DESIGNING DSSS COMMUNICATION SYSTEMS

Submitted by:-

MANISH KUMAR – 071074

DIVESH KUMAR SOLANKI - 071122

SARVDAMAN SANGRA – 071055

Project Guide:-

Prof. T.S. Lamba

May – 2011

Submitted in partial fulfilment of the Degree of

Bachelor of Technology

Department of ELECTRONICS and COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY of INFORMATION TECHNOLOGY,

WAKNAGHAT

TABLE of CONTENTS

CERTIFICATE

This is to certify that the work entitled "**DESIGNING DSSS COMMUNICATION SYSTEMS**" submitted by "**MANISH KUMAR (071074) , DIVESH KUMAR SOLANKI (071122) and SARVDAMAN SANGRA (071055)**" in partial fulfilment for the award of degree of **BACHELOR of TECHNOLOGY**, of **JAYPEE UNIVERSITY of INFORMATION TECHNOLOGY**, Waknaghat, has been carried out under my supervision. This work has not been submitted partially or wholly to any other university or institute for the award of this or any other degree or diploma.

_____

Prof. **T.S. Lamba**

DEAN (A&R)

Department of **ECE**

Jaypee University of Information Technology,

Waknaghat, Solan.

DATE :-

Acknowledgement

We would like to express our gratitude to our Project guide **Prof. T.S. Lamba** for giving us the opportunity and suggesting us a very useful topic to be taken as the final year **B.Tech** project for the academic session **2010-2011**. The work which we have done in this project wouldn't have been possible without his support. He guided us by enlightening our path throughout the project and his attitude has been a morale booster for all of us and we could complete the project in stipulated time. It has been a pleasure to be working under him, and we would like to thank him for his support and guidance. Looking forward, to his support in all future endeavours.

Manish Kumar	Divesh Kumar Solanki	Sarvdaman Sangra
071074, ECE	071122, ECE	071055, ECE

SUMMARY

The purpose of our project is to design **DSSS** communication systems, in which many users will communicate with each other in the same channel, without affecting each other. The main aim of the project was to design a system using various available **code sequences**, and analyse their *performance*.

The present explosion in digital communications and multi-user wireless cellular networks has urged a demand for more effective modulation methods, utilizing the available frequency spectrum more efficiently. To accommodate a large number of users sharing the same available frequency band, **our requirement is the availability of large families of spreading sequences with excellent Autocorrelation and Cross-Correlation properties.** These spreading sequences also pose the property of orthogonality. And hence these sets of orthogonal basis functions can be used to extend capacity by exploiting all available degrees of freedom (e.g. temporal, frequency and spatial dimensions) or can be used to employ orthogonal multi-code operation in parallel, such as used in the latest *Wide-band Code Division Multiple Access* modulation standards by employing sets of orthogonal codes to improve the overall data throughput capacity.

It is known that the above property have a large potential for application in point to point, and particularly micro-cellular WLANs and WLL RF link. **Our focus** was on finding the unique large families of spreading sequences with excellent autocorrelation and cross-correlation properties. Conventional spread spectrum communication systems employ binary spreading sequences, such as **PRNS or Gold** sequences. The practical implementation of such a system is

relatively simple. But the further research is going on employing complex sequences in these systems, and has not been yet implemented on the hardware.

The project work was divided into three parts:-

i. Study of literature available on **SPREAD SPECTRUM**, and finally understanding the basic techniques of **DSSS** transmitter and receiver. The thorough study of **PRN** and **GOLD** codes was also done.
ii. The **PRN-Seq**. generator was designed both on software (using C++) and on hardware (on bread board using IC's). The result was obtained and compared with theoretical result available in different literatures.
iii. Finally, the DATA was spread and sent through the channel and at the receiver end it was de-spread and obtained successfully.

The above mentioned work has also been done for **GOLD codes**. The autocorrelation and cross-correlation of **PRN-Seq**. and **GOLD codes** are compared and the better result was taken as final consideration.

Manish Kumar	Divesh Kumar Solanki	Sarvdaman Sangra
071074, ECE	071122, ECE	071055, ECE

LIST of FIGURES

LIST of SYMBOLS and ACRONYMS

S. No.	SYMBOLS and ACRONYMS	DESCRIPTION
01	**DSSS**	Direct Sequence Spread Spectrum
02	**FHSS**	Frequency Hoping Spread Spectrum
03	**THSS**	Time Hoping Spread Spectrum
04	**PRN Sequence**	Pseudo Random Noise Sequence
05	***fc*** and ***f***	Chip rate and Bit rate
06	**CDMA**	Code Division Multiple Access
07	**LFSR**	Linear Feedback Shift Register
08	**GPS**	Geo Positioning System
09	**LPF**	Low Pass Filter (or Integrator)
10	**CRO**	Cathode Ray Oscilloscope
11	***Ak*** **and** ***Ad***	Chiping Seq. And Data for user A
12	***Bk*** **and** ***Bd***	Chiping Seq. And Data for user B
13	***S =Ak*Ad+Bk*Bd***	Received Signal
14	**XOR**	Exclusively OR ($B1 \oplus B0$)
15	**N**	Band of Frequency for FHSS
16	**RF**	Radio Frequency used in transmission

