**Student Name:Shivam Sharma** **Roll No :62**

**Date of performance : 3/8/2023** **Date of Completion :** 08/08/2023

**Experiment No. - 03**

**Title-** PN Sequence generator

**Aim-** Design a PN Sequence Generator of length n

**Objective-** Students will understand concept of PN sequence generator and understand properties of PN sequence.

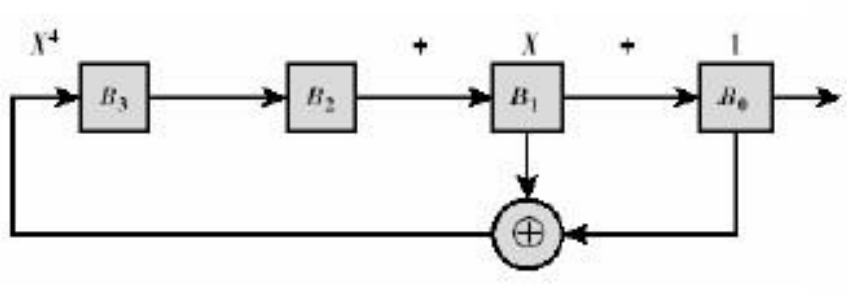
**Learning Outcome :** Students will to design PN sequence generator for given length. And find out autocorrelation property

**Theory :**A Pseudo-random Noise (PN) sequence is a sequence of binary numbers, e.g. ±1, which appears to be random; but is in fact perfectly deterministic.

A software or hardware device designed to produce a PN sequence is called a PN Generator.

Pseudo-random noise sequences or PN sequences are known Sequences that exhibit the properties or characteristics of random sequences. They can be used to logically isolate users on the same frequency channel. They can also be used to perform scrambling as well as spreading and dispreading functions. The reason we need to use PN sequences is that if the code sequences were deterministic, then everybody could access the channel. If the code sequences were truly random on the other hand, then nobody, including the intended receiver, would be able to access the channel

**Example of PN generator**




Here

here B3= B0 Xor B1
and polynomial is- 1+x+x^3

Table

| State | $B_3$ | $B_2$ | $B_1$ | $B_0$ | $B_0 \oplus B_1$ | output |
|---|---|---|---|---|---|---|
| Initial = 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 | 1 | 1 |
| 4 | 1 | 1 | 0 | 0 | 0 | 0 |
| 5 | 0 | 1 | 1 | 0 | 1 | 0 |
| 6 | 1 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 1 | 0 | 1 | 0 |
| 9 | 1 | 1 | 0 | 1 | 1 | 1 |
| 10 | 1 | 1 | 1 | 0 | 1 | 0 |
| 11 | 1 | 1 | 1 | 1 | 0 | 1 |
| 12 | 0 | 1 | 1 | 1 | 0 | 1 |
| 13 | 0 | 0 | 1 | 1 | 0 | 1 |
| 14 | 0 | 0 | 0 | 1 | 1 | 1 |
| 15 = 0 | 1 | 0 | 0 | 0 | 0 | 0 |

Output

**Properties of PN Sequence-**

The n-sequence has the following properties in every period of length L =( 2^n) – 1:

**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 = 2^n–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.

**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**
We define the term 'run' in a general way as a succession of items of the same class. In a period of PN sequence the distribution of sequential occurrences of groups of 1s, and 0s (runs property for $1 \leq k \leq n-2$ ), is governed by a rule and we present this in the form of the following theorem.
THEOREM 1. In every period ($p = 2n-1$) 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 $1 \leq k \leq n-2$ . We prove our stated properties (P1 to P6) and Theorem 1 by presenting an example as below.

**Property 8: P8 (The property of auto-correlation)**
To study the statistical property PN sequences, it is important to analyse them through their correlation functions. Correlation function of two sequences can be described as the comparison of two sequences to see how much they correspond with one another. Various parameters effect the correlation of two sequences including the length of sequence, phase between the sequences, and clock rate of LFSR. The act of correlating a signal through all variations of itself is known as autocorrelation. The autocorrelation $((s_i \in N) = S1, S2, ...Si.....)$ function, AC
(k) of an m-sequence where, $N = 1$ to $2n-1$ can be
given for its kth shift as:
$AC(k) = 1/N \sum S_{i}, S_{i+k}$

$0 \leq k \leq N-1$ where, si is the value of
the ith sequence.

