1. INTRODUCTION

A sequence of numbers $\{b_k\} = b_0 b_1 \dots \dots \dots b_{n-1}$ is a binary sequence if $\forall\, i = 0,1, \dots \dots \dots n-1$, $b_i \in \{0,1\}$. However for $\{b_k\}$ to be a pseudo-random sequence it needs to satisfy some special properties which would be discussed later in great detail. Pseudo-random Noise(PN) is a randomized sequence of binary numbers. The name Pseudo (from Greek *pseudēs* 'false') arises from the fact that the sequence is predictable; however, the prediction becomes more and more difficult with increase in sequence length and complexity of feedback logic. It is also called as maximal length sequence generator concerning the fact that they produce every binary sequence except the locked state.PN Sequence generators have been in use since the beginning of World War 2 especially for the security and encryption level it provides and while in terms of contemporary technology it has got major applications in communication engineering.

2. CONCEPT BEHIND PN SEQUENCE GENERATOR

2.1 Linear Feedback Shift Registers (LFSRs)

Any register whose input bit is a linear function of its previous state is defined as a Linear Function Shift Register. Every shift register contains a series of D-flip flops which are capable of storing a single bit during the interval of a single clock pulse. So an n-bit shift register shall generate a PN sequence of length $2^n - 1$. The 2^nth bit would be the same as the first bit of the sequence.

2.2 Locked State

The circuitry of any PN sequence register contains a linear feedback path which typically a combinational logic combining outputs of the last flip flop (and at times outputs of intermediate flip flops) which are feedback as input to the first flip flop. Hence, every combinational circuitry produces exactly one state which never occurs in the entire logic table of the PN sequence generator.(Hence $2^n - 1$ instead of 2^n states). However, by mistake if the designer initializes an LFSR with the "lock state" for its circuit, the same state will repeat infinitely and will no more remain a PN sequence.

2.3 Logic Diagram

There are very large number of distinct sequences which can be obtained from a standard LFSR. For an n-bit shift register the number of distinct sequences that can be obtained is equal to the number of Boolean functions that can be made from n distinct variables which is simply 2^{2^n}. Following is a general circuitry of the same. *In general, the consecutive outputs of the last flip flop form the designated sequence.*