CHAPTER 1

SPREAD SPECTRUM

1.1 INTRODUCTION

With the growth in wireless communication and increasing demand for better methods of communication has raised the need for more robust and effective technologies to improve communication systems. One such technique is Spread Spectrum, which has become very popular and an important method for communication. Spread Spectrum involves spreading the desired signal over a bandwidth much larger than the minimum bandwidth necessary to send the signal. Though Spread spectrum was first used for military purposes, but it has become very popular in commercial communication systems recently. Spread Spectrum methods have many advantages over other basic communication methods, such as very good interference performance, resistance to jamming, good performance in multipath fading, more robust in noise etc. In this chapter we will cover the details behind the method of Spread Spectrum communications, as well as describe two main types of Spread Spectrum systems, *Direct-Sequence Spread Spectrum* (DS-SS) and *Frequency-Hopped Spread Spectrum* (FH-SS). We will talk about benefits of Spread Spectrum techniques. We also discuss in brief that how different Spread Spectrum techniques can be combined to have a good effective communication system. Finally, a general comparison between the two will be given, trying to indicate the positives and negatives for each with respect to the other, and to indicate when one system might be preferable over the other.

1.2 HISTORY

Spread Spectrum scheme was first proposed by a well known Austrian actress *Hedy Lamarr* and a music composer *George Antheil*. This technique was proposed to control torpedoes over long distances. The traditional guiding system for torpedoes was prone to detection and can easily be jammed, so to have a better and more robust guiding system which is immune to jamming and detection the Spread Spectrum technique was proposed. This new guiding system which mainly implemented the *frequency-hopping spread spectrum* (FH-SS) was very effective against jamming and detection as their signal would hop from one frequency to another in a *pseudorandom* fashion known only to an authorized receiver (i.e.,

the torpedo).This would cause the transmitted spectrum to spread over a range much greater than the message bandwidth. Later the *frequency-hopping spread spectrum* (FH-SS) was patented by *Lamarr* and *Antheil*.

1.3 Direct-Sequence Spread Spectrum (DSSS)

In the DSSS technique, the PRN is applied directly to data before the modulation stage. The modulator has to modulate a much larger data which is because of spreading of data by PRN sequence just before modulation. At the modulator the spread signal is used for modulation of the carrier. Thus modulator has to handle a larger data rate, which is the chipping rate of the PRN sequence. Modulating an RF carrier with such a code sequence produces a direct-sequence-modulated spread spectrum with $((\sin x)/x)^2$ frequency spectrum, centred at the carrier frequency.

The main lobe of the frequency spectrum null to null has a bandwidth twice the clock rate of the modulating code, and the side lobes have null-to-null bandwidths equal to the code's clock rate. Illustrated in **Figure 1** is the most common type of direct-sequence-modulated spread-spectrum signal. Direct-sequence spectra vary somewhat in spectral shape, depending on the actual carrier and data modulation used. Below is a binary phase shift keyed (BPSK) signal, which is the most common modulation type used in direct-sequence systems.

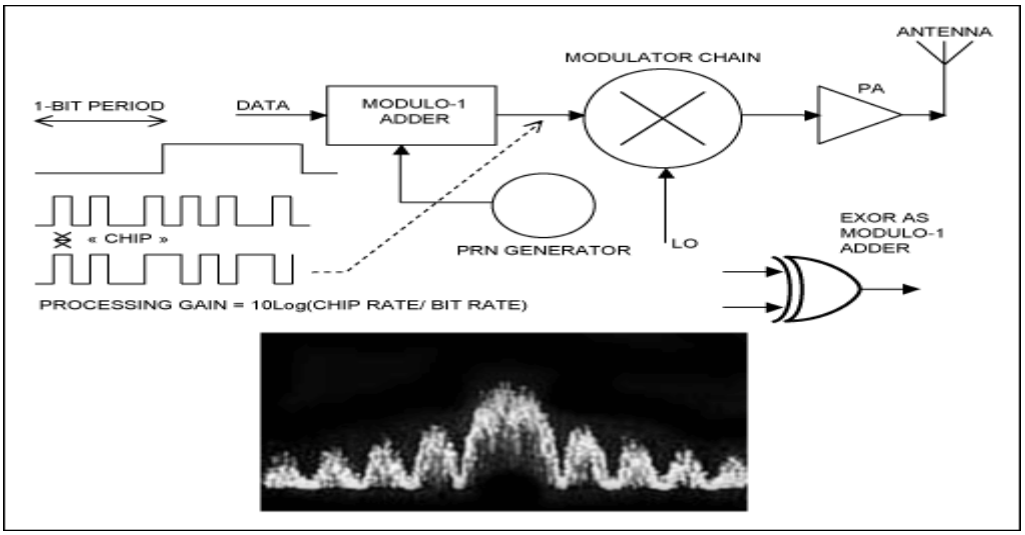


Figure 1. Spectrum-analyzer photo of a DSSS signal. Note the original signal (non-spread) would only occupy half of the central lobe (this fig. Is taken from Maxim Integrated Products Note [d]*).*

Let *f* be the frequency of the data signal, with appropriate pulse time T=1/*f*. Let the PRN sequence be transmitted at a rate f_c, so that the increase in the data rate is f_c/f. The frequency f_c is known as the *chipping rate*, with each individual bit in the modulating sequence known as a *chip*. Thus the width of each pulse in the modulating sequence is T_C, or a *chip time*. The following figure illustrates the two signals, the data signal for one pulse width, and the PRN sequence over the same time (since the PRN sequence takes values of ±1, the indicated PRN sequence also indicates a normalized version of the signal to be transmitted).

