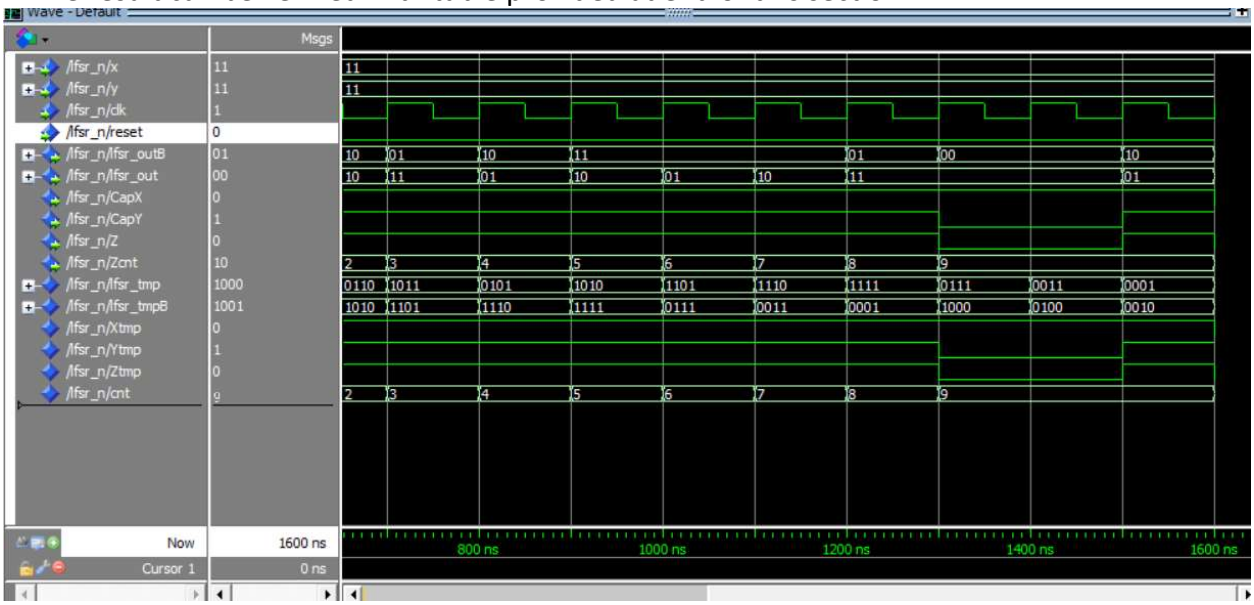


Fig.6.4 Simulation for x=0.11 and y=0.11; Zcnt=9

b) When input is x=0.11 and y=0.01 in binary

Z shows the graph of result after multiplication

The number of 1s in the Z is show through a count recorded through counter

Zcnt = 6

The result can be verified with table provided at end of this section.

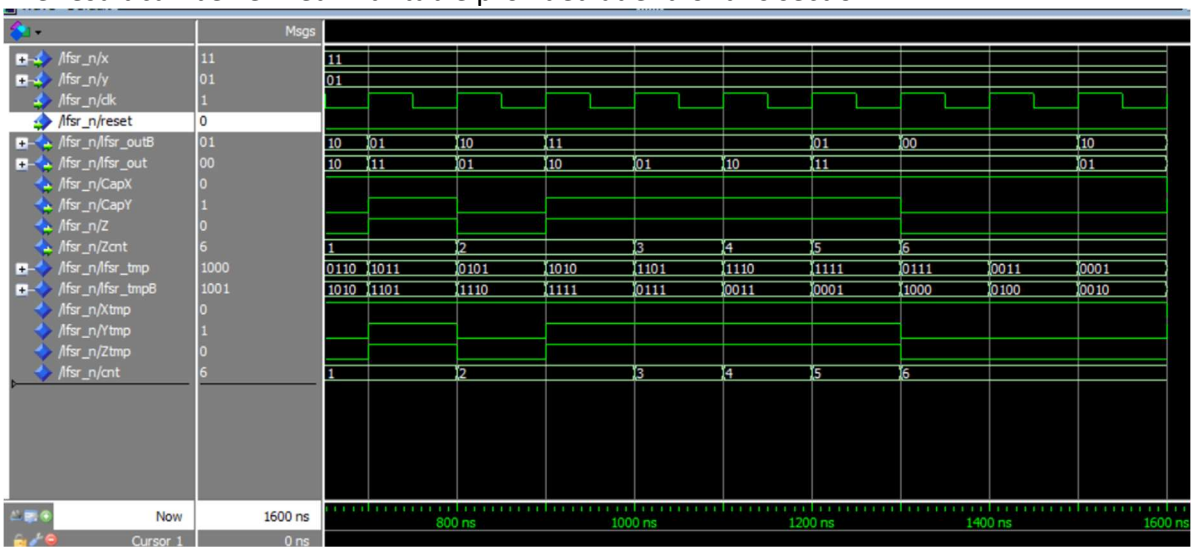


Fig.6.5 Simulation for x=0.11 and y=0.01; Zcnt=6

c) When input is x=0.01 and y=0.01 in binary

Z shows the graph of result after multiplication

The number of 1s in the Z is show through a count recorded through counter

Zcnt = 4

The result can be verified with table provided at end of this section.

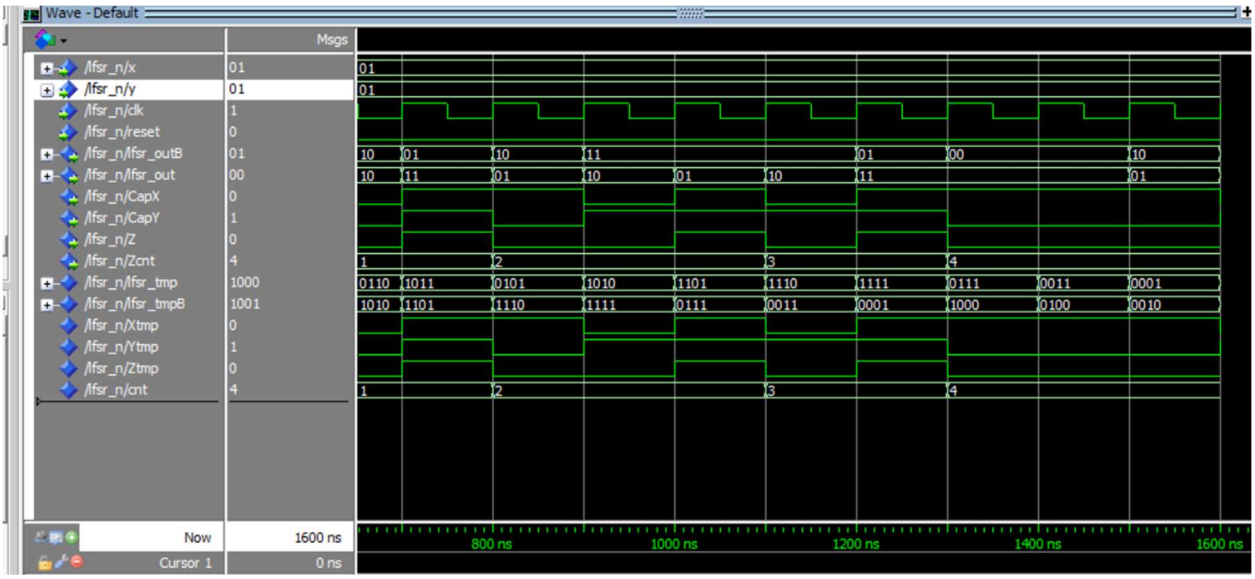


Fig.6.6 Simulation for x=0.01 and y=0.01; Zcnt=4

d) When input is x=0.01 and y=0.10 in binary

Z shows the graph of result after multiplication

The number of 1s in the Z is show through a count recorded through counter

Zcnt = 2

The result can be verified with table provided at end of this section.