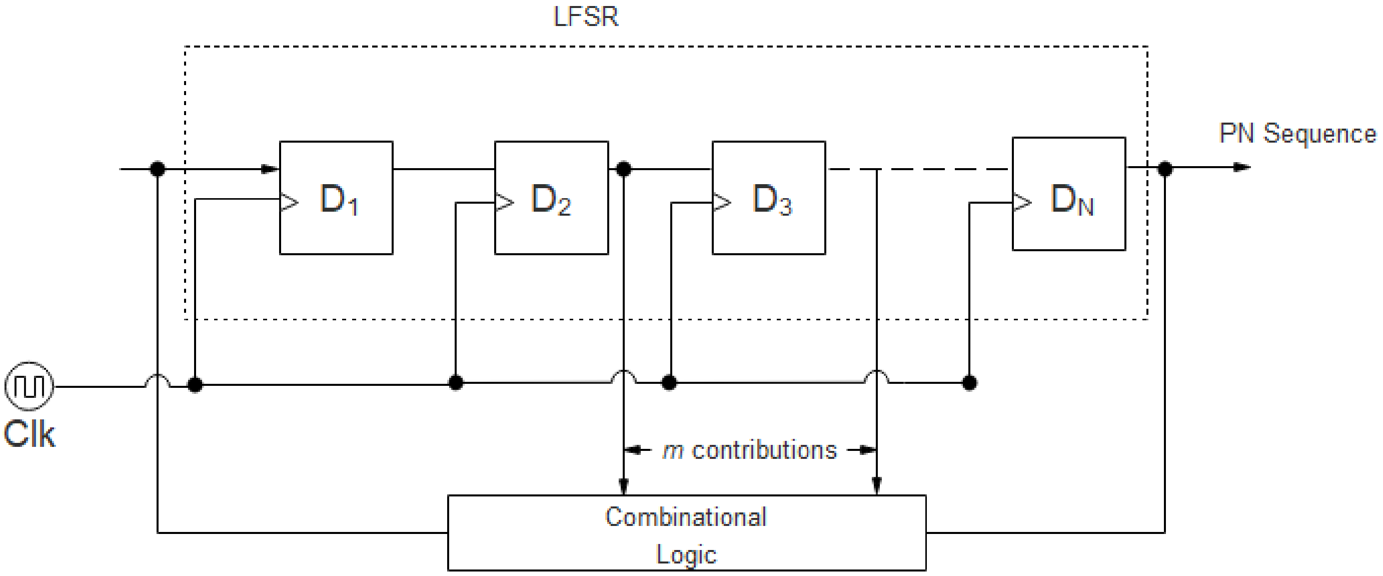


Figure 2.1 *A general Linear Feedback Shift Register consisting of a feedback path which has m contributions from intermediate flip flops passed through a combinational circuit*

3. 4-BIT PN SEQUENCE GENERATOR

3.1 Circuit Diagram

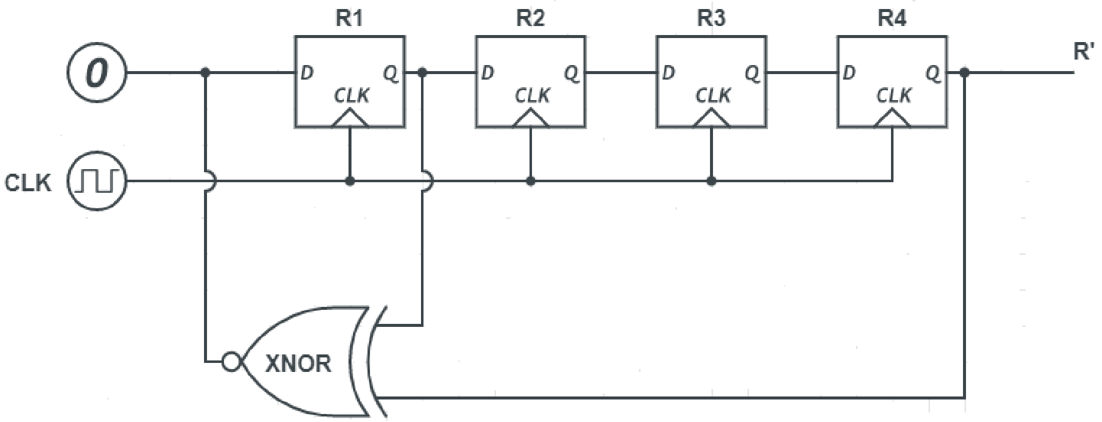


Figure 3.1 *A 4-bit PN sequence generator having an XNOR gate to form the feedback path*

3.2 Logic table, Lock state and State equation

Following is the logic table of the previously illustrated 4-bit PN sequence generator. Let Q_{1_n},Q_{2_n},Q_{3_n} and Q_{4_n} denote the present state of output of the successive flip flops. Let $D_{1_{n+1}}$ denote the next state of input to the first flip flop. Then the state equation is given by

$$\boxed{D_{1_{n+1}} = Q_{1_n} \odot Q_{4_n}}$$

Table 3.1 *Logic Table for 4-bit PN Sequence Generator with the highlighted column being the desired sequence*

CLK Pulse	Q_1	Q_2	Q_3	Q_4	Y
1	0	0	0	0	0
2	1	0	0	0	0
3	0	1	0	0	0
4	1	0	1	0	0
5	0	1	0	1	0
6	0	0	1	0	1
7	1	0	0	1	0
8	1	1	0	0	1
9	0	1	1	0	0
10	1	0	1	1	0
11	1	1	0	1	1
12	1	1	1	0	1
13	0	1	1	1	0
14	0	0	1	1	1
15	0	0	0	1	1

It is clearly visible that the LOCK state for the above sequence generator is “1111”. Initialising the logic table with this state would simply create an infinite sequence of 1s.