In other words, the auto-correlation of a sequence is the special case of cross correlation. The auto-correlation of a sequence is correlation of a sequence with itself.The autocorrelation function R(x) provides a measure of how closely the signal matches a copy of itself as the copy is shifted units in time. The variable plays the role of a scanning or searching parameter. R(x) is not a function of time; it is only a function of the time difference between the waveform and its shifted copy.

Autocorrelation , also known as serial correlation, is the cross – correlation of a signal with itself. Informally, it is the similarity between observations as a function of the time lag between them. It is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise, or identifying themissing fundamental frequency in a signal implied by its harmonic frequencies. It is often used in signal processing for analyzing functions or series of values, such as time domain signals

The m-sequence has good autocorrelation property and is used in many applications including IS-95 CDMA system

**3. Cross correlation Property**

Cross correlation of two codes is the measure of agreement between two different codes. The m-sequences are not immune to cross correlation problem and they may have large crosscorrelation values

**Algorithm**

1. Start

- Input the No. Of stages (m) and initial status of the LFSR (Linear Feedback shift Register)
- Determine all the primitive polynomials for a particular value of m
- Depending upon the feedback given by primitive polynomial perform the modulo 2 addition
- Shift the bits of the LFSR circularly to get the next stage
- PN sequence is formed at the o/p of the 1st stage of LFSR.
- Stop

**Observation Table**

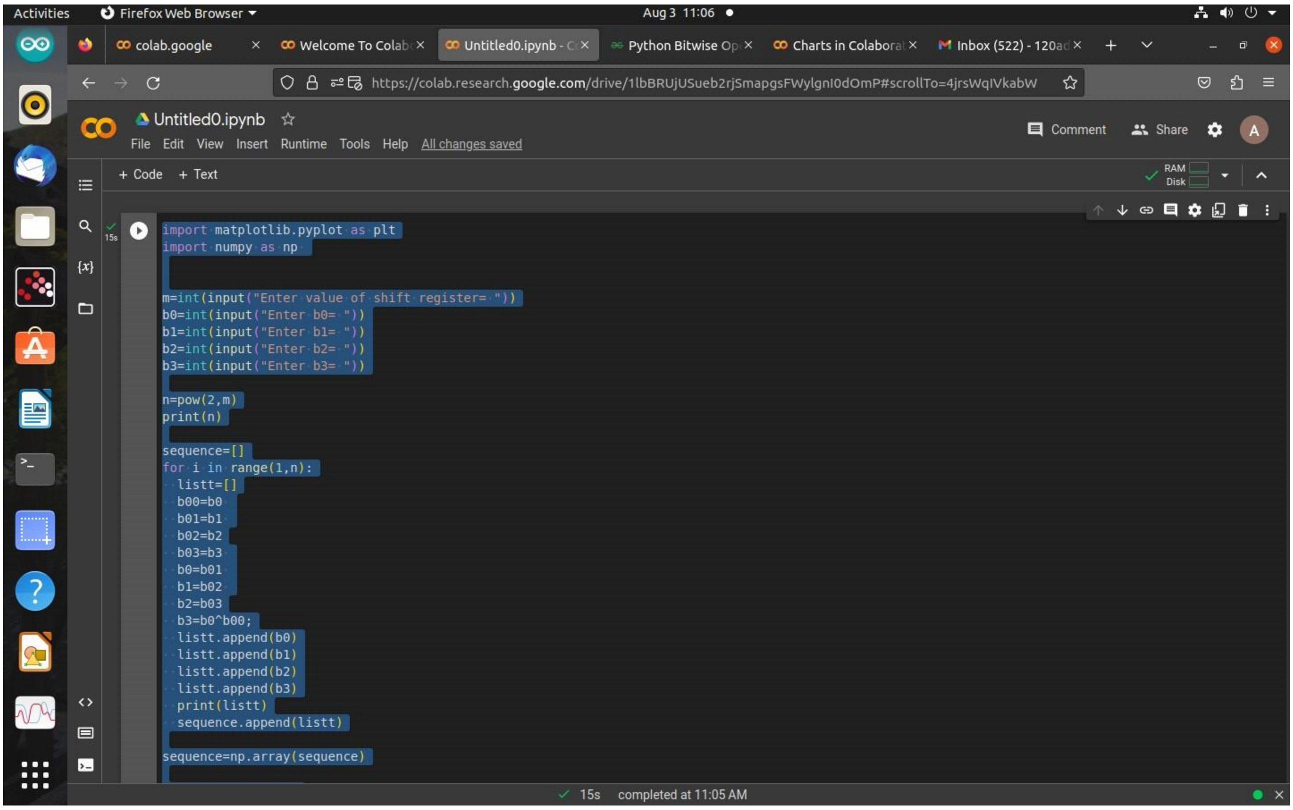
| Sr. No | No. of Runs | Sucession of | Comment |
|---|---|---|---|
| 1 | - | No. of 1s = | Property 1 verified |
| 2 | - | No. of 0s = | Property 1 verified |
| 3 | 1 | 4-1s | Property 3 verified |
| 4 | 0 | 4 – 0s | Property 4 verified |
| 5 | 0 | 3 – 1s | Property 5 verified |
| 6 | 1 | 3-0s | Property 6 verified |
| 7 | 1 | 2 – 1s | Property 7 verified |
| 8 | 1 | 2 – 0s | Property 7 verified |
| 9 | 2 | 1 – 1s | Property 7 verified |
| 10 | 2 | 1 – 0s | Property 7 verified |