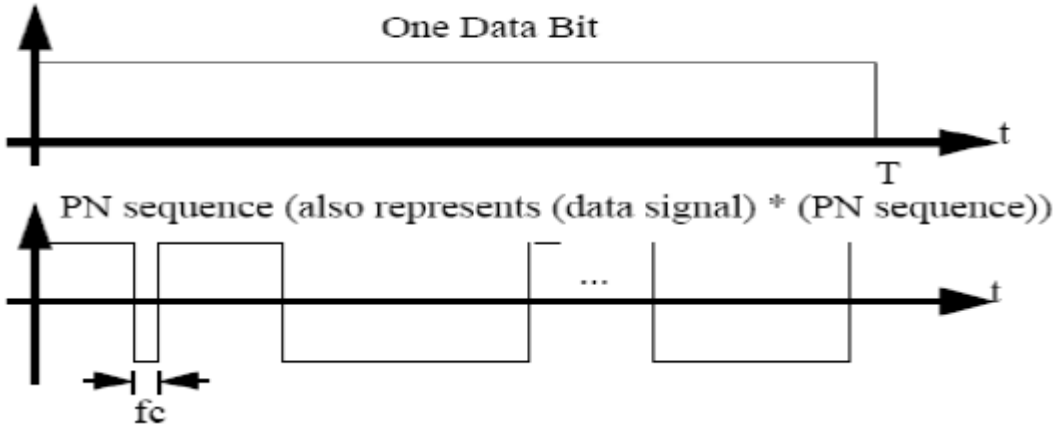


Figure 2. Data signal and PRN sequence in Time Domain

As a result, the frequency domain will look something like the diagram shown in **Figure 3**.

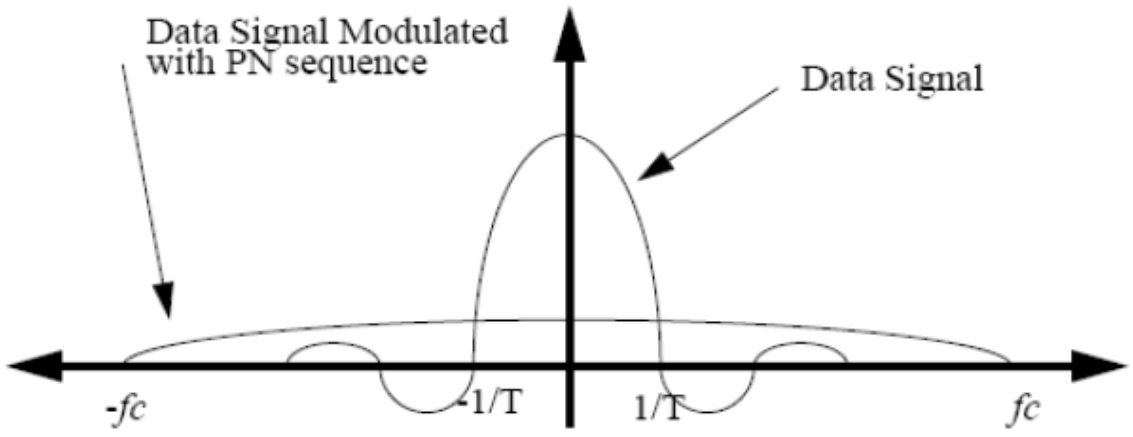


Figure 3. Data signal and PN sequence modulated in Frequency Domain

Mathematically, the following happens. Let the data signal be *D(t)*, transmitted at frequency *f*, and let the PRN sequence be *PN(t)*, with frequency f_c. So the transmitted signal is

$$S(t) = D(t)PN(t)$$

The PN sequence is designed such that it has very good autocorrelation properties:

$$R_{PN}(\tau) = \begin{cases} 1, & t = 0, N, 2N \\ -1/N & \text{otherwise} \end{cases}$$

Where N is the length of the PRN sequence, therefore, when the signal is correlated with the PRN sequence at the receiver, the received signal will be recovered exactly (assuming that there is synchronization between the send and receive PRN sequences), i.e.

$$S(t)PN(t) = D(t)PN(t)PN(t) = D(t)$$

1.4 Frequency-Hopping Spread Spectrum (FHSS)

In FHSS method the PRN sequence is applied to frequency synthesizer or local oscillator such that the local oscillator generates different carrier frequencies at different times in a random fashion. In this way the carrier frequency keeps hopping from one frequency to other frequency over a wide band according to a sequence defined by the PRN. The speed at which the hops are executed depends on the data rate of the original information. There are two types of FHSS methods one is *fast frequency hopping* (FFHSS) and other is *low frequency hopping* (LFHSS). In low frequency hopping (LFHSS) several consecutive data bits modulate the same frequency, whereas in fast frequency hopping FFHSS is characterized by several hops within each data bits.

The transmitted spectrum of a frequency-hopping signal is quite different from that of a direct-sequence system. Instead of a $((\sin x)/x)^2$ - shaped envelope, the frequency hopper's output is flat over the band of frequencies used (see **Figure 4**). The bandwidth of a frequency-hopping signal is simply N times the number of frequency slots available, where N is the bandwidth of each hop channel.