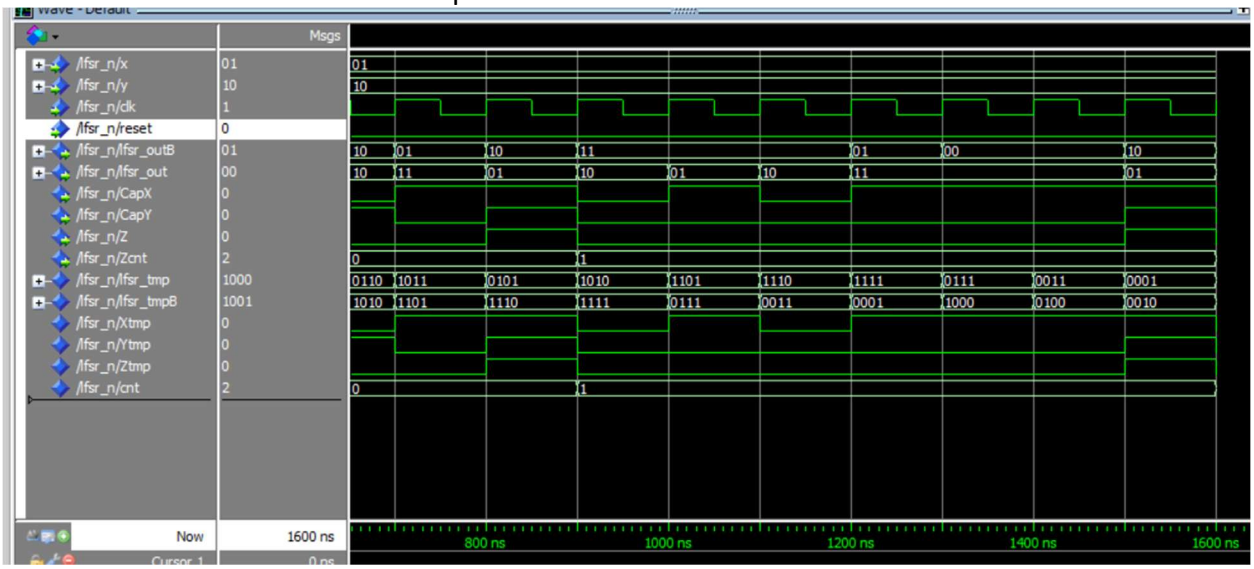


Fig.6.7 Simulation for x=0.01 and y=0.10; Zcnt=2

6.5 Circuit Synthesis

a) Complete Circuit

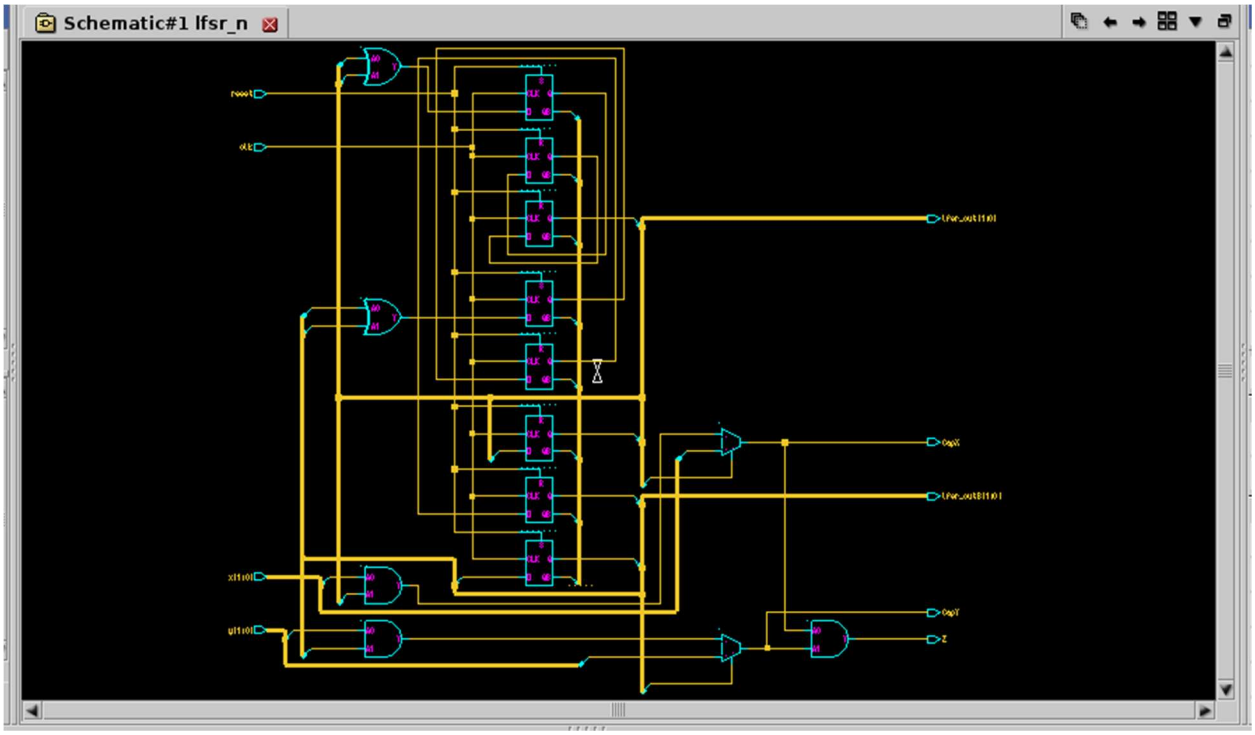


Fig.6.8 Circuit obtained from Mentor Leonardo

b) Symbol of above Circuit

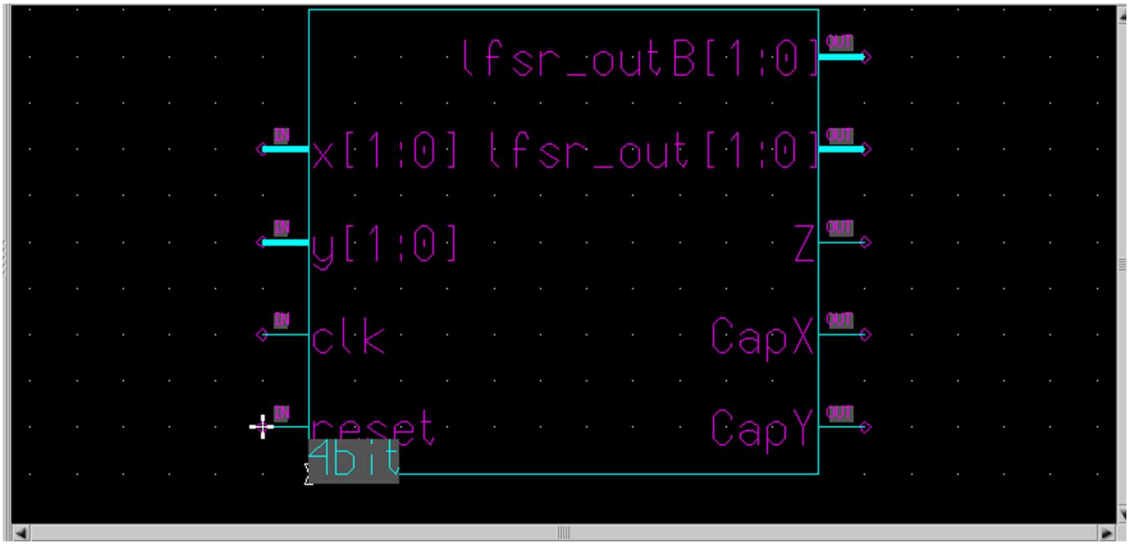


Fig.6.9 Symbol of 4-bit Multiplier

c) Creating input output pads to the circuit of 4 bit PN sequence Multiplier. There were 6 input pads and 7 output pads followed by 2 Vdd and 2 Gnd Pads.