3.3 VHDL Implementation

3.3.1 VHDL code

The VHDL code was designed and developed in Xilinx V9.2. It consists of a single top level module named as top_module.vhd which contains the core program code with two files placed inside it namely EX_nor.vhd and logic.vhd which are the VHDL files corresponding to a standard XNOR gate and a positive edge triggered D-flip flop with synchronous active high reset mechanism respectively.

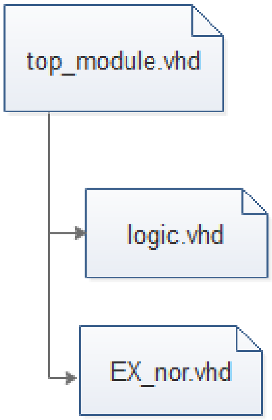


Figure 3.2 *File Hierarchy*

EX_nor.vhd

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

entity EX_nor is
Port ( a,b : in STD_LOGIC;
         y : out STD_LOGIC);
end EX_nor;

architecture Behavioral of EX_nor is

begin

 y <= a xnor b;

end Behavioral;
```

logic.vhd

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

entity logic is
 Port ( D,clk,reset : in STD_LOGIC;
                  Q : out STD_LOGIC);
end logic;
```

```
architecture Behavioral of logic is

begin

process (clk,reset)
begin
if clk'event and clk='1' then
   if reset='1' then Q <='0';
   else Q <= D;
   end if;
end if;
end process;

end Behavioral;
```

top_module.vhd

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

entity top_module is
  Port ( clk1,rst : in STD_LOGIC;
            pn_out: out STD_LOGIC);
end top_module;

architecture hierarchy of top_module is

component logic port ( D,clk,reset : in STD_LOGIC; --Component representing D flip flop
                                 Q : out STD_LOGIC);
end component;

component EX_nor port ( a,b : in STD_LOGIC;     --Component representing XNOR gate
                          y : out STD_LOGIC);
end component;

Signal Q1,Q2,Q3,Q4: STD_LOGIC;     --Intermediate outputs from individual flip flops
Signal x: STD_LOGIC;

begin

D1:logic port map(x,clk1,rst,Q1); --Cascading the required flip flops to form an LFSR
D2:logic port map(Q1,clk1,rst,Q2);
D3:logic port map(Q2,clk1,rst,Q3);
D4:logic port map(Q3,clk1,rst,Q4);
G1:EX_nor port map(Q1,Q4,x);     --Giving feedback
pn_out <= Q4;