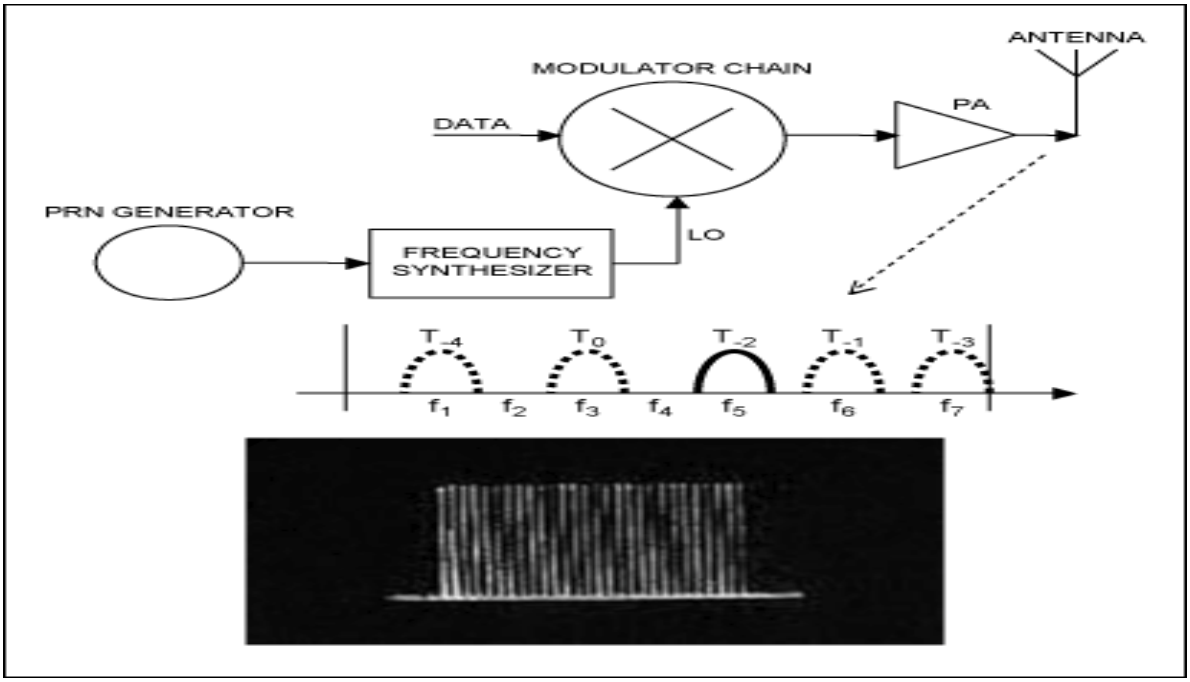


Figure 4. Spectrum-analyzer photo of a FHSS signal (this fig. Is taken from Maxim Integrated Products Note [d]*).*

In FHSS, the signal itself is not spread across the entire large bandwidth; instead the wide bandwidth is divided into N sub-bands, and the signal "hops" from one band to the next in a pseudorandom manner. The centre frequency of the signal changes from one hop to the next, changing from one sub-band to another, as shown in **Figure 5**.

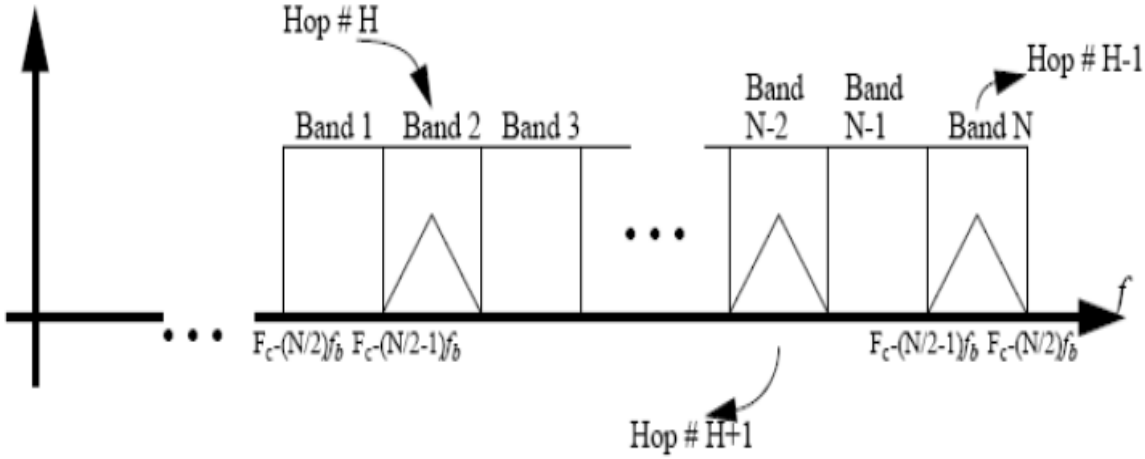


Figure 5. A Frequency Domain view of FHSS

1.5 Different Modulation Spreading Techniques for SPREAD SPECTRUM

Different spread-spectrum techniques are distinguished according to the point in the system at which a PRN is inserted in the communication channel. If the PRN sequence is applied before carrier modulation then it is known as Direct Sequence Spread Spectrum (DSSS) (In practice, the pseudo-random sequence is mixed or multiplied with the information signal, giving an impression that the original data flow was "hashed" by the PRN). If the PRN sequence is applied to the frequency synthesizer which generates carrier frequency for modulation then it is frequency hopped spread spectrum (FHSS). If the PRN acts as an on/off gate to the transmitted signal, this is a time-hopping spread-spectrum technique (THSS). There is also the "chirp" technique, which linearly sweeps the carrier frequency in time. This is very basically illustrated in the RF front-end schematic in **Figure 6**.