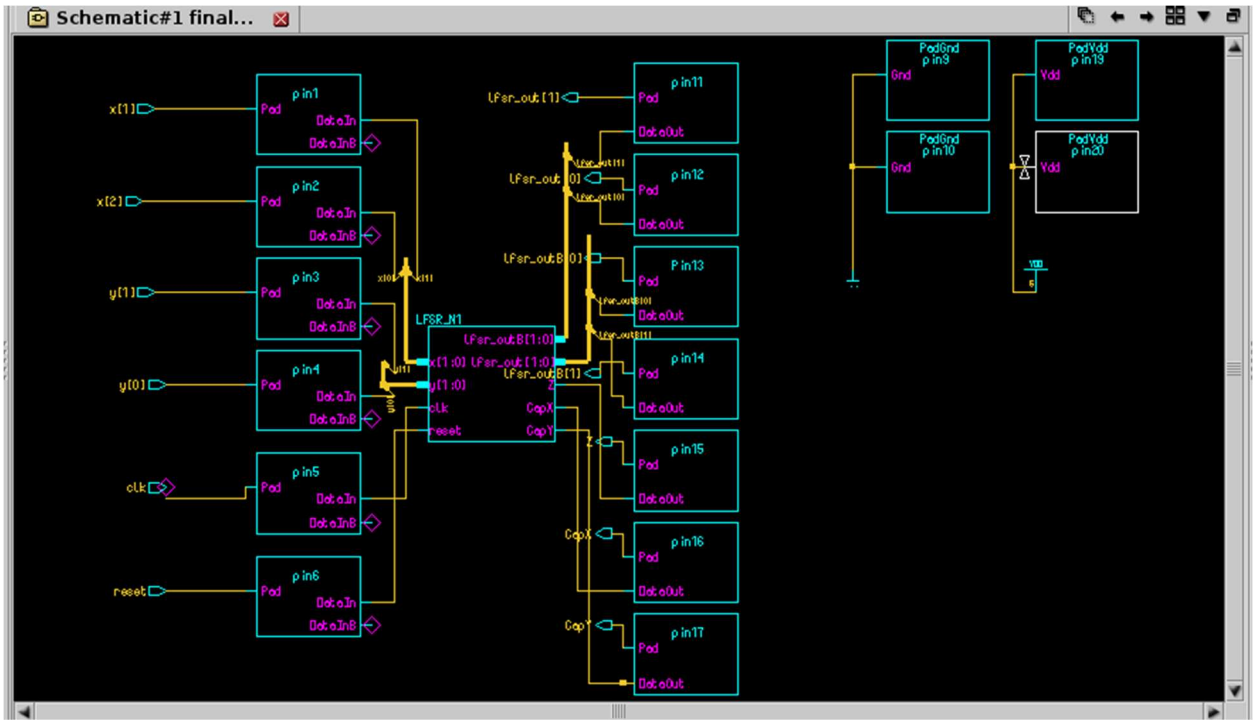


Fig6.10 Pad in and Pad out

6.6 Layout of 4 bit PN Sequence Multiplier

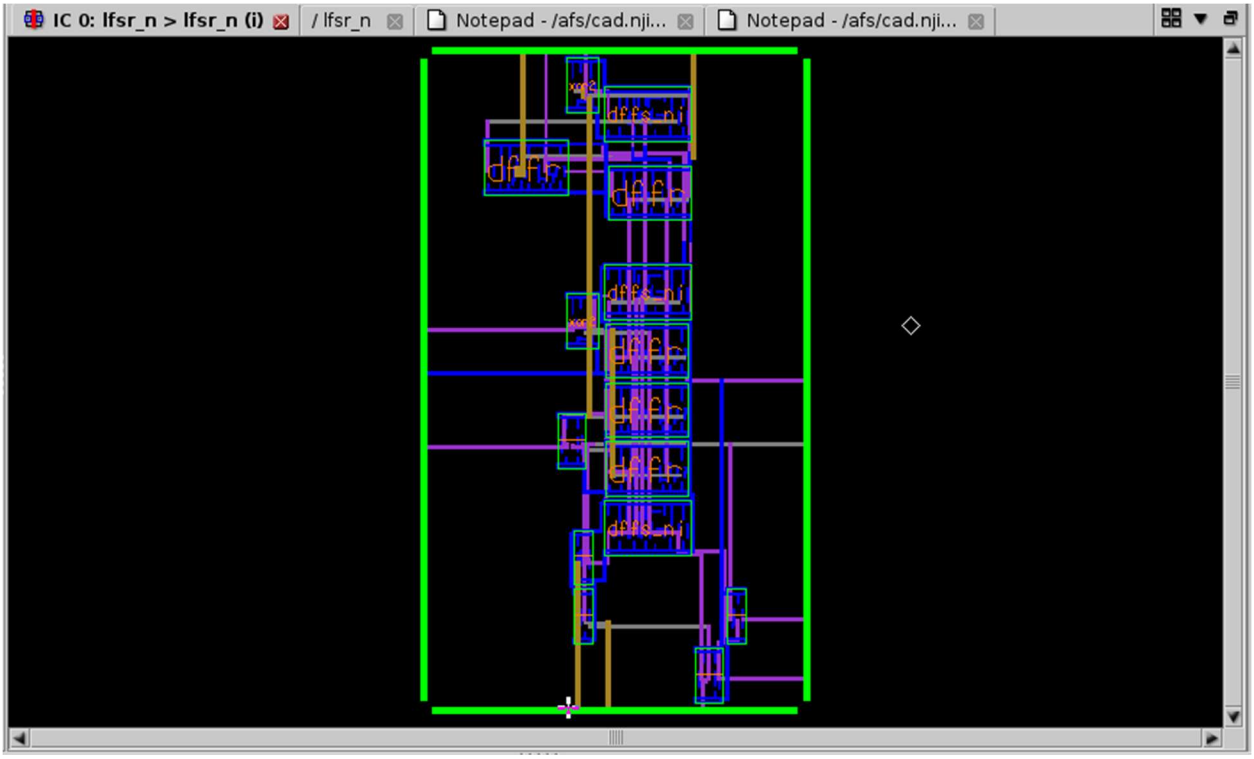


Fig6.11 Layout with Auto Routing

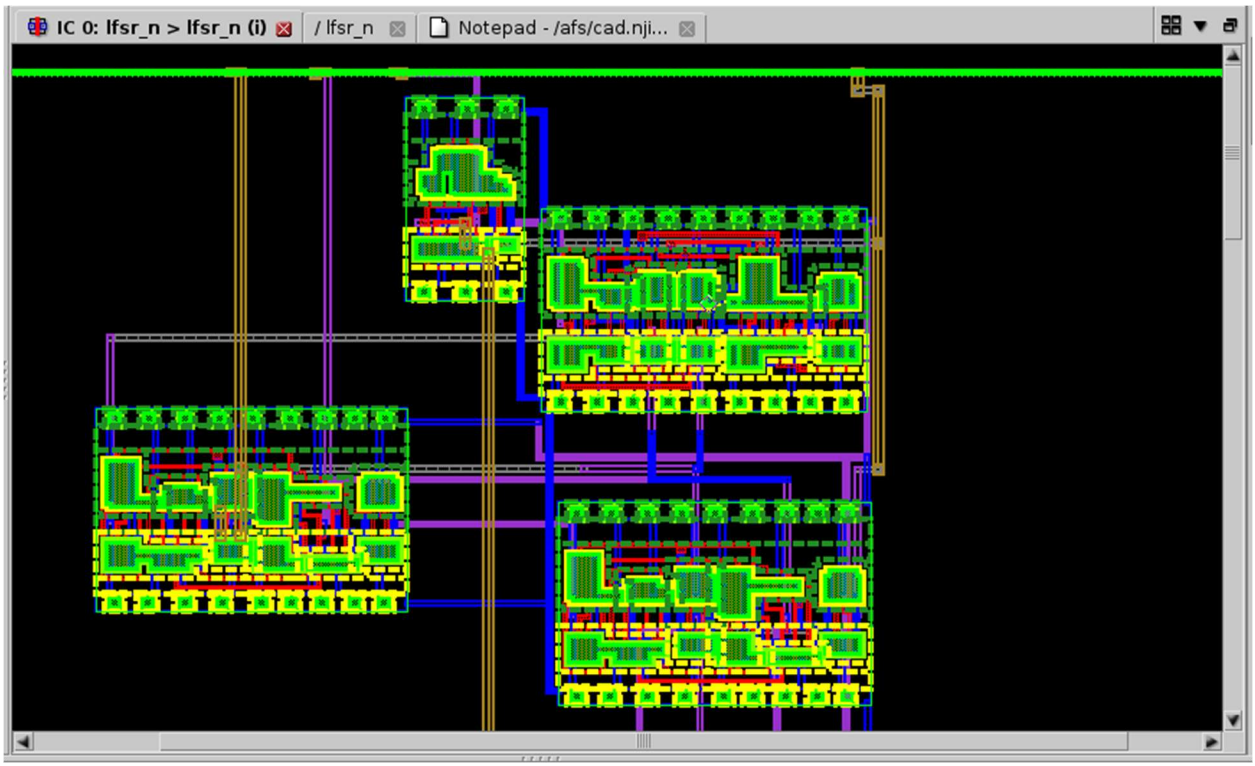


Fig.6.12 Layout showing Peek Area

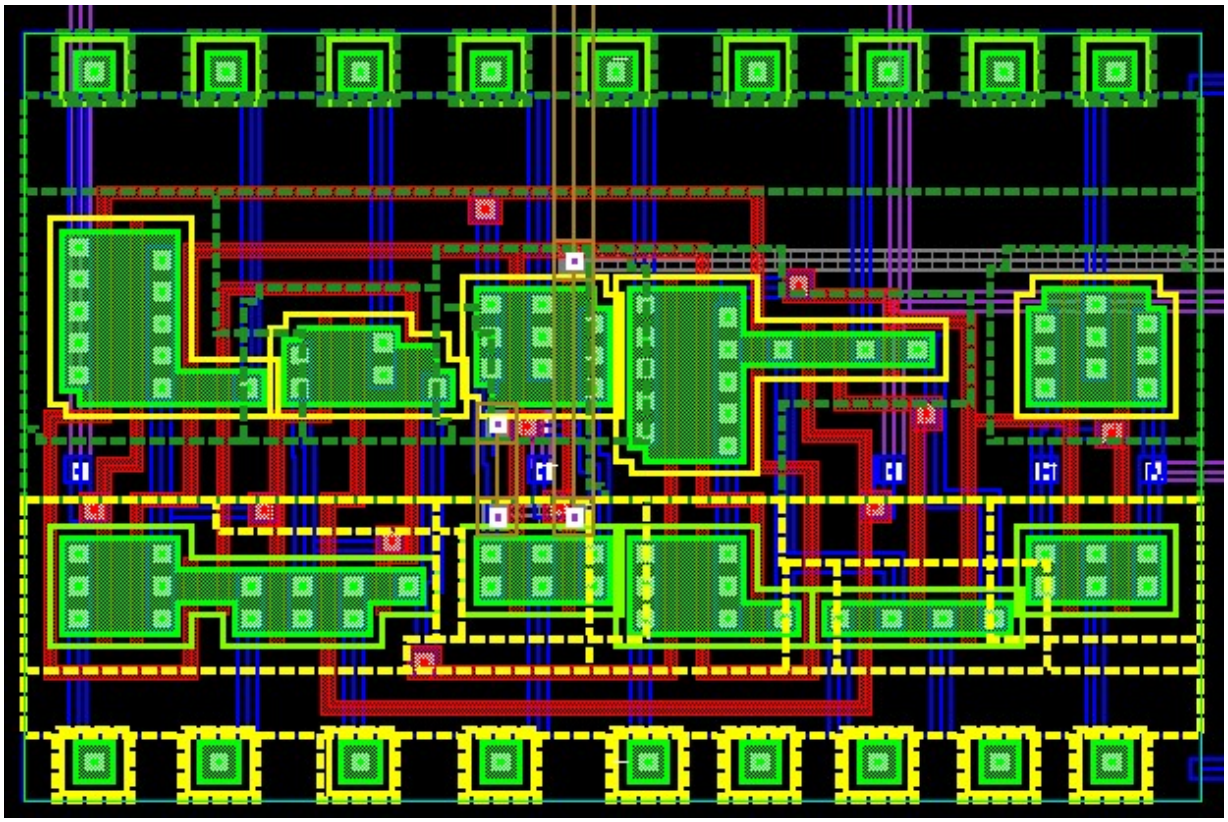


Fig6.13 A part of layout showing the D flip flop layout

IC 0: lfsr_n > lfsr_n (i) | / lfsr_n | Notepad - /afs/cad.nji... | Notepad - /afs/cad...

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


REPORT FILE NAME:         /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/output/layout/lvs.rep
LAYOUT NAME:              /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/output/layout/lfsr_n
SOURCE NAME:              /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/output/lfsr_n/sdl
RULE FILE:                /afs/cad/sw.common/mentor.2012/adk3_1/technology/ic/process/tsmc035.rules
LVS MODE:                 Mask
RULE FILE NAME:           /afs/cad/sw.common/mentor.2012/adk3_1/technology/ic/process/tsmc035.rules
CREATION TIME:            Mon May  8 18:11:18 2017
CURRENT DIRECTORY:        /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/output/layout
USER NAME:                sk2339


*****************************************************************************************************************
                              OVERALL COMPARISON RESULTS
*****************************************************************************************************************


                      #       ###################       _   _
                    #         #                 #       *   *
                  #   #       #     CORRECT     #         |
                   # #        #                 #       \___/
                    #         ###################


-----------------------------------------------------------------------------------------------------------------

INITIAL NUMBERS OF OBJECTS
```

Fig.6.14 LVS Report Successful

6.7 Timing Analysis and Delay report

a) Critical Path & Area Report &Critical Path Report

```
**************************************************

Cell: lfsr_n    View: behavioral    Library: work

**************************************************

 Number of ports :                  13
 Number of nets :                   21
 Number of instances :              15
 Number of references to this view :  0

Total accumulated area :
 Number of gates :                  57
 Number of accumulated instances :  15
Info, Command 'report_area' finished successfully
LEONARDO{7}:
```

```
                    Critical Path Report

Critical path #1, (unconstrained path)
NAME                            GATE          ARRIVAL            LOAD
------------------------------------------------------------------------
clock information not specified
delay thru clock network                      0.00 (ideal)


reg_lfsr_tmpB(2)/Q              dffr     0.00 0.53 dn          0.05
ix47/Y                          and02    0.26 0.78 dn          0.01
ix55/Y                          mux21_ni 0.26 1.04 dn          0.01
ix57/Y                          and02    0.19 1.23 dn          0.00
Z/                                       0.00 1.23 dn          0.00
data arrival time                             1.23

data required time                      not specified
------------------------------------------------------------------------
data required time                      not specified
data arrival time                         1.23
                                        ---------
                                      unconstrained path
------------------------------------------------------------------------
```

Fig.6.15 Area and delay report obtained by Mentor Leonardo

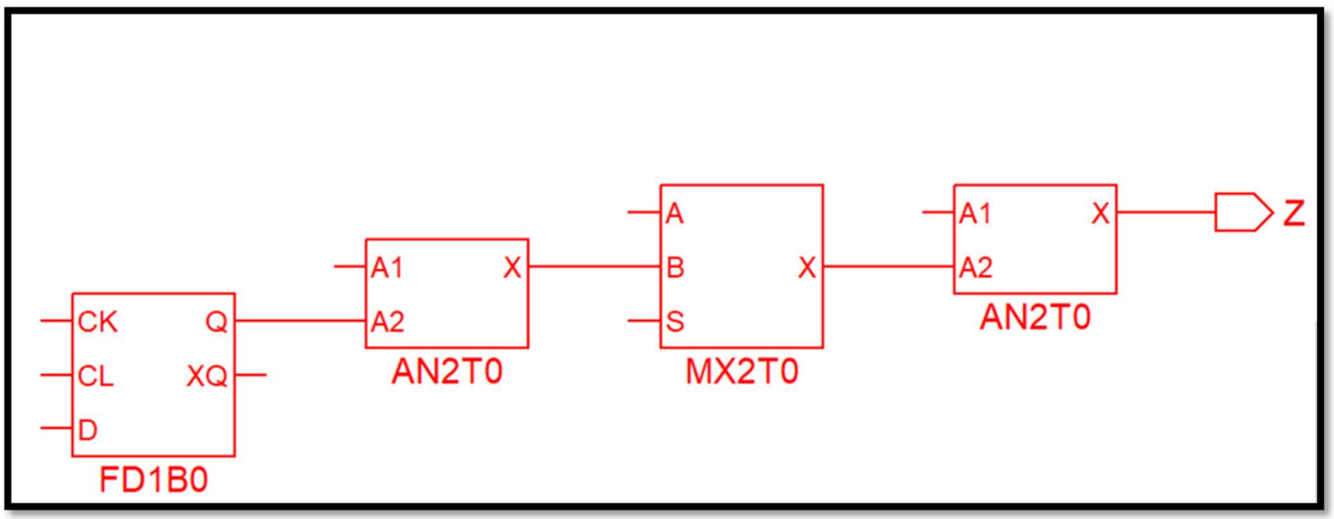


Fig.6.16 Critical Path

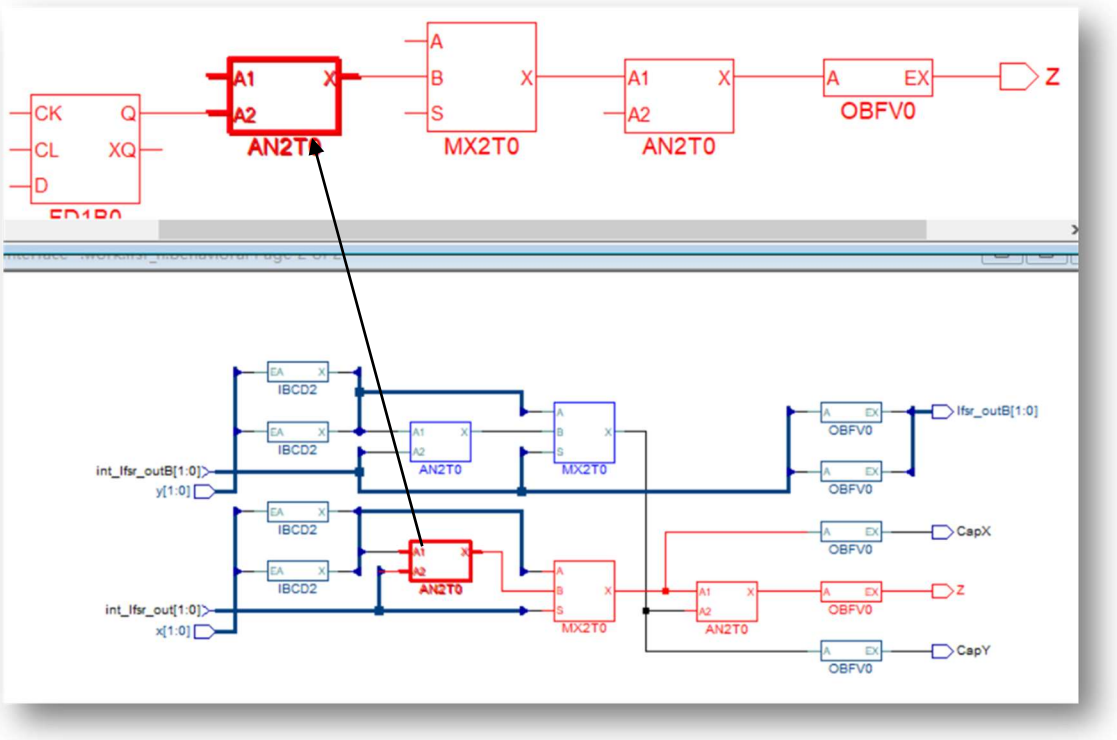


Fig.6.17 Optimized Circuit Showing critical path

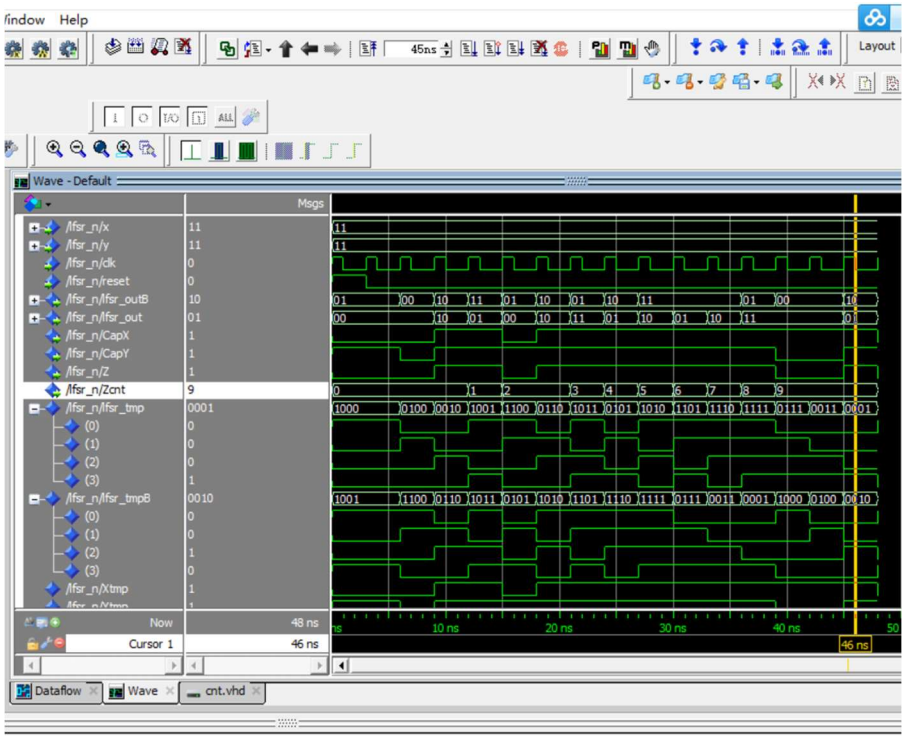
b)Verilog Code for Execution in Primetime

```
module lfsr_n ( x, y, clk, reset, lfsr_outB, lfsr_out, CapX, CapY, Z ) ;

    input [1:0]x ;
    input [1:0]y ;
    input clk ;
    input reset ;
    output [1:0]lfsr_outB ;
    output [1:0]lfsr_out ;
    output CapX ;
    output CapY ;
    output Z ;

    wire lfsr_tmp_1, lfsr_tmp_0, nx0, nx18, lfsr_tmpB_1, lfsr_tmpB_0, nx28, nx46
        ;
```