end hierarchy;
```

3.3.2 Simulation

The simulation was done on a standard test bench of size 1000 ns. Following are the simulation results. Its obvious from the simulation results that the sequence repeats after $15\ (2^4 - 1)$ clock cycles.

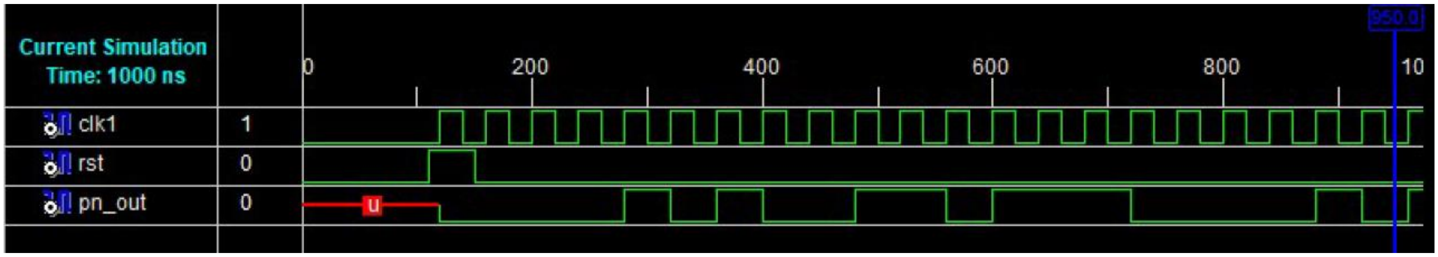


Figure 3.3 *Simulation results obtained from a 4- bit PN Sequence Generator with pn_out being the desired sequence*

3.3.2 Synthesis

Standard synthesis tools were used in Xilinx and technology and RTL schematics of the various components and modules present were obtained. Following is a detailed illustration of each one of them.

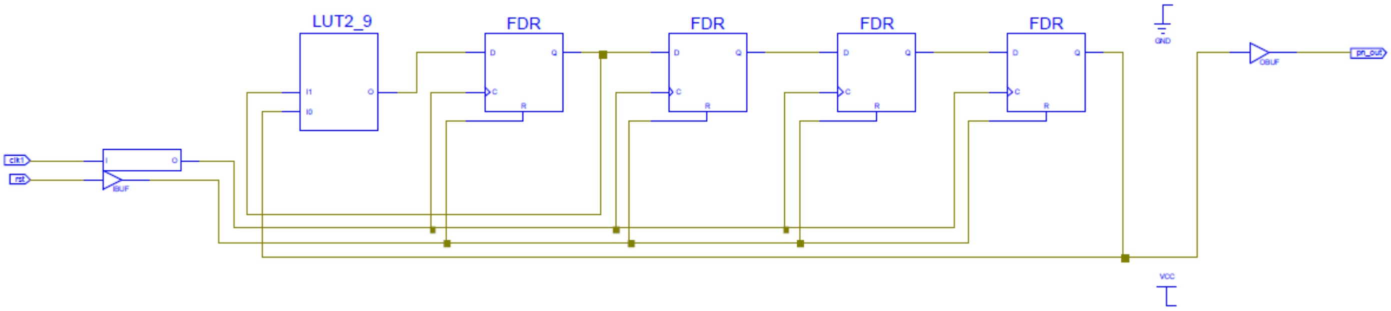


Figure ***3.4*** *Technology schematic of a 4-bit PN Sequence Generator*

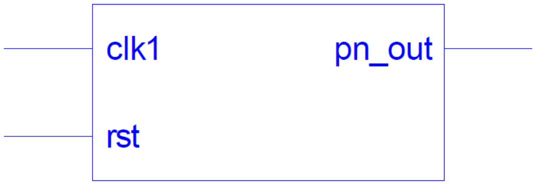


Figure 3.5.1.A *RTL Schematic of top level module*

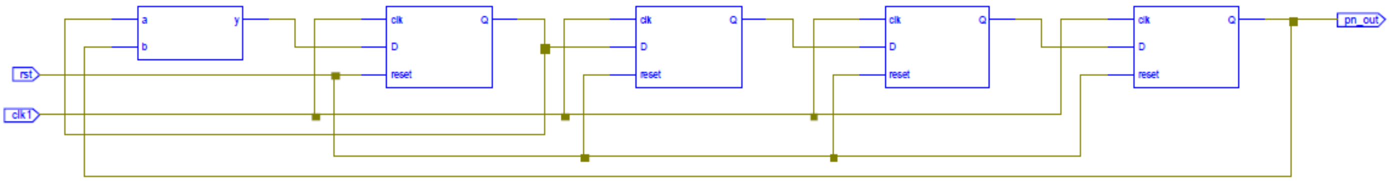


***Figure 3.5.1.B** Second level RTL schematic of top level module*

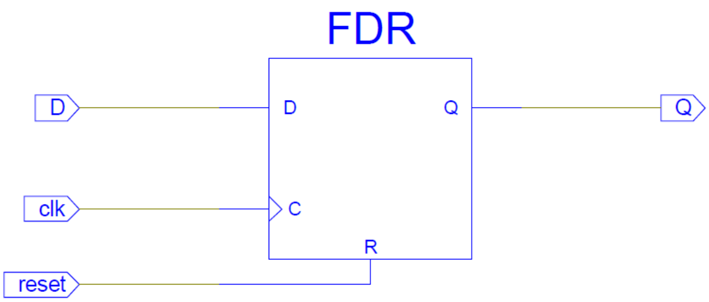


***Figure 3.5.2** RTL schematic of D flip flop*

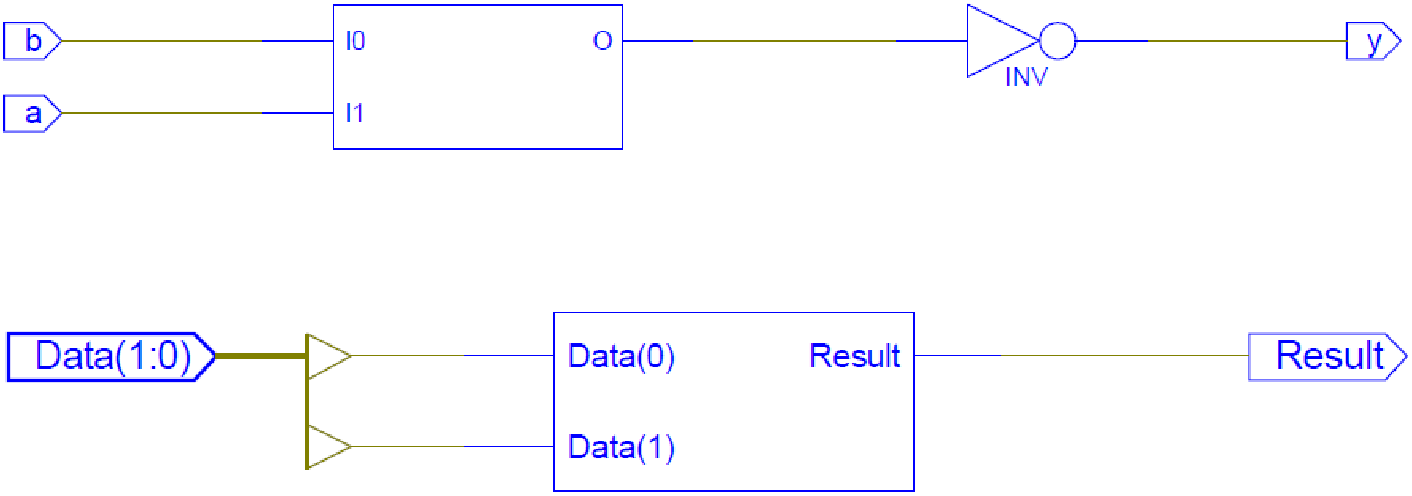


***Figure 3.5.3.A** and **Figure 3.5.3.B** (top to bottom) Level 1 and Level 2 RTL schematic of XNOR gate*

4. PROPERTIES OF PN SEQUENCE

As described in the introduction not every binary sequence $\{b_k\}$ is a random (or pseudo-random) sequence until and unless it satisfies some specific properties which are as follows.

For explaining the properties, we would be taking the output sequence of our own PN sequence generator i.e. **000010100110111**.

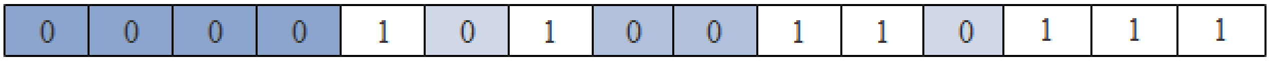
(a) **Balance**: *The no. of* $0s$ *and* $1s$ *should differ by atmost one*. Now for a given sequence let $N(0)$ and $N(1)$ denote the no. of 0s and 1s in the sequence respectively. Then we can say

$$\boxed{|N(0) - N(1)| \leq 1}$$