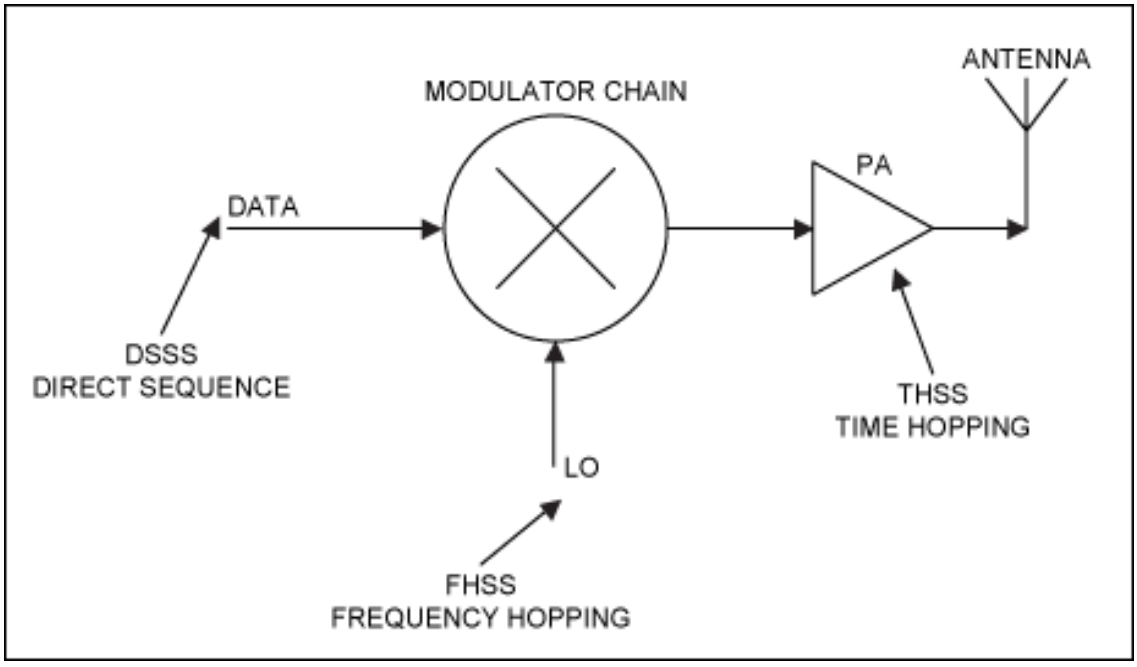


Figure 6. Several spreading techniques are applied at different stages of the transmit chain (this fig. Is taken from Maxim Integrated Products Note [d]*).*

One can also mix these different techniques to get a combination. More preferred are DSSS and FHSS, but these two can be used with THSS for better results.

1.6 BENEFITS OF SPREAD SPECTRUM

Resistance to Interference and Anti-jamming Effects

There are many benefits to spread-spectrum technology. Resistance to interference is the most important advantage. Intentional or unintentional interference and jamming signals are rejected because they do not contain the spread-spectrum key. Only the desired signal, which has the key, will be seen at the receiver when the de-spreading operation is exercised. You can practically ignore the interference, narrowband or wideband, if it does not include the key used in the de-spreading operation. That rejection also applies to other spread-spectrum signals that do not have the right key. Thus different spread-spectrum communications can be active simultaneously in the same band. Note that spread spectrum is a wideband technology, but the reverse is not true: wideband techniques need not involve spread-spectrum technology.

Resistance to Interception

Resistance to interception is the second advantage provided by Spread Spectrum techniques. Because non authorized listeners do not have the key used to spread the original signal, those listeners cannot decode it. Without the right key, the spread-spectrum signal appears as noise or as an interferer. (Scanning methods can break the code, however, if the key is short). The key involved is PRN sequence or other orthogonal sequences like gold and kasami sequences. Even better, signal levels can be below the noise floor, because the spreading operation reduces the spectral density. The message is thus made invisible, an effect that is particularly strong with the direct-sequence Spread Spectrum (DSSS) technique. Other receivers cannot "see" the transmission; they only register a slight increase in the overall noise level.

Resistance to Fading (Multipath Effect)

Wireless channels often include multiple-path propagation in which the signal has more than one path from transmitter to the receiver. Such multipath can be caused by atmospheric reflection or refraction, and by reflection from the ground or from objects such as buildings. The reflected path (R) can interfere with the direct path (D) in a phenomenon called fading. Because the de-spreading process synchronizes to signal D, signal R is rejected even though

it contains the same key. Methods are available to use the reflected-path signals by de-spreading them and adding the extracted results to the main one.