```
    wire [7:0] \$dummy ;
    AN2 ix57 (.Z (Z), .A (CapX), .B (CapY)) ;
    MUX21H ix27 (.Z (CapX), .A (nx18), .B (x[0]), .S (lfsr_out[0])) ;
    AN2 ix19 (.Z (nx18), .A (x[1]), .B (lfsr_out[1])) ;
    FD1 reg_lfsr_tmp_2 (.Q (lfsr_out[1]), .QN (\$dummy [0]), .D (lfsr_tmp_1), .CP (
        clk)) ;
    FD1 reg_lfsr_tmp_1 (.Q (lfsr_tmp_1), .QN (\$dummy [1]), .D (lfsr_tmp_0), .CP (
        clk)) ;
    FD1 reg_lfsr_tmp_0 (.Q (lfsr_tmp_0), .QN (\$dummy [2]), .D (nx0), .CP (
            clk)) ;
    EOP ix1 (.Z (nx0), .A (lfsr_out[1]), .B (lfsr_out[0])) ;
    FD1 reg_lfsr_tmp_3 (.Q (lfsr_out[0]), .QN (\$dummy [3]), .D (lfsr_out[1]),
        .CP (clk)) ;
    MUX21H ix55 (.Z (CapY), .A (nx46), .B (y[0]), .S (lfsr_outB[0])) ;
    AN2 ix47 (.Z (nx46), .A (y[1]), .B (lfsr_outB[1])) ;
    FD1 reg_lfsr_tmpB_2 (.Q (lfsr_outB[1]), .QN (\$dummy [4]), .D (lfsr_tmpB_1)
        , .CP (clk)) ;
    FD1 reg_lfsr_tmpB_1 (.Q (lfsr_tmpB_1), .QN (\$dummy [5]), .D (lfsr_tmpB_0)
        , .CP (clk)) ;
   FD1 reg_lfsr_tmpB_0 (.Q (lfsr_tmpB_0), .QN (\$dummy [6]), .D (nx28), .CP (
            clk)) ;
    EOP ix29 (.Z (nx28), .A (lfsr_outB[1]), .B (lfsr_outB[0])) ;
    FD1 reg_lfsr_tmpB_3 (.Q (lfsr_outB[0]), .QN (\$dummy [7]), .D (
            lfsr_outB[1]), .CP (clk)) ;
endmodule
```

C) Primetime Slack, Hold and Setup Time Violation Check

I) Fig 6.18 Simulation of Circuit with **CLOCK CYCLE 3 ns** with accurate results

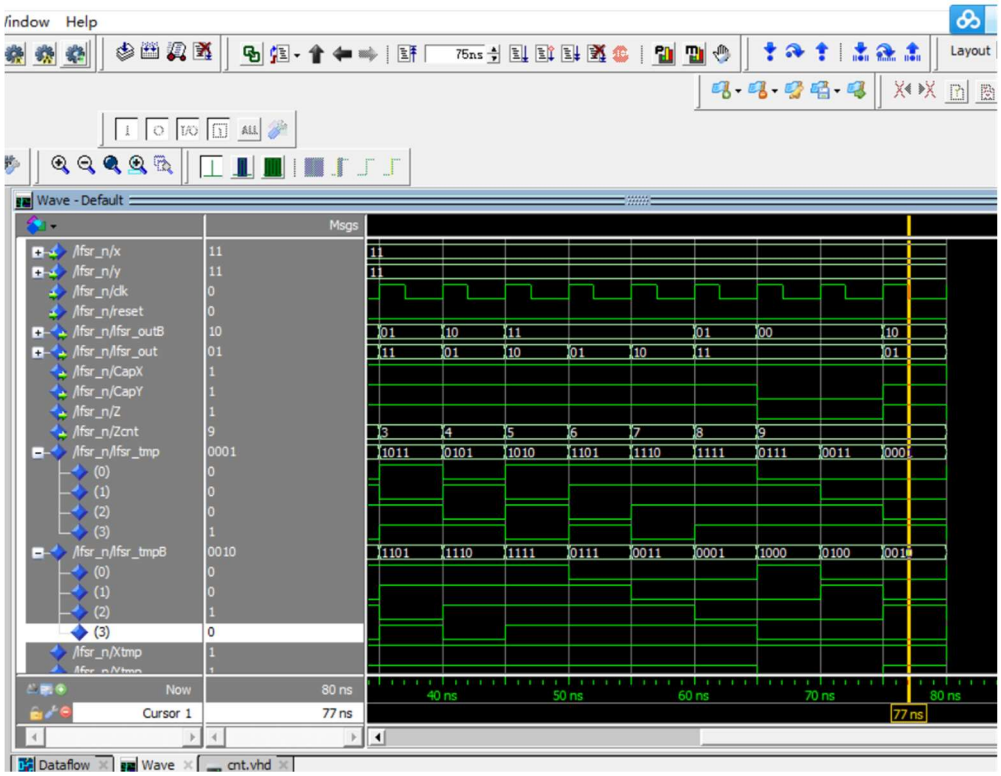


II) Fig 6.19 Timing Report Showing Slack Violated at 3 ns

```
Last common pin: clk
Path Group: clk
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (propagated)         0.00       0.00
reg_lfsr_tmpB_3/CP (FD1)                 0.00       0.00 r
reg_lfsr_tmpB_3/Q (FD1)                  1.78       1.78 r
ix29/Z (EOP)                             1.11       2.89 f
reg_lfsr_tmpB_0/D (FD1)                  0.00       2.89 f
data arrival time                                   2.89

clock clk (rise edge)                    3.00       3.00
clock network delay (propagated)         0.00       3.00
clock reconvergence pessimism            0.00       3.00
clock uncertainty                       -0.50       2.50
reg_lfsr_tmpB_0/CP (FD1)                            2.50 r
library setup time                      -0.80       1.70
data required time                                  1.70
-----------------------------------------------------------
data required time                                  1.70
data arrival time                                  -2.89
-----------------------------------------------------------
slack (VIOLATED)                                   -1.19
```

Hence the clock cycle was increased to 5 ns to check the slack violation followed by hold and setup time violation

I)Fig 6.20 Simulation of Circuit with **CLOCK CYCLE 5 ns** with accurate results



II) Fig 6.21 Timing Report Showing Slack met at 5 ns

```
Startpoint: reg_lfsr_tmpB_3
            (rising edge-triggered flip-flop clocked by clk)
Endpoint: reg_lfsr_tmpB_0
            (rising edge-triggered flip-flop clocked by clk)
Last common pin: clk
Path Group: clk
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (propagated)         0.00       0.00
reg_lfsr_tmpB_3/CP (FD1)                 0.00       0.00 r
reg_lfsr_tmpB_3/Q (FD1)                  1.78       1.78 r
ix29/Z (EOP)                             1.11       2.89 f
reg_lfsr_tmpB_0/D (FD1)                  0.00       2.89 f
data arrival time                                   2.89

clock clk (rise edge)                    5.00       5.00
clock network delay (propagated)         0.00       5.00
clock reconvergence pessimism            0.00       5.00
clock uncertainty                       -0.50       4.50
reg_lfsr_tmpB_0/CP (FD1)                            4.50 r
library setup time                      -0.80       3.70
data required time                                  3.70
-----------------------------------------------------------
data required time                                  3.70
data arrival time                                  -2.89
-----------------------------------------------------------
slack (MET)                                         0.81
```

6.8 Results and Calculations

X	11	11	11	10	01	01	10	01
Y	11	10	01	11	11	10	01	01
Z(#of 1s/15)	9/15	3/15	6/15	4/15	6/15	2/15	2/15	4/15
ACCURATE	0.56	0.375	0.1875	0.375	0.1875	0.125	0.125	0.375
4-BIT	0.6	0.2	0.4	0.27	0.4	0.13	0.13	0.27

Table 3 Optimized Circuit Showing critical path

7. 8 Bit PN Sequence Multiplier

7.1 8 bit LFSR General Block Diagram

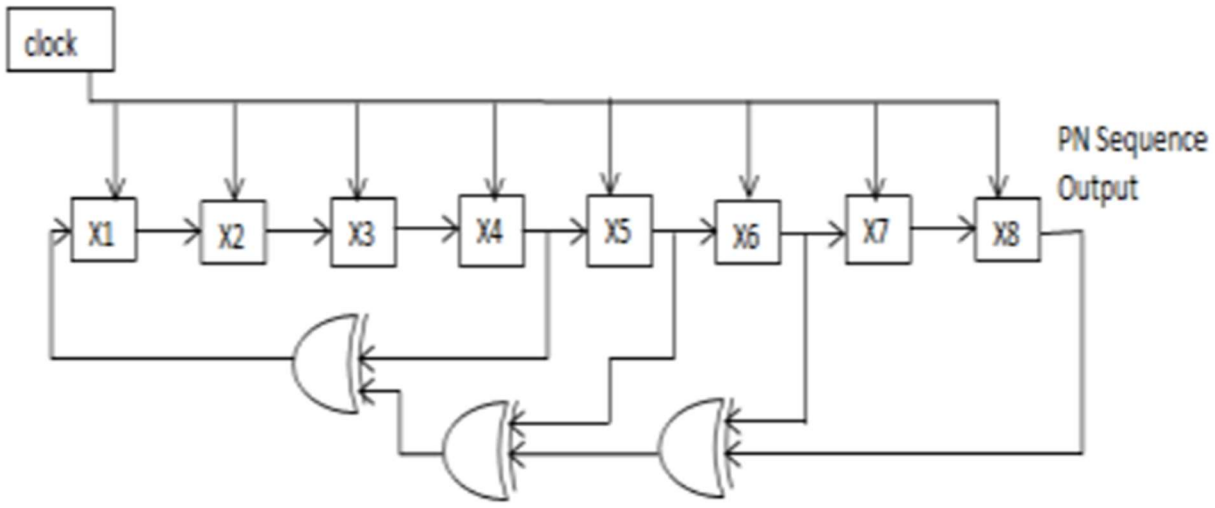


Fig.7.1 General 8 bit PN sequence Generator

7.2 VHDL CODES

a) VHDL Code with Counter

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lfsr_n is
        generic(constant N: integer :=8
        );

        port (
          x,y:in std_logic_vector(3 downto 0);
                clk                     :in  std_logic;
                reset                   :in  std_logic;
                lfsr_outB:out std_logic_vector(3 downto 0);
                lfsr_out                :out std_logic_vector (3 downto 0);
                CapX,CapY: out std_logic;
                Z:out std_logic;
                Zcnt:out integer)
```

```
                ;
end entity;

architecture behavioral of lfsr_n is
        signal lfsr_tmp                 :std_logic_vector (0 to N-1);
        signal lfsr_tmpB                :std_logic_vector (0 to N-1);

         signal Xtmp,Ytmp,Ztmp:std_logic;
         signal cnt: integer:=0;

begin
        process (clk,reset)
                variable feedback       :std_logic;
        begin
                 if (reset = '1') then
                lfsr_tmp <= (0=>'1', others=>'1');
        elsif (rising_edge(clk)) then
           feedback:=(((lfsr_tmp(5) xor lfsr_tmp(7)) xor lfsr_tmp(4))xor lfsr_tmp(3));
                lfsr_tmp <= feedback & lfsr_tmp(0 to N-2);
        end if;
        end process;
        lfsr_out(0)<= lfsr_tmp(7);
        lfsr_out(1)<= lfsr_tmp(6);
        lfsr_out(2)<= lfsr_tmp(5);
        lfsr_out(3)<= lfsr_tmp(4);



        process (clk,reset)
                variable feedbackB     :std_logic;
        begin
                 if (reset = '1') then
                lfsr_tmpB <= ('1','1','1','1','1','1','1','1');
        elsif (rising_edge(clk)) then
           feedbackB:=(((lfsr_tmpB(5) xor lfsr_tmpB(7)) xor lfsr_tmpB(4))xor lfsr_tmpB(3));
                lfsr_tmpB <= feedbackB & lfsr_tmpB(0 to N-2);
        end if;
        end process;
        lfsr_outB(0)<= lfsr_tmpB(7);
        lfsr_outB(1)<= lfsr_tmpB(6);
        lfsr_outB(2)<= lfsr_tmpB(5);
        lfsr_outB(3)<= lfsr_tmpB(4);



      Xtmp<=(lfsr_tmp(7) and x(0)) or  ((lfsr_tmp(6) and not lfsr_tmp(7)) and x(1)) or
((lfsr_tmp(5) and not lfsr_tmp(6)) and x(2)) or ((lfsr_tmp(4) and not lfsr_tmp(5)) and x(3));
```

```
      Ytmp<=(lfsr_tmpB(7) and y(0)) or  ((lfsr_tmpB(6) and not lfsr_tmpB(7)) and y(1)) or
((lfsr_tmpB(5) and not lfsr_tmpB(6)) and y(2)) or ((lfsr_tmpB(4) and not lfsr_tmpB(5)) and
y(3));

     CapX <=Xtmp;
      CapY<=Ytmp;
      Ztmp<=Xtmp and Ytmp;
      process(clk)
        --variable cnt: integer;
        begin

          if clk'event and clk='1' then
          if Ztmp='1'
          then cnt<=cnt+1;
          else cnt<=cnt;
          end if;
        end if;
        end process;

        Z<=Ztmp;
        Zcnt<=cnt;

end architecture;
```

b) VHDL Code without Counter

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lfsr_n is
        generic(constant N: integer :=8
        );

        port (
          x,y:in std_logic_vector(3 downto 0);
                clk                     :in  std_logic;
                reset                   :in  std_logic;
                lfsr_outB:out std_logic_vector(3 downto 0);
                lfsr_out                :out std_logic_vector (3 downto 0);
                CapX,CapY: out std_logic;
                Z:out std_logic)
                ;
end entity;

architecture behavioral of lfsr_n is
        signal lfsr_tmp                         :std_logic_vector (0 to N-1);
```