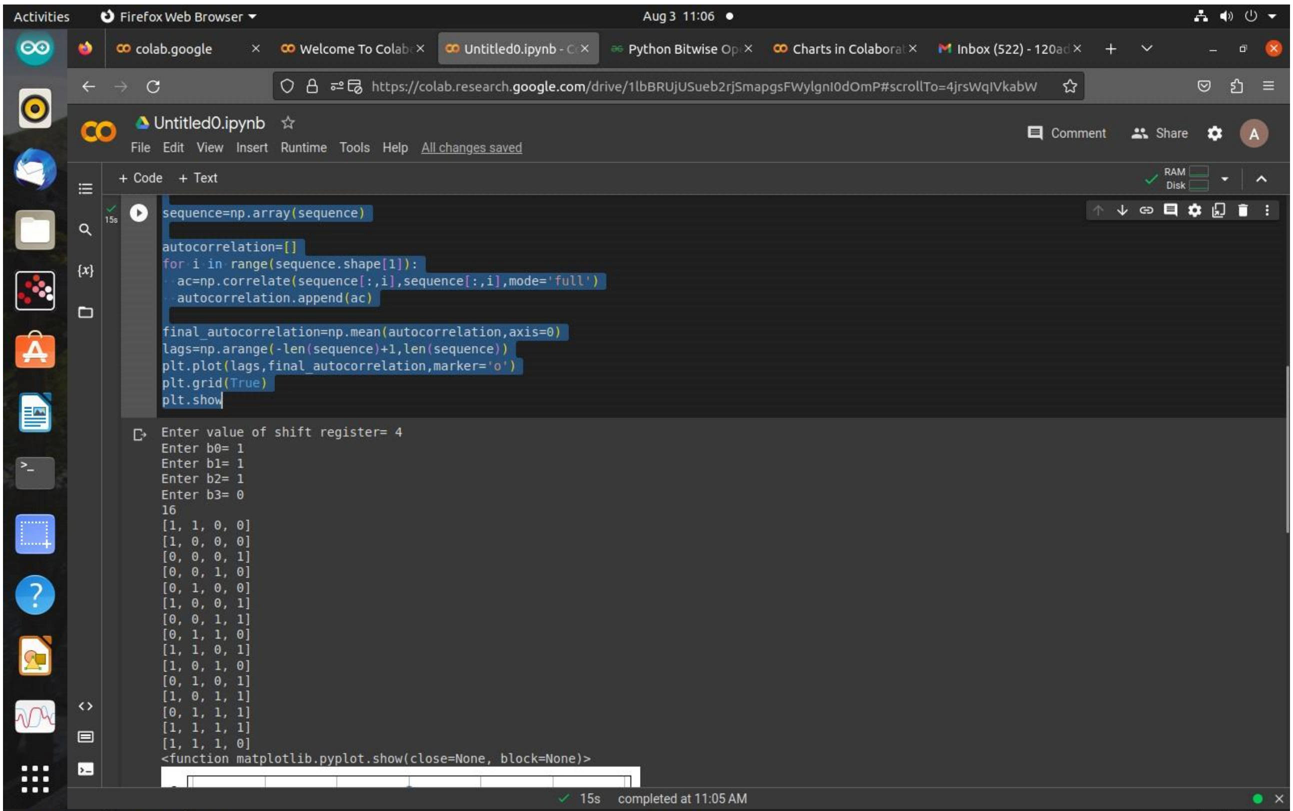
**Code:**

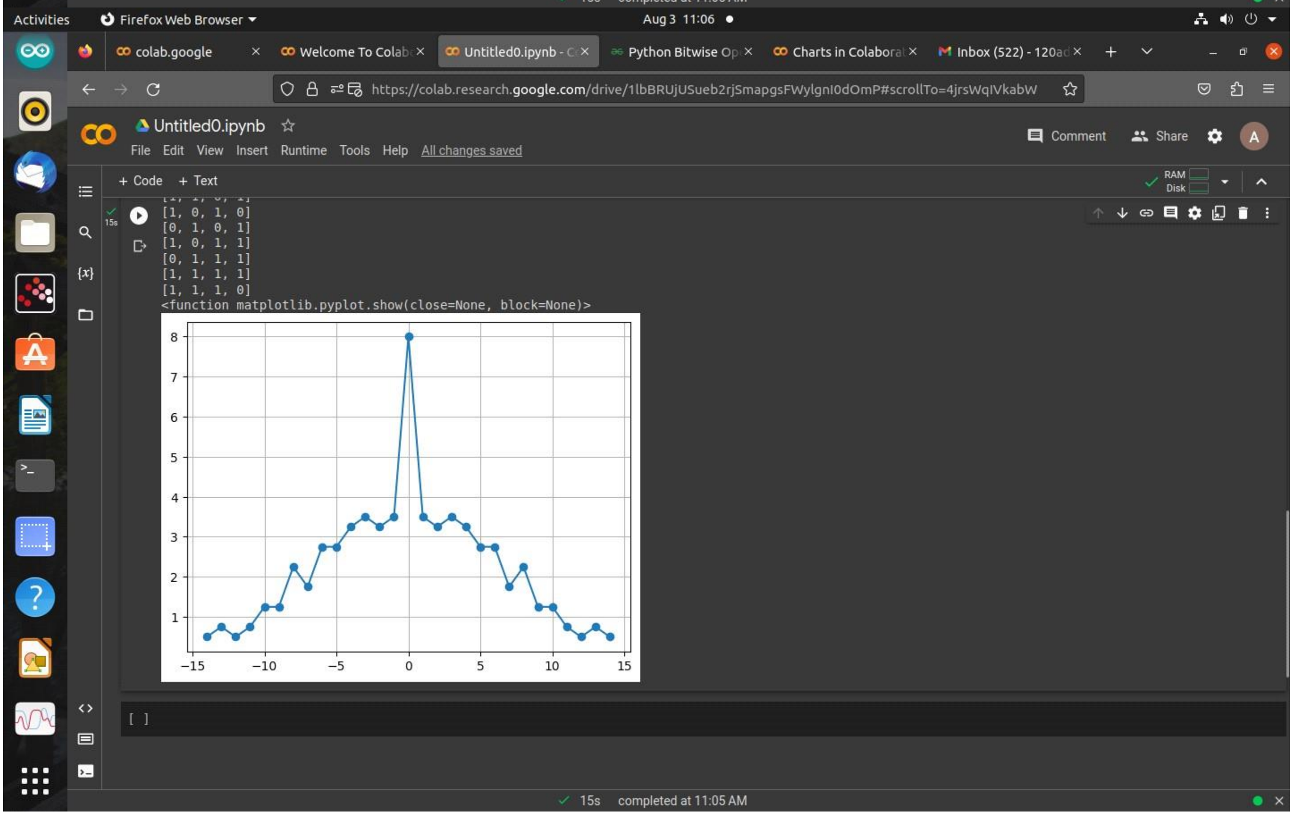
```
import matplotlib.pyplot as plt
import numpy as np
```

```
m=int(input("Enter value of shift register= "))
b0=int(input("Enter b0= ")) b1=int(input("Enter
b1= ")) b2=int(input("Enter b2= "))
b3=int(input("Enter b3= "))

n=pow(2,m) print(n)

sequence=[] for i
in range(1,n):
listt=[]
b00=b0 b01=b1
b02=b2 b03=b3
b0=b01 b1=b02
b2=b03
b3=b0^b00;
listt.append(b0)
listt.append(b1)
listt.append(b2)
listt.append(b3)
print(listt)
sequence.append(listt)

sequence=np.array(sequence)

autocorrelation=[] for i in
range(sequence.shape[1]):
ac=np.correlate(sequence[:,i],[:,i],mode='full')
autocorrelation.append(ac)

final_autocorrelation=np.mean(,axis=0) lags=np.arange(-len(sequence)+,len(sequence))
plt.plot(lags,final_,marker='o')
plt.grid(True) plt.show
```