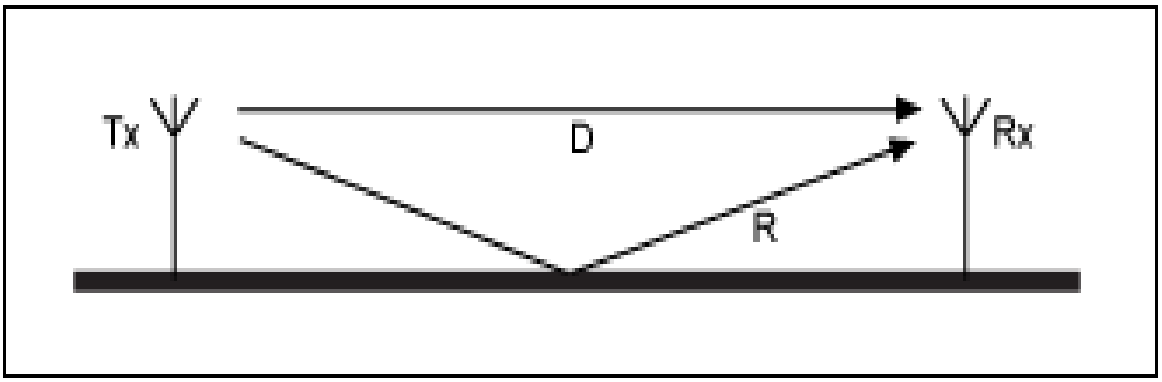


Figure 7. Illustration of how the signal can reach the receiver over multiple paths.

DS suppresses multipath by de-correlating the delayed signal. When multipath signals are delayed by more than one chip relative to the direct path signal, the direct signal has a processing gain advantage. When the multipath signal arrives within a one-chip delay, this creates fading. That is the direct signal can be either enhanced or suppressed. Therefore, for DS to achieve significant multipath rejection, its bandwidth must be wider than the coherence delay of the environment.

1.7 FHSS Vs DSSS

Frequency Hopping Spread Spectrum (FHSS)	**Direct Sequence Spread Spectrum (DSSS)**
When using FHSS, the frequency spectrum is divided into channels. Data packets are split up and transmitted on these channels in a random pattern known only to the transmitter and receiver.	The DSSS encoder spreads the data across a broad range of frequencies using a mathematical key. The receiver uses the same key to decode the data.
If interference is present on one channel, data transmission is blocked. The transmitter and receiver 'hop' to the next channel in the hop table and the transmitter resends the data packet.	In interference the wider band transmission is decoded back to its original narrowband format while the interference is decoded to a lower power density signal, thereby reducing its effects. When broadband interference is present, however, the resulting decoded

	broadband interference can give a much higher noise floor, almost as high as the decoded signal.
Frequency hopping technology works best for small data packets in high interference environments.	DSSS works best for large data packets in a low to medium interference environment
Traditional FH signals lower their average power spectral density by hopping over many channels.	DS spreads its energy by rapidly phase chopping the signal so that it is continuous only for very brief time intervals. the total power is the same, but the spectral density is lower.

CHAPTER 2

PRN SEQUENCES

2.1 PRN codes

The PRN codes are set of deterministically generated sequences, which mimic certain properties of noise. The properties of noise which satisfy the principle properties of PRN codes are :-

- Sharp Autocorrelation, so that any time shifted version of a PRN code has almost no correlation with the original sequence.
- Equal number of "1"s and "0"s in any long segment of the sequence, so that the signal has no bias.
- Random and independent appearance of "1"s and "0"s, so that it is difficult to reconstruct the sequence from any short segment.

2.2 Generation of PRN Sequences

The PRN generator for spread spectrum is usually implemented as a circuit consisting of XOR gates and a shift register, called a *linear feedback shift register* (LFSR). The LFSR is a string of 1-bit storage devices. Each device has an output line, which indicates the value currently stored, and input line. At discrete time instants, known as clock times, the value in the storage device is replaced by the value indicated by its input line. The entire LFSR is clocked simultaneously, causing a 1-bit shift along the entire register. The LFSR contains n bits. There are from 1 to (n-1) XOR gates. The presence or absence of a gate corresponds to the presence or absence of a term in the generator polynomial(X), excluding the X^n term.

2.2.1 Generating polynomials

Finite (Galois) field mathematics are used to derive m-sequence feedback taps. Any LFSR can be represented as a polynomial of variable X, referred to as the *generator polynomial.*

$$G(X) = g_m X^m + g_{m-1} X^{m-1} + g_{m-2} X^{m-2} + \ldots + g_2 X^2 + g_1 X + g_0$$

The coefficients g_i represent the tap weights, and are 1 for taps that are connected (feedback), and 0 otherwise. The order of the polynomial, m, represents the number of LFSR stages. Rules of linear algebra apply to the polynomial, but all mathematical operations are performed in modulo-2:

Modulo-2 addition:

0 + 0 = 0
0 + 1 = 1
1 + 1 = 0

Modulo-2 multiplication:

0 * 0 = 0
0 * 1 = 0
1 * 1 = 1

The generator polynomial of m-sequence is *primitive polynomial*. g(x) is a primitive polynomial of degree m if the smallest integer n for which g(x) divides $X^n + 1$ is $n=2^m - 1$. The generator polynomial is said to be *primitive* if it cannot be factored (i.e. it is prime).