```
        signal lfsr_tmpB                        :std_logic_vector (0 to N-1);

         signal Xtmp,Ytmp,Ztmp:std_logic;


begin
        process (clk,reset)
                variable feedback       :std_logic;
        begin
                  if (reset = '1') then
                lfsr_tmp <= (0=>'1', others=>'1');
        elsif (rising_edge(clk)) then
           feedback:=(((lfsr_tmp(5) xor lfsr_tmp(7)) xor lfsr_tmp(4))xor lfsr_tmp(3));
                lfsr_tmp <= feedback & lfsr_tmp(0 to N-2);
        end if;
        end process;
        lfsr_out(0)<= lfsr_tmp(7);
        lfsr_out(1)<= lfsr_tmp(6);
        lfsr_out(2)<= lfsr_tmp(5);
        lfsr_out(3)<= lfsr_tmp(4);



        process (clk,reset)
                variable feedbackB     :std_logic;
        begin
                  if (reset = '1') then
                lfsr_tmpB <= ('1','1','1','1','1','1','1','1');
        elsif (rising_edge(clk)) then
           feedbackB:=(((lfsr_tmpB(5) xor lfsr_tmpB(7)) xor lfsr_tmpB(4))xor lfsr_tmpB(3));
                lfsr_tmpB <= feedbackB & lfsr_tmpB(0 to N-2);
        end if;
        end process;
        lfsr_outB(0)<= lfsr_tmpB(7);
        lfsr_outB(1)<= lfsr_tmpB(6);
        lfsr_outB(2)<= lfsr_tmpB(5);
        lfsr_outB(3)<= lfsr_tmpB(4);



      Xtmp<=(lfsr_tmp(7) and x(0)) or  ((lfsr_tmp(6) and not lfsr_tmp(7)) and x(1)) or
((lfsr_tmp(5) and not lfsr_tmp(6)) and x(2)) or ((lfsr_tmp(4) and not lfsr_tmp(5)) and x(3));
      Ytmp<=(lfsr_tmpB(7) and y(0)) or  ((lfsr_tmpB(6) and not lfsr_tmpB(7)) and y(1)) or
((lfsr_tmpB(5) and not lfsr_tmpB(6)) and y(2)) or ((lfsr_tmpB(4) and not lfsr_tmpB(5)) and
y(3));

     CapX <=Xtmp;
```

```
CapY<=Ytmp;
Ztmp<=Xtmp and Ytmp;


 Z<=Ztmp;