For the sequence 000010100110111 we have $N(0) = 8$ and $N(1) = 7$ and therefore we can say $|N(0) - N(1)| = |8 - 7| \leq 1$ clearly satisfies the given property and hence its a balanced sequence.

(b) **Run length property**: A run is defined as a sequence of single type of binary digits. Appearance of the other digit automatically starts a new run. Length of the run is defined as the no. of bits in that run and (here) would be denoted by $l(run_i^k)$ where run_i^k indicates the ith run in that sequence when read from left to right made up of the bit k where $k \in \{0,1\}$.

Now consider the sequence 000010100110111 which has the following set of runs as illustrated below.



0	0	0	0	1	0	1	0	0	1	1	0	1	1	1

Figure 4.1 *Sequence with runs' of zeroes shown in decreasing order of shade of blue in accordance with their lengths.*

The run length property states that $\frac{1}{2^m}$ *of the number of runs of each type (i.e. either* run_i^0 *or* run_i^1*) would be of length* m.

Written symbolically

$$\boxed{\frac{1}{2^m} \times \sum_{all\ i} run_i^k = N(\{run_i^k \mid l(run_i^k) = m\})}$$

We have for our sequence

$$\sum_{all\,i} N(\,run_i^0) = 4$$

Table 4.1 *Run count in PN sequence*

run_i^k	Illustration	$l(run_i^k)$	$N(run_i^k)$
run_3^0	0 0	1	$2 = \frac{1}{2^1} \times 4$
run_8^0	0 0	2	$1 = \frac{1}{2^2} \times 4$

Clearly the sequence satisfies the run length property as well.

(c) **Auto correlation**: This is the most important property which gives a quantitative expression for a sequence $c(n)$ with its rotated sequence by $\pm\tau$ places to the right denoted by $c(n-\tau)$. *The auto correlation property states that the auto correlation function given by*

$$R(\tau) = \frac{1}{N}\sum_n N(A) - N(D)$$