As mentioned above the sequence for n th coefficient can be written in a mathematical equation as :-

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} \ldots\ldots\ldots\ldots + c_r a_{n-r} = \sum_{i=1}^{r} c_i \; a_{n-i} \qquad (2.1)$$

where, c_1 to c_r are connection variables (1 for connection and 0 for no connection)

multiplication is simple and addition is modulo -2 (or "Exclusive OR"). In the above equation linear and distributive rule is applicable. In general, the generating function of the sequence , with nonnegative index can be defined as

$$G(D) \triangleq a_0 + a_1 D + a_2 D^2 + \cdots = \sum_{n=0}^{\infty} a_n D^n \qquad (2.2)$$

where D is the delay operator and number of clock cycles of delay are the power of D of polynomial .

On combining equation (2.1) and (2.2), now we can reduce the equation (2.1) to finite recurrence relation

$$G(D) = \sum_{n=0}^{\infty} a_n D^n = \sum_{n=0}^{\infty} \sum_{i=1}^{r} a_n D^n$$

$$= \sum_{i=1}^{r} c_i D^i \left[\sum_{n=0}^{\infty} a_{n-i} D^{n-i}\right]$$

$$= \sum_{n=0}^{\infty} c_i D^i \left[a_{-i} D^{-i} + \cdots + a_{-1} D^{-1} + G(D)\right]$$

Now we can express the G(D) as a ratio of finite polynomials as

$$G(D)\left(1 - \sum_{i=1}^{r} c_i D^i\right) = \sum_{i=1}^{r} c_i D^i \left[a_{-i} D^{-i} + \cdots + a_{-1} D^{-1}\right]$$

$$G(D) = \frac{\sum_{i=1}^{r} c_i D^i \left[a_{-i} D^{-i} + \cdots + a_{-1} D^{-1}\right]}{1 - \sum_{i=1}^{r} c_i D^i} \triangleq \frac{g_0(D)}{f(D)} \qquad (2.3)$$

where $f(D) = 1 - \sum_{i=1}^{r} c_i D^i$ (2.4)

f(D)is called the characteristics Polynomial of the shift register sequence generator and depends solely on the connection vector $c_{1,\ldots,}c_r$. The polynomial $g_0(D)$ depends as well on the initial condition vector $a_{-r},a_{-r-1,\ldots,}a_{-1}$. $g_0(D)$ can be written as :

$$g_0(D) = \sum_{i=1}^{r} c_i (a_{-i} + a_{-i} D + \cdots + a_{-i} D^{i-1}) \qquad (2.5)$$

$$= c_1 a_{-1} + c_2 (a_{-2} + a_{-1} D) + c_2 (a_{-3} + a_{-2} D + a_{-1} D^2) + \ldots + c_r (a_{-r} + a_{-r+1} + \ldots + a_{-1} D^{r-1}).$$

In the above equation , of all the connection variables, atleast $c_r=1$, for otherwise the shift register would no longer need to have r stages . Here we consider the initial vector :

$a_{-r}=1, a_{-r+1}=\ldots=a_{-2}=a_{-1}=0$,

in which case (2.5) and (2.3) reduce to

$$g_0(D) = 1,\ G(D) = \frac{1}{f(D)} \qquad (2.6)$$

For better understanding of the mathematics used here one can refer to the book entitled "*Wireless Communication*" by Andrea Goldsmith [a].

The mathematics used is very useful in determining the Generator polynomial. But, Engineers don't need to do all those calculations, as Generating Polynomials for many m-

sequences are available in different literatures. The table given below has been taken from book entitled "*Wireless Communication*" by Andrea Goldsmith [a].

Table for Maximal Length Shift Register Sequences

Number of Shift Register stages, N	Sequence Length, $L = 2^N-1$	Number of m-sequences	Example Generating Polynomial	Tap value of Shift Register
2	3	1	X^2+X+1	$B_1 \oplus B_0$
3	7	2	X^3+X+1	$B_1 \oplus B_0$
4	15	2	X^4+X+1	$B_1 \oplus B_0$
5	31	6	X^5+X2+1	$B_2 \oplus B_0$
6	63	6	X^6+X+1	$B_1 \oplus B_0$
7	127	18	X^7+X+1	$B_1 \oplus B_0$
8	255	16	$X^8+X^6+X^5+X+1$	$B_6 \oplus B_5 \oplus B_1 \oplus B_0$
9	511	48	X^9+X^4+1	$B_4 \oplus B_0$
10	1023	60	$X^{10}+X^3+1$	$B_3 \oplus B_0$
11	2047	176	$X^{11}+X^2+1$	$B_2 \oplus B_0$

Though this table enlist some of the m-sequences, but there are many more sequences available for a particular length. For more details on this topic one can read a book entitled "*Error Correcting Codes*" by W.W Peterson [b].

In our project, we have worked on PN-Seq. of length 31. Hence, the available 6 Generating Polynomial for PN-Seq. of length 31 are:-

i. $\mathbf{1 + X^2 + X^5}$
ii. $\mathbf{1 + X^3 + X^5}$
iii. $\mathbf{1 + X^2 + X^3 + X^4 + X^5}$
iv. $\mathbf{1 + X + X^3 + X^4 + X^5}$
v. $\mathbf{1 + X + X^2 + X^4 + X^5}$
vi. $\mathbf{1 + X + X^2 + X^3 + X^5}$

For the above mentioned Generating Polynomials, we have worked on their implementation on hardware and software using Linear Feedback Shift Register (LFSR). Their autocorrelation and cross-correlation has also been calculated.