end architecture;
```

7.3 Compilation Results of two codes

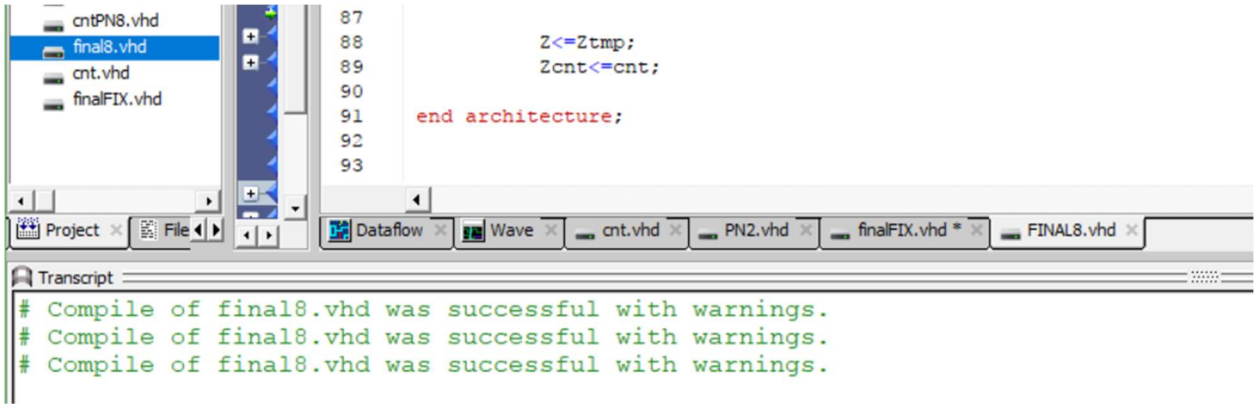


Fig.7.2 Compile results of code with counter

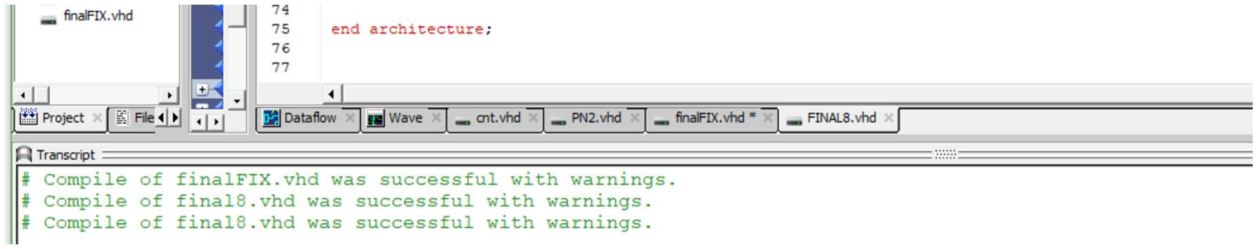


Fig.7.3 Compile results of code without counter

7.4 Simulation of the code

a) When input is x=0.1100 and y=0.1100 in binary

Z shows the graph of result after multiplication

The number of 1s in the Z is show through a count recorded through counter

Zcnt = 128 (Final Value in Graph)

The result can be verified with table provided at end of this section.

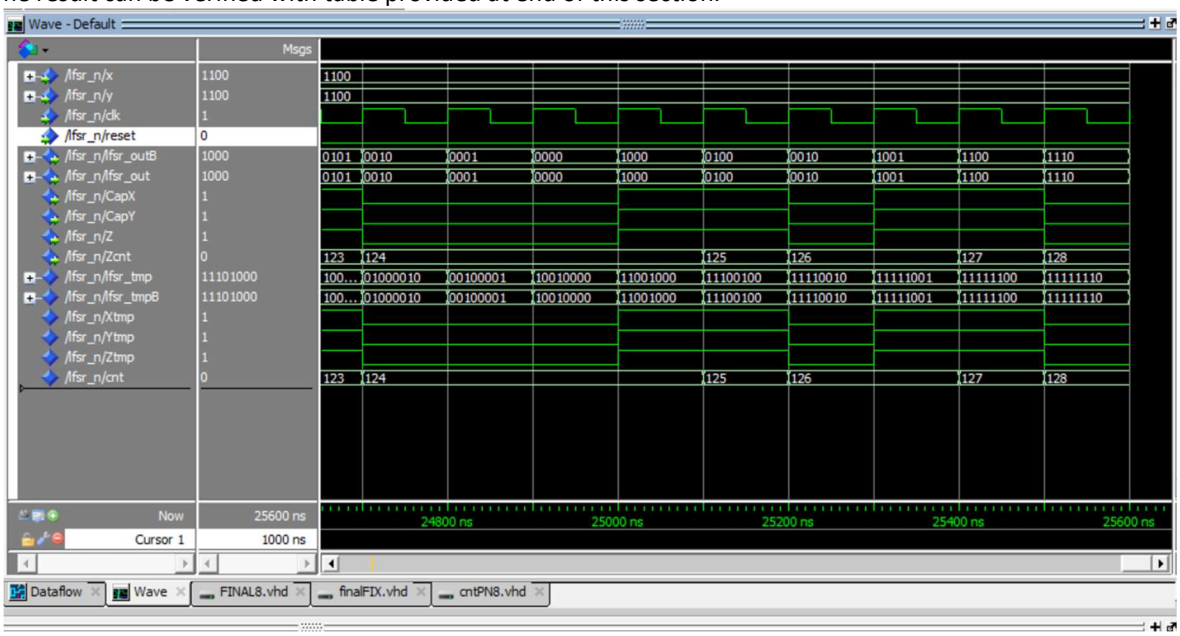


Fig.7.4 Simulation for x=0.1100 and y=0.1100; Zcnt=128

b) When input is x=0.1100 and y=0.1000 in binary

Z shows the graph of result after multiplication

The number of 1s in the Z is show through a count recorded through counter

Zcnt = 64 (Final Value in Graph)

The result can be verified with table provided at end of this section.

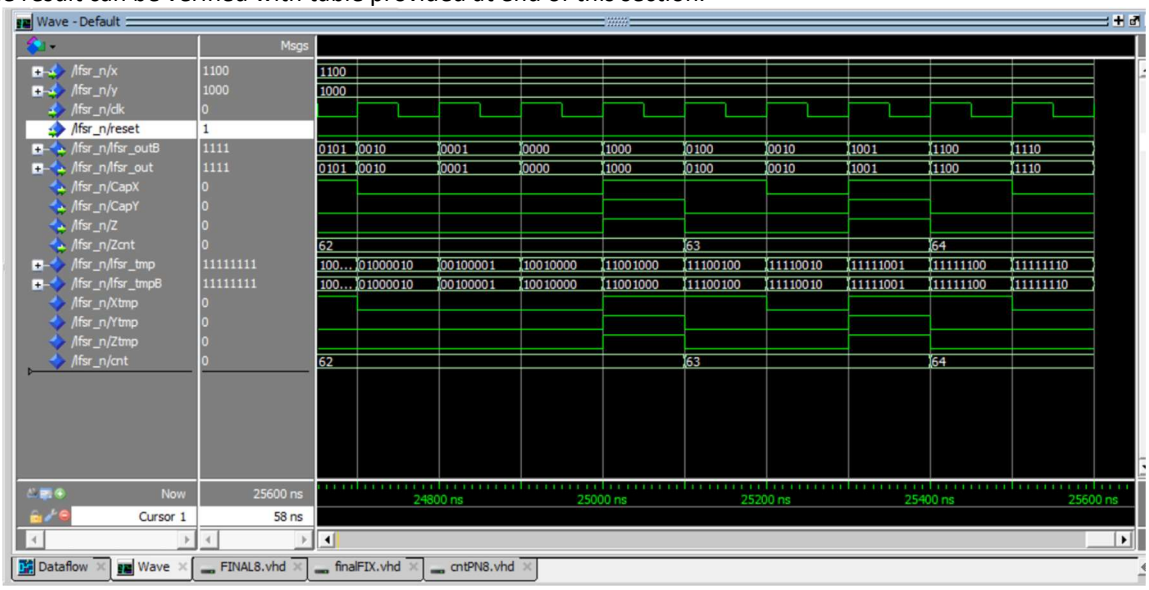


Fig.7.5 Simulation for x=0.1100 and y=0.1000; Zcnt=64

7.5 Synthesis of Circuit

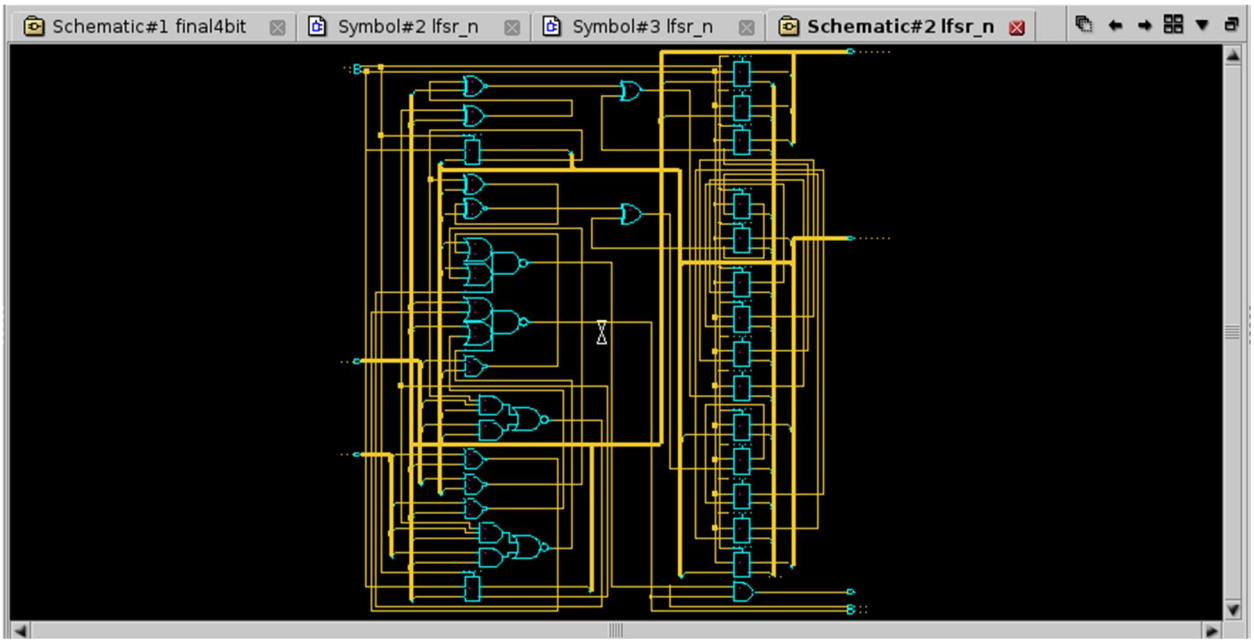


Fig.7.6 Circuit obtained from 8-bit code

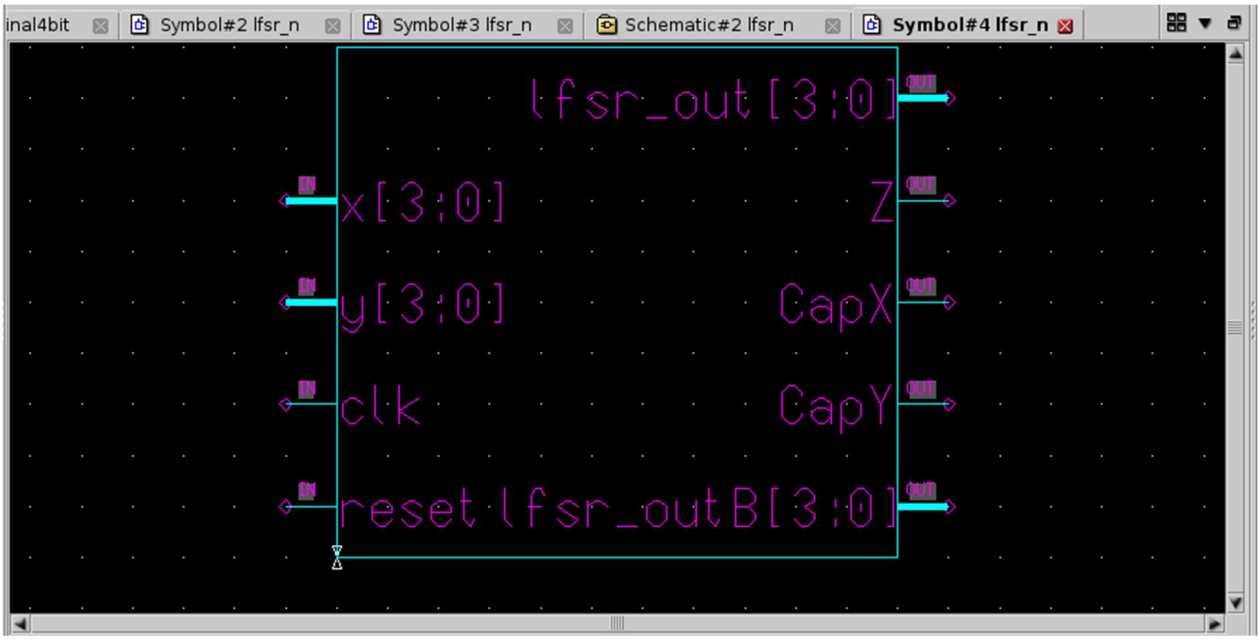


Fig.7.7 Symbol Circuit

7.6 Layouts

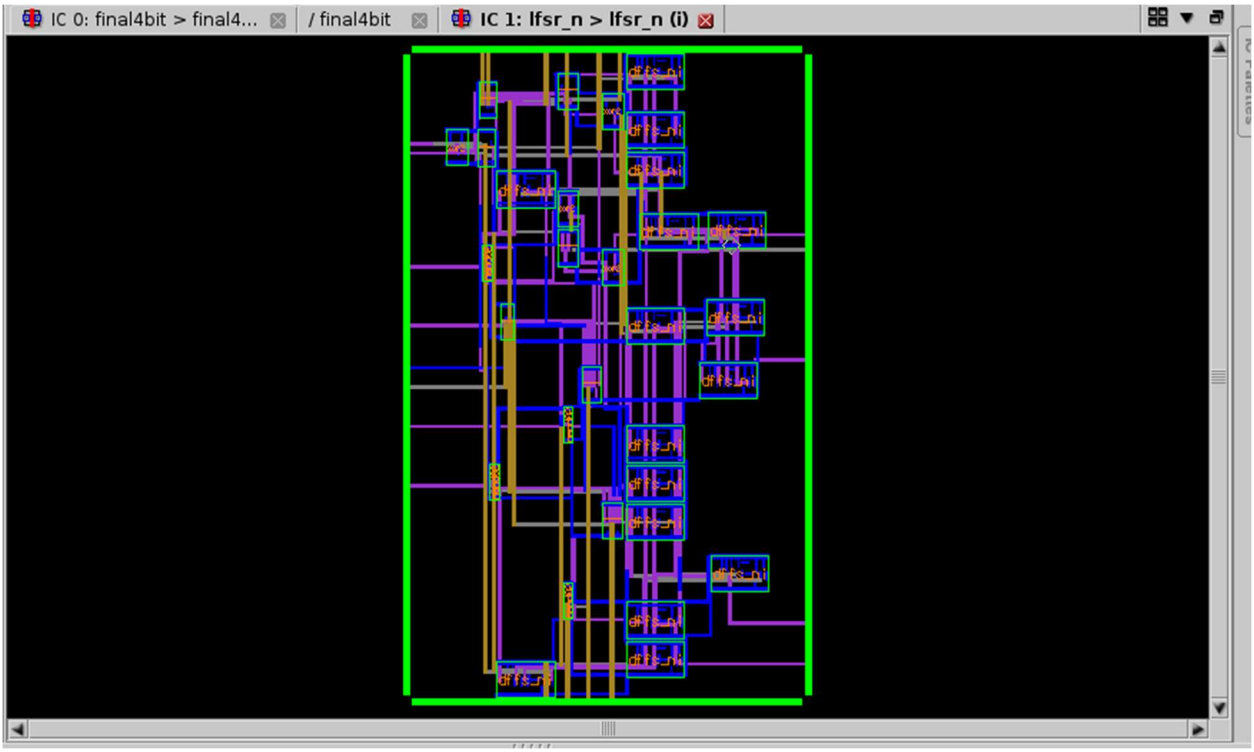


Fig.7.8 Layout Generated Using Auto Routing

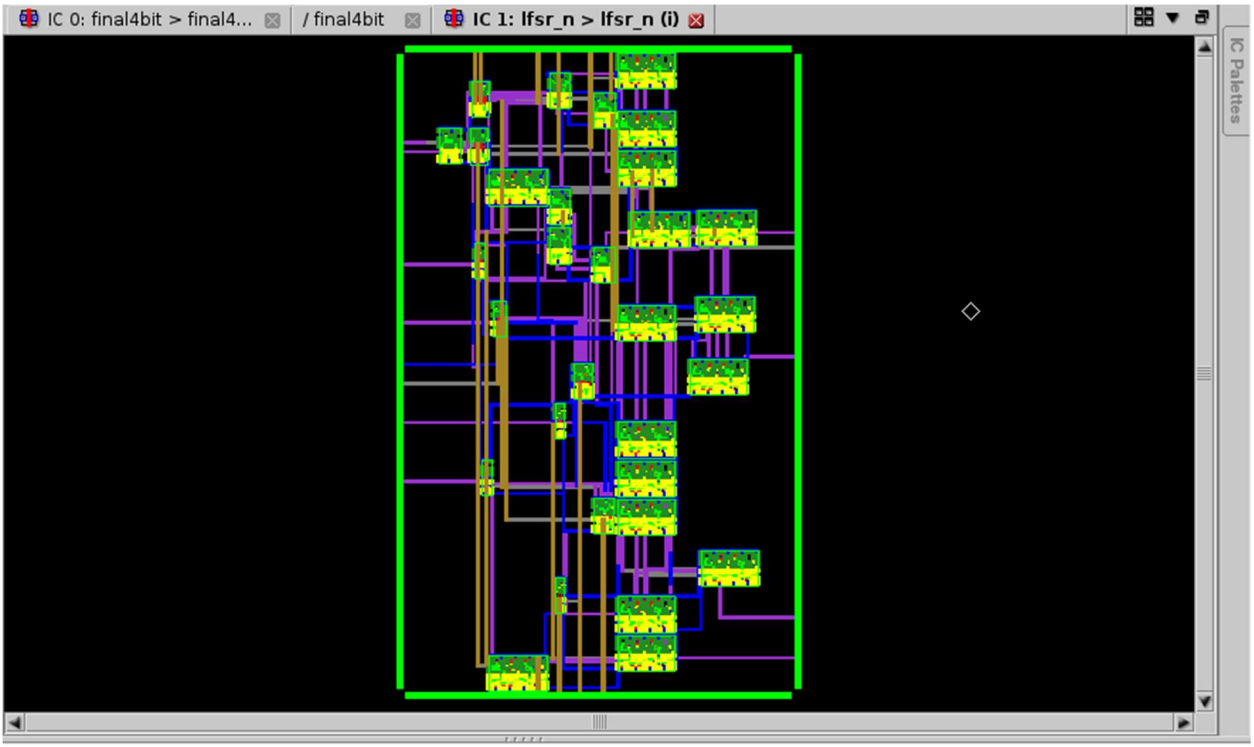


Fig.7.9 Layout Showing Peek Area

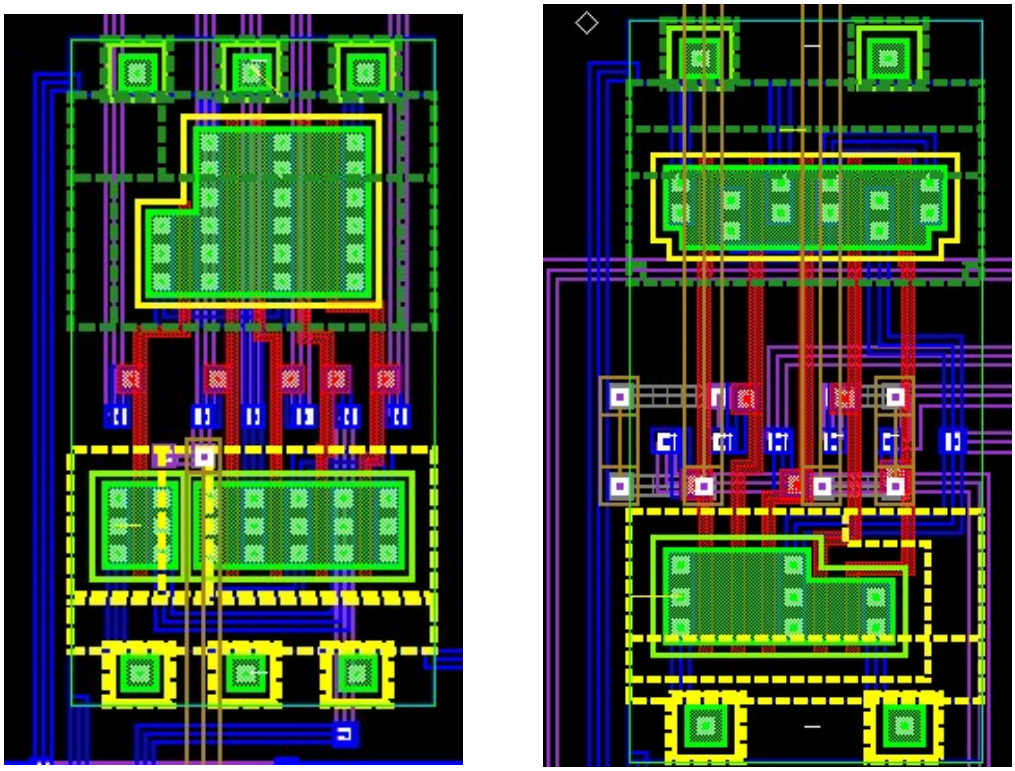


Fig.7.10 Zoomed Layout of Different Components

```
                ##                                                ##
                ##        C A L I B R E   S Y S T E M             ##
                ##                                                ##
                ##            L V S   R E P O R T                 ##
                ##                                                ##
                ####################################################