takes up only either of the two values i.e. 1 *or* $\frac{-1}{N}$ *depending on a zero or non-zero shift for* $c(n)$. Here N denotes the total number of bits in the sequence and $N(A)$ and $N(D)$ denote the no. of agreements and disagreements amongst the standard and rotated sequence respectively. Now consider our sequence being rotated by $\tau = 3$ places as illustrated below.

0	0	0	0	1	0	1	0	0	1	1	0	1	1	1
1	1	1	0	0	0	0	1	0	1	0	0	1	1	0

Figure 4.2 *Normal PN sequence* $c(n)$ *along with its rotated sequence* $c(n-3)$

From the above figure we have $N(A) = 7$ and $N(D) = 8$ (if similar elements occur at a given bit it is counted as a single agreement and a disagreement otherwise). Accordingly, we have $\sum_n N(A) - N(D) = -1$ and $R(\tau) = -\frac{1}{15}$ since $N = 15$ terms are there in the sequence. Similarly if $\tau = 0$, $N(A) = N$ and $N(D) = 0$ (since there would be no disagreements in a sequence when compared with itself) giving us $R(\tau) = -\frac{1}{N}$. Clearly the auto correlation property is also satisfied by our output. So, qualitatively speaking we have for any $c(n)$

$$\boxed{R(\tau) = \begin{cases} 1 & \tau = 0 \\ -\frac{1}{N} & \tau \neq 0 \end{cases}}$$

Hence our sequence satisfies all the properties for a standardized PN sequence.

5. APPLICATIONS

Applications of a PN sequence are numerous. Some of the most common ones have been mentioned below.

(a) GPS satellite systems

(b) Cellular communications in Code Division Multiple Access i.e. CDMA

(c) LAN/MAN/WAN internet connections

(d) Bluetooth connectivity

(e) Railroads and transportation

Each of the aforementioned applications arise due to the following needs:

(i) Anti-jamming capabilities

(ii) Secure communications

(iii) Low Probability of Detection(LPD) and Low Probability of Intercept(LPI)

(iv) Low Probability of Position Fix(LPPF)

(v) Multiple access