2.2.2 LFSR Generator Implementations

The implementation of a Linear feedback shift register, the content of which is modified at every step by a binary-weighted value of the output stage using modulo-2 math. For any given tap, weight g_i is either 0, meaning "no connection," or 1, meaning it is fed back. Two exceptions are g_0 and g_m, which are always 1 and thus always connected.

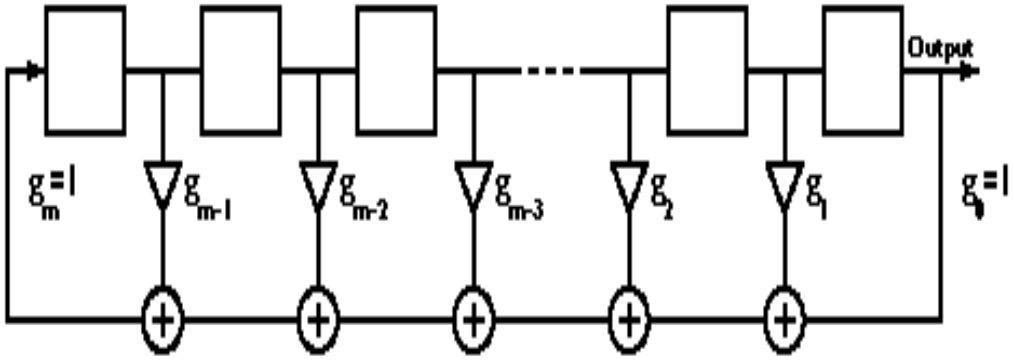


Figure 8. implementation of LFSR.

When implemented in hardware, modulo-2 addition is performed using exclusive-OR (XOR) gates.

2.3 Properties of PN Sequences

Property I – The Shift Property

A cyclic shift(left-cyclic or right-cyclic) of an m-sequence is also an m-sequence.

Property II – The window Property

If a window of width m is slid along an m-sequence in *Sm*, each of 2^m-1 nonzero binary m-tuples is seen exactly once in one period of the sequence.

Property III – One more 1 than 0's

Any m-sequence in *Sm* contains 2^{m-1} 1's and 2^{m-1}-1 0's.

Property IV – The Shift and Add Property

The sum of an m-sequence and a cyclic shift of itself (mod2,term by term) results in to the shifted version of the same sequence.

Property V – Runs

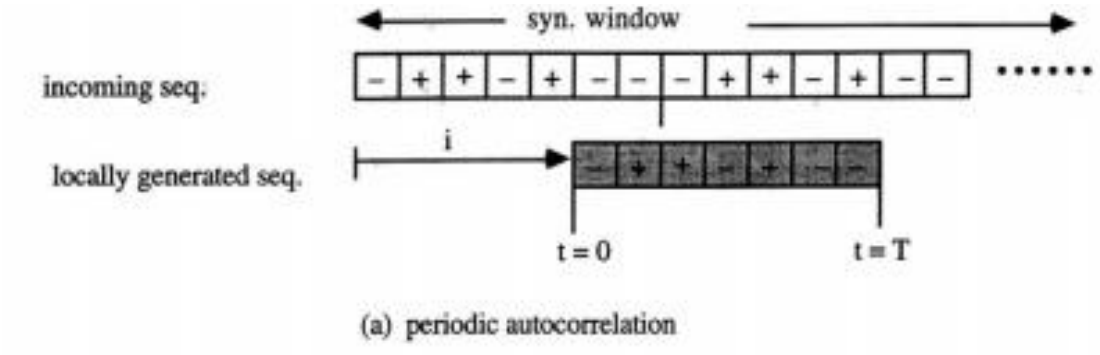
A run is string of consecutive 1's or a string of consecutive 0's. In any m-sequence, one-half of the runs have length 1, one quarter have length 2, one-eighth have length 3, and so on. In particular, there is one run of length m of 1's , one run of length m-1 of 0's.

2.3.1 Autocorrelation of PRN Sequences

a. Periodic autocorrelation

i. The Autocorrelation calculated when autocorrelation window is longer than the length of the sequence for which we are calculating the Autocorrelation.

ii. PRN sequences show very sharp Autocorrelation property, that is they show very high peak at complete matching of Autocorrelation window with the sequence or very low negligible value for mismatched window.



(a) periodic autocorrelation

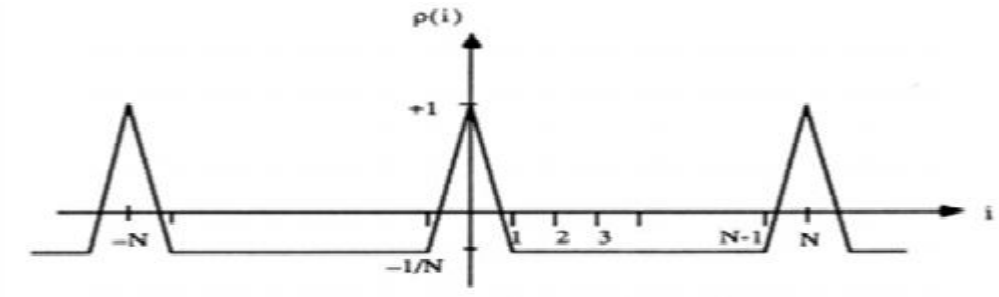


Figure 9. Periodic autocorrelation.

b. Aperiodic autocorrelation

i. If the autocorrelation window is just one period long or less than that, then we will get the aperiodic autocorrelation.