REPORT FILE NAME:          /afs/cad/u/s/k/sk2339/.MGC/lvs.rep
LAYOUT NAME:               /afs/cad/u/s/k/sk2339/ece758/project/0508/8bit/layout/lfsr_n
SOURCE NAME:               /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/8bit/OUTPUT/lfsr_n/sdl
RULE FILE:                 /afs/cad/sw.common/mentor.2012/adk3_1/technology/ic/process/tsmc035.rules
LVS MODE:                  Mask
RULE FILE NAME:            /afs/cad/sw.common/mentor.2012/adk3_1/technology/ic/process/tsmc035.rules
CREATION TIME:             Wed May 10 01:30:04 2017
CURRENT DIRECTORY:         /afs/cad.njit.edu/u/s/k/sk2339/.MGC
USER NAME:                 sk2339

*****************************************************************************************************************
                          OVERALL COMPARISON RESULTS
*****************************************************************************************************************

                     #       ####################       _   _
                   #         #                  #       *   *
                 #   #       #     CORRECT      #         |
                  # #        #                  #       \___/
                   #         ####################
```

Fig7.11 LVS Check Successful

7.7 Calculations and Results

X	1100	1100	1100	1000	0100	0100	1000	0100
Y	1100	1000	0100	1100	1100	1000	0100	0100
Z	128/225	64/255	64/255	80/255	64/255	0	0	64/255
ACCURATE	0.56	0.375	0.1875	0.375	0.1875	0.125	0.125	0.375
8-BIT	0.5	0.25	0.25	0.313	0.25	0	0	0.25

Table 4 Calculations and results from 8-bit Multiplier

8. Comparison of 4 bit PN Sequence Multiplier and 8 bit PN sequence Multiplier

X	11	11	11	10	01	01	10	01
Y	11	10	01	11	11	10	01	01
Z	9/15	3/15	6/15	4/15	6/15	2/15	2/15	4/15
ACCURATE	0.56	0.375	0.1875	0.375	0.1875	0.125	0.125	0.375
4-BIT	0.6	0.2	0.4	0.27	0.4	0.13	0.13	0.27
8-BIT	0.5	0.25	0.25	0.25	0.25	0	0	0.25
Difference4	0.04	0.175	0.2125	0.105	0.2125	0.005	0.005	0.105
Difference8	0.06	0.125	0.0625	0.125	0.0625	0.125	0.125	0.125

Table.5Comparison of results from 4 bit and 8 bit

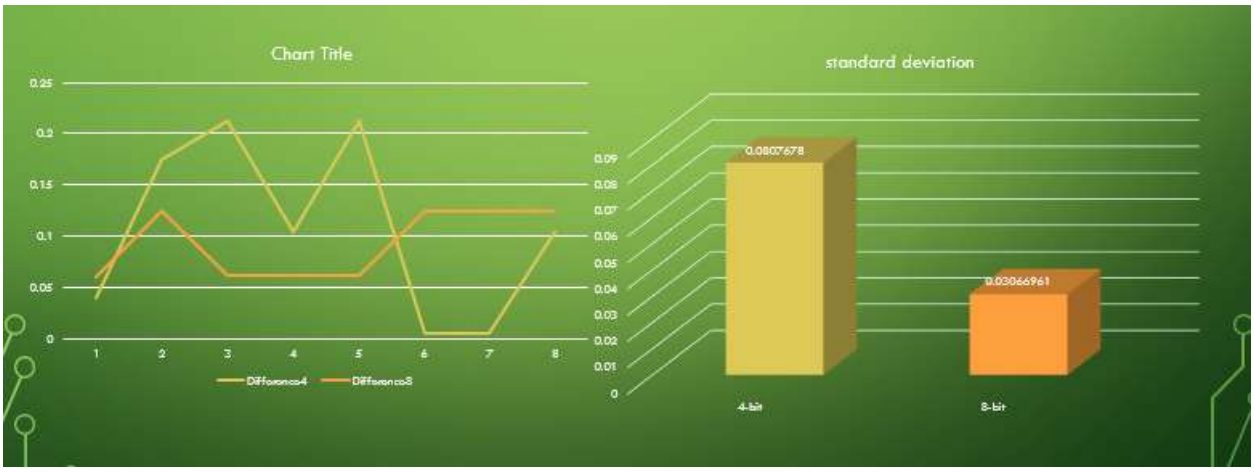


Fig.8.1 Graphs showing the difference in results and standard deviation

We have provided the output comparison of results in form of table and to show the difference is less in 8 bits we have plotted the graph which distinctly shows the more accuracy in 8 bit.

9. Conclusion

- From the results, it is concluded that accuracy increases as the number of bits in the multiplier increases.
- In summary, we have presented a sparse unary coding technique for binary numbers with built-in fault tolerance and the property that the product can be computed by the use of simple logic gates.

Appendix

LVS REPORT OF 4 BIT PN SEQUENCE MULTIPLIER

```
        ################################################
        ##                            ##
        ##      C A L I B R E   S Y S T E M       ##
        ##                            ##
        ##         L V S   R E P O R T          ##
        ##                            ##
        ################################################
```

REPORT FILE NAME: /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/output/layout/lvs.rep
LAYOUT NAME: /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/output/layout/lfsr_n
SOURCE NAME: /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/output/lfsr_n/sdl
RULE FILE: /afs/cad/sw.common/mentor.2012/adk3_1/technology/ic/process/tsmc035.rules
LVS MODE: Mask
RULE FILE NAME: /afs/cad/sw.common/mentor.2012/adk3_1/technology/ic/process/tsmc035.rules
CREATION TIME: Mon May 8 18:11:18 2017
CURRENT DIRECTORY: /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/output/layout
USER NAME: sk2339

****
OVERALL COMPARISON RESULTS
****

```
          #     ###################     _ _
         #      #           #     * *
       # #      #    CORRECT    #      |
        ##      #           #     \___/
         #      ###################
```

INITIAL NUMBERS OF OBJECTS

	Layout	Source	Component Type
Ports:	15	15	

```
 Nets:           547       190    *

 Instances:      172       172        mn (4 pins)
                 172       172        mp (4 pins)
                ------    ------
 Total Inst:     344       344


NUMBERS OF OBJECTS AFTER TRANSFORMATION
---------------------------------------

              Layout    Source        Component Type
              ------    ------        --------------
 Ports:          15        15

 Nets:          109       109

 Instances:      88        88        mn (4 pins)
                 92        92        mp (4 pins)
                 40        40        SUP2 (3 pins)
                 39        39        SMN2 (4 pins)
                  2         2        SPMN_2_1 (5 pins)
                ------    ------
 Total Inst:     261       261


      * = Number of objects in layout different from number in source.



*********************************************************************************************
****************
                          LVS PARAMETERS
*********************************************************************************************
****************


o LVS Setup:

  LVS COMPONENT TYPE PROPERTY          phy_comp element comp
  LVS COMPONENT SUBTYPE PROPERTY       model
  LVS PIN NAME PROPERTY                phy_pin
  LVS POWER NAME                     "VDD"
  LVS GROUND NAME                    "VSS"
  LVS CELL SUPPLY                   NO
  LVS RECOGNIZE GATES                 ALL
  LVS IGNORE PORTS                  NO
  LVS CHECK PORT NAMES                NO
  LVS IGNORE TRIVIAL NAMED PORTS      NO
  LVS BUILTIN DEVICE PIN SWAP         YES
  LVS ALL CAPACITOR PINS SWAPPABLE    NO
  LVS DISCARD PINS BY DEVICE          NO
  LVS SOFT SUBSTRATE PINS             NO
  LVS INJECT LOGIC                  YES
  LVS EXPAND UNBALANCED CELLS         YES
```

```
LVS EXPAND SEED PROMOTIONS            NO
LVS PRESERVE PARAMETERIZED CELLS      NO
LVS GLOBALS ARE PORTS                 YES
LVS REVERSE WL                        NO
LVS SPICE PREFER PINS                 NO
LVS SPICE SLASH IS SPACE              YES
LVS SPICE ALLOW FLOATING PINS         YES
// LVS SPICE ALLOW INLINE PARAMETERS
LVS SPICE ALLOW UNQUOTED STRINGS      NO
LVS SPICE CONDITIONAL LDD             NO
LVS SPICE CULL PRIMITIVE SUBCIRCUITS  NO
LVS SPICE IMPLIED MOS AREA            NO
// LVS SPICE MULTIPLIER NAME
LVS SPICE OVERRIDE GLOBALS            NO
LVS SPICE REDEFINE PARAM              NO
LVS SPICE REPLICATE DEVICES           NO
LVS SPICE SCALE X PARAMETERS          NO
LVS SPICE STRICT WL                   NO
// LVS SPICE OPTION
LVS EDDM PROCESS M                    NO
LVS STRICT SUBTYPES                   NO
LVS EXACT SUBTYPES                    NO
LAYOUT CASE                           NO
SOURCE CASE                           NO
LVS COMPARE CASE                      NO
LVS DOWNCASE DEVICE                   NO
LVS REPORT MAXIMUM                    50
LVS PROPERTY RESOLUTION MAXIMUM       32
// LVS SIGNATURE MAXIMUM
// LVS FILTER UNUSED OPTION
// LVS REPORT OPTION
LVS REPORT UNITS                      YES
// LVS NON USER NAME PORT
// LVS NON USER NAME NET
// LVS NON USER NAME INSTANCE

// Reduction

LVS REDUCE SERIES MOS                 NO
LVS REDUCE PARALLEL MOS               YES
LVS REDUCE SEMI SERIES MOS            NO
LVS REDUCE SPLIT GATES                YES
LVS REDUCE PARALLEL BIPOLAR           YES
LVS REDUCE SERIES CAPACITORS          YES
LVS REDUCE PARALLEL CAPACITORS        YES
LVS REDUCE SERIES RESISTORS           YES
LVS REDUCE PARALLEL RESISTORS         YES
LVS REDUCE PARALLEL DIODES            YES
LVS REDUCTION PRIORITY                PARALLEL

// Filter

LVS FILTER  sch_filter_direct_open  OPEN SOURCE DIRECT
LVS FILTER  sch_filter_direct_short  SHORT SOURCE DIRECT
LVS FILTER  sch_filter_mask_open  OPEN SOURCE MASK
LVS FILTER  sch_filter_mask_short  SHORT SOURCE MASK
```

```
LVS FILTER  lay_filter_direct_open  OPEN LAYOUT DIRECT
LVS FILTER  lay_filter_direct_short  SHORT LAYOUT DIRECT
LVS FILTER  v  OPEN
LVS FILTER  i  OPEN
LVS FILTER  e  OPEN
LVS FILTER  f  OPEN
LVS FILTER  g  OPEN

// Trace Property

// TRACE PROPERTY  mn  instpar(w) width w 0
// TRACE PROPERTY  mp  instpar(w) width w 0
// TRACE PROPERTY  me  instpar(w) width w 0
// TRACE PROPERTY  md  instpar(w) width w 0
// TRACE PROPERTY  mn  instpar(l) length l 0
// TRACE PROPERTY  mp  instpar(l) length l 0
// TRACE PROPERTY  me  instpar(l) length l 0
// TRACE PROPERTY  md  instpar(l) length l 0
// TRACE PROPERTY  r  instpar(r) resistance r 0
// TRACE PROPERTY  c  instpar(c) capacitance c 0
// TRACE PROPERTY  d  instpar(a) area a 0
// TRACE PROPERTY  d  instpar(p) perimeter p 0



********************************************************************************************
*****************
                    INFORMATION AND WARNINGS
********************************************************************************************
*****************


             Matched    Matched    Unmatched    Unmatched    Component
              Layout     Source       Layout       Source     Type
             -------    -------    ---------    ---------    ---------
  Ports:          15         15           0            0

  Nets:          109        109           0            0

  Instances:      88         88           0            0    mn(nmos4)
                  92         92           0            0    mp(pmos4)
                  40         40           0            0    SUP2
                  39         39           0            0    SMN2
                   2          2           0            0    SPMN_2_1
             -------    -------    ---------    ---------
  Total Inst:    261        261           0            0


o Statistics:

  357 isolated layout nets were deleted.


o Isolated Layout Nets:

    (Layout nets which are not connected to any instances or ports).
```

112(-177.950,1189.700) 113(-167.950,1189.700) 114(-159.450,1189.700) 115(-154.450,1189.700)
116(-145.950,1189.700) 117(-133.450,1189.700) 118(-124.950,1189.700) 119(-108.450,1189.700)
120(-100.450,1189.700) 121(-84.450,1180.700) 122(-79.450,1180.700) 123(-70.950,1180.700)
124(-62.950,1195.700) 125(-57.950,1195.700) 126(-49.950,1195.700) 127(-29.450,1189.700)
128(-13.950,1189.700) 129(-11.250,597.050) 130(-2.750,597.050) 131(2.250,597.050)
132(10.250,597.050) 133(12.800,853.400) 134(12.800,1373.800) 135(15.250,597.050)
136(21.800,866.400) 137(21.800,1386.800) 138(23.750,597.050) 139(24.050,216.900)
140(24.050,347.000) 141(30.300,853.400) 142(30.300,1373.800) 143(32.050,216.900)
144(32.050,347.000) 145(35.300,853.400) 146(35.300,1373.800) 147(40.550,216.900)
148(40.550,347.000) 149(43.800,853.400) 150(43.800,1373.800) 151(52.300,853.400)
152(52.300,1373.800) 153(88.650,527.150) 154(88.650,657.250) 155(88.650,787.350)
156(89.900,397.050) 157(89.900,917.450) 158(89.900,1307.750) 159(94.750,1134.350)
160(94.900,397.050) 161(94.900,917.450)

o Initial Correspondence Points:

Ports: VDD CapX Z clk reset CapY lfsr_out[0] x[0] y[0] lfsr_outB[0] lfsr_out[1] lfsr_outB[1] x[1] y[1] GND
Nets: lfsr_tmp_0 $dummy[1] lfsr_tmp_1 nx18 nx46 nx28 nx0 lfsr_tmpB_1 lfsr_tmpB_0 $dummy[4] $dummy[3] $dummy[5] $dummy[7] $dummy[6] $dummy[2] $dummy[0]

****
SUMMARY
****

Total CPU Time: 0 sec
Total Elapsed Time: 0 sec

LVS REPORT OF 8 BIT PN SEQUENCE MULTIPLIER

```
##################################################
##                                              ##
##      C A L I B R E   S Y S T E M     ##
##                                              ##
##        L V S   R E P O R T         ##
##                                              ##
##################################################
```

REPORT FILE NAME: /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/8bit/layout/lvs.rep
LAYOUT NAME: /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/8bit/layout/lfsr_n
SOURCE NAME: /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/8bit/OUTPUT/lfsr_n/sdl
RULE FILE: /afs/cad/sw.common/mentor.2012/adk3_1/technology/ic/process/tsmc035.rules
LVS MODE: Mask
RULE FILE NAME: /afs/cad/sw.common/mentor.2012/adk3_1/technology/ic/process/tsmc035.rules
CREATION TIME: Mon May 8 20:46:52 2017
CURRENT DIRECTORY: /afs/cad.njit.edu/u/s/k/sk2339/ece758/project/0508/8bit/layout
USER NAME: sk2339

```
********************************************************************************************************
****************
                    OVERALL COMPARISON RESULTS
********************************************************************************************************
****************


              #     ###################     _  _
             #      #           #     *  *
           #  #      #    CORRECT    #      |
           ##       #           #     \___/
            #       ###################
```

INITIAL NUMBERS OF OBJECTS

	Layout	Source	Component Type
Ports:	23	23	
Nets:	1103	381 *	
Instances:	353	353	mn (4 pins)
	353	353	mp (4 pins)
Total Inst:	706	706	

NUMBERS OF OBJECTS AFTER TRANSFORMATION

	Layout	Source	Component Type
Ports:	23	23	
Nets:	214	214	
Instances:	133	133	mn (4 pins)
	227	227	mp (4 pins)
	58	58	SUP2 (3 pins)
	2	2	SPUP_3_2 (6 pins)
	93	93	SMN2 (4 pins)
	2	2	SMN3 (5 pins)
	6	6	SPMN_2_1 (5 pins)
	2	2	SPMN_2_2_1 (7 pins)
Total Inst:	523	523	

* = Number of objects in layout different from number in source.

****
LVS PARAMETERS
****

o LVS Setup:

```
LVS COMPONENT TYPE PROPERTY           phy_comp element comp
LVS COMPONENT SUBTYPE PROPERTY        model
LVS PIN NAME PROPERTY                 phy_pin
LVS POWER NAME                        "VDD"
LVS GROUND NAME                       "VSS"
LVS CELL SUPPLY                       NO
LVS RECOGNIZE GATES                   ALL
LVS IGNORE PORTS                      NO
LVS CHECK PORT NAMES                  NO
LVS IGNORE TRIVIAL NAMED PORTS        NO
LVS BUILTIN DEVICE PIN SWAP           YES
LVS ALL CAPACITOR PINS SWAPPABLE      NO
LVS DISCARD PINS BY DEVICE            NO
LVS SOFT SUBSTRATE PINS               NO
LVS INJECT LOGIC                      YES
LVS EXPAND UNBALANCED CELLS           YES
LVS EXPAND SEED PROMOTIONS            NO
LVS PRESERVE PARAMETERIZED CELLS      NO
LVS GLOBALS ARE PORTS                 YES
LVS REVERSE WL                        NO
LVS SPICE PREFER PINS                 NO
LVS SPICE SLASH IS SPACE              YES
LVS SPICE ALLOW FLOATING PINS         YES
// LVS SPICE ALLOW INLINE PARAMETERS
LVS SPICE ALLOW UNQUOTED STRINGS      NO
LVS SPICE CONDITIONAL LDD             NO
LVS SPICE CULL PRIMITIVE SUBCIRCUITS  NO
LVS SPICE IMPLIED MOS AREA            NO
// LVS SPICE MULTIPLIER NAME
LVS SPICE OVERRIDE GLOBALS            NO
LVS SPICE REDEFINE PARAM              NO
LVS SPICE REPLICATE DEVICES           NO
LVS SPICE SCALE X PARAMETERS          NO
LVS SPICE STRICT WL                   NO
// LVS SPICE OPTION
LVS EDDM PROCESS M                    NO
LVS STRICT SUBTYPES                   NO
LVS EXACT SUBTYPES                    NO
LAYOUT CASE                           NO
SOURCE CASE                           NO
LVS COMPARE CASE                      NO
LVS DOWNCASE DEVICE                   NO
LVS REPORT MAXIMUM                    50
LVS PROPERTY RESOLUTION MAXIMUM       32
```

```
// LVS SIGNATURE MAXIMUM
// LVS FILTER UNUSED OPTION
// LVS REPORT OPTION
LVS REPORT UNITS                YES
// LVS NON USER NAME PORT
// LVS NON USER NAME NET
// LVS NON USER NAME INSTANCE

// Reduction

LVS REDUCE SERIES MOS           NO
LVS REDUCE PARALLEL MOS         YES
LVS REDUCE SEMI SERIES MOS      NO
LVS REDUCE SPLIT GATES          YES
LVS REDUCE PARALLEL BIPOLAR     YES
LVS REDUCE SERIES CAPACITORS    YES
LVS REDUCE PARALLEL CAPACITORS  YES
LVS REDUCE SERIES RESISTORS     YES
LVS REDUCE PARALLEL RESISTORS   YES
LVS REDUCE PARALLEL DIODES      YES
LVS REDUCTION PRIORITY          PARALLEL

// Filter

LVS FILTER  sch_filter_direct_open  OPEN SOURCE DIRECT
LVS FILTER  sch_filter_direct_short  SHORT SOURCE DIRECT
LVS FILTER  sch_filter_mask_open  OPEN SOURCE MASK
LVS FILTER  sch_filter_mask_short  SHORT SOURCE MASK
LVS FILTER  lay_filter_direct_open  OPEN LAYOUT DIRECT
LVS FILTER  lay_filter_direct_short  SHORT LAYOUT DIRECT
LVS FILTER  v  OPEN
LVS FILTER  i  OPEN
LVS FILTER  e  OPEN
LVS FILTER  f  OPEN
LVS FILTER  g  OPEN

// Trace Property

// TRACE PROPERTY  mn  instpar(w) width w 0
// TRACE PROPERTY  mp  instpar(w) width w 0
// TRACE PROPERTY  me  instpar(w) width w 0
// TRACE PROPERTY  md  instpar(w) width w 0
// TRACE PROPERTY  mn  instpar(l) length l 0
// TRACE PROPERTY  mp  instpar(l) length l 0
// TRACE PROPERTY  me  instpar(l) length l 0
// TRACE PROPERTY  md  instpar(l) length l 0
// TRACE PROPERTY  r  instpar(r) resistance r 0
// TRACE PROPERTY  c  instpar(c) capacitance c 0
// TRACE PROPERTY  d  instpar(a) area a 0
// TRACE PROPERTY  d  instpar(p) perimeter p 0



********************************************************************************
**********
                    INFORMATION AND WARNINGS
```

****
****

	Matched Layout	Matched Source	Unmatched Layout	Unmatched Source	Component Type
Ports:	23	23	0	0	
Nets:	214	214	0	0	
Instances:	133	133	0	0	mn(nmos4)
	227	227	0	0	mp(pmos4)
	58	58	0	0	SUP2
	2	2	0	0	SPUP_3_2
	93	93	0	0	SMN2
	2	2	0	0	SMN3
	6	6	0	0	SPMN_2_1
	2	2	0	0	SPMN_2_2_1
Total Inst:	523	523	0	0	

o Statistics:

722 isolated layout nets were deleted.

o Isolated Layout Nets:

(Layout nets which are not connected to any instances or ports).

227(-154.550,1902.200) 228(-145.550,1915.200) 229(-137.050,1902.200) 230(-132.050,1902.200)
231(-123.550,1902.200) 232(-115.050,1902.200) 233(-50.850,1922.200) 234(-48.150,2081.750)
235(-42.850,1922.200) 236(-40.150,2081.750) 237(-38.300,1534.100) 238(-34.850,1922.200)
239(-32.150,2081.750) 240(-30.300,1534.100) 241(-26.850,1922.200) 242(-24.150,2081.750)
243(-18.850,1922.200) 244(-16.150,2081.750) 245(-15.700,806.350) 246(-7.700,806.350)
247(9.800,136.850) 248(9.800,1763.150) 249(14.800,136.850) 250(14.800,1763.150)
251(23.300,136.850) 252(23.300,1763.150) 253(24.300,1337.550) 254(28.300,143.850)
255(28.300,1770.150) 256(32.300,1337.550) 257(36.800,136.850) 258(36.800,1763.150)
259(40.800,1337.550) 260(44.800,136.850) 261(44.800,1763.150) 262(55.800,158.850)
263(55.800,1785.150) 264(60.800,137.850) 265(60.800,1764.150) 266(68.800,137.850)
267(68.800,1764.150) 268(84.800,137.850) 269(84.800,1764.150) 270(92.800,137.850)
271(92.800,1764.150) 272(114.300,136.850) 273(114.300,1763.150) 274(119.300,166.850)
275(119.300,1793.150) 276(127.800,136.850)

o Initial Correspondence Points:

Ports: VDD Z clk reset CapX CapY lfsr_out[1] lfsr_out[2] lfsr_out[3] lfsr_out[0]
lfsr_outB[1] lfsr_outB[2] lfsr_outB[3] lfsr_outB[0] x[3] x[0] y[3] y[0] x[2]
x[1] y[2] y[1] GND
Nets: nx274 nx265 nx276 nx247 nx280 nx253 nx251 nx2 nx66 nx238 nx278 nx249 lfsr_tmp_3
lfsr_tmpB_3 nx4 nx68 lfsr_tmp_1 lfsr_tmpB_0 $dummy[0] $dummy[3] lfsr_tmp_2
$dummy[5] lfsr_tmp_0 $dummy[6] $dummy[12] $dummy[11] lfsr_tmpB_1 $dummy[8]
$dummy[13] $dummy[7] $dummy[1] lfsr_tmpB_2 $dummy[9] $dummy[10] $dummy[2]

```
$dummy[4]


********************************************************************************************
****************
                        SUMMARY
********************************************************************************************
****************

Total CPU Time:      0 sec
Total Elapsed Time:  0 sec
```

References

1.Binary Multiplication with PN Sequences. K. GUPTA AKD R. KUMARESAN2

2. DESIGN OF 8 BIT, 16 BIT AND 32 BITLFSR FOR PN SEQUENCE GENERATION USING VHDL Siddesh GaonkarM.Tech Scholar - VLSI Design and Embedded systems Department of Electronics and Communication Engineering NMAM Institute of Technology, Nitte, India siddeshgaonkar92@gmail.com
International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Special Issue 31(September, 2015), PP. 305-308
305 | P a g e

3. Stochastic and deterministic computing with pseudorandom sequences: Digital filtering and other application P. K. Gupta and R. Kumaresan,in Proc. IEEE Asilomar Circuits Syst. Conf., Pacific Grove, CA. Nov. 1986.

4.On Properties of PN Sequences Generated by LFSR – a Generalized Study and Simulation Modeling Afaq Ahmad, Sayyid Samir Al-Busaidi and Mufeed Juma Al-Musharafi Department of Electrical and Computer Engineering, College of Engineering, Sultan Qaboos University, P. O. Box 33, Postal Code 123; Muscat, Sultanate of Oman; afaq@squ.edu.om, albusaid@squ.edu.om,mufeed03@hotmail.com

www.indjst.org | Vol 6 (10) | October 2013 Indian Journal of Science and Technology | Print ISSN: 0974-6846 | Online ISSN: 0974-5645