## Screenshots:


Activities
Firefox Web Browser
Aug 3 11:06
colab.google
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
import matplotlib.pyplot as plt
import numpy as np
m=int(input("Enter value of shift register= "))
b0=int(input("Enter b0= "))
b1=int(input("Enter b1= "))
b2=int(input("Enter b2= "))
b3=int(input("Enter b3= "))
n=pow(2,m)
print(n)
sequence=[]
for i in range(1,n):
listt=[]
b00=b0
b01=b1
b02=b2
b03=b3
b0=b01
b1=b02
b2=b03
b3=b0^b00;
listt.append(b0)
listt.append(b1)
listt.append(b2)
listt.append(b3)
print(listt)
sequence.append(listt)
sequence=np.array(sequence)
15s completed at 11:05 AM


Untitled0.ipynb
sequence=np.array(sequence)
autocorrelation=[]
for i in range(sequence.shape[1]):
ac=np.correlate(sequence[:,i],sequence[:,i],mode='full')
autocorrelation.append(ac)
final_autocorrelation=np.mean(autocorrelation,axis=0)
lags=np.arange(-len(sequence)+1,len(sequence))
plt.plot(lags,final_autocorrelation,marker='o')
plt.grid(True)
plt.show
Enter value of shift register= 4
Enter b0= 1
Enter b1= 1
Enter b2= 1
Enter b3= 0
16
[1, 1, 0, 0]
[1, 0, 0, 0]
[0, 0, 0, 1]
[0, 0, 1, 0]
[0, 1, 0, 0]
[1, 0, 0, 1]
[0, 0, 1, 1]
[0, 1, 1, 0]
[1, 1, 0, 1]
[1, 0, 1, 0]
[0, 1, 0, 1]
[1, 0, 1, 1]
[0, 1, 1, 1]
[1, 1, 1, 1]
[1, 1, 1, 0]
<function matplotlib.pyplot.show(close=None, block=None)>
15s completed at 11:05 AM


Untitled0.ipynb
[1, 0, 1, 0]
[0, 1, 0, 1]
[1, 0, 1, 1]
[0, 1, 1, 1]
[1, 1, 1, 1]
[1, 1, 1, 0]
<function matplotlib.pyplot.show(close=None, block=None)>
8
7
6
5
4
3
2
1
−15
−10
−5
0
5
10
15
15s completed at 11:05 AM

## Conclusion:

With the help of python we were able to design a PN Sequence Generator of length n and verify the various properties of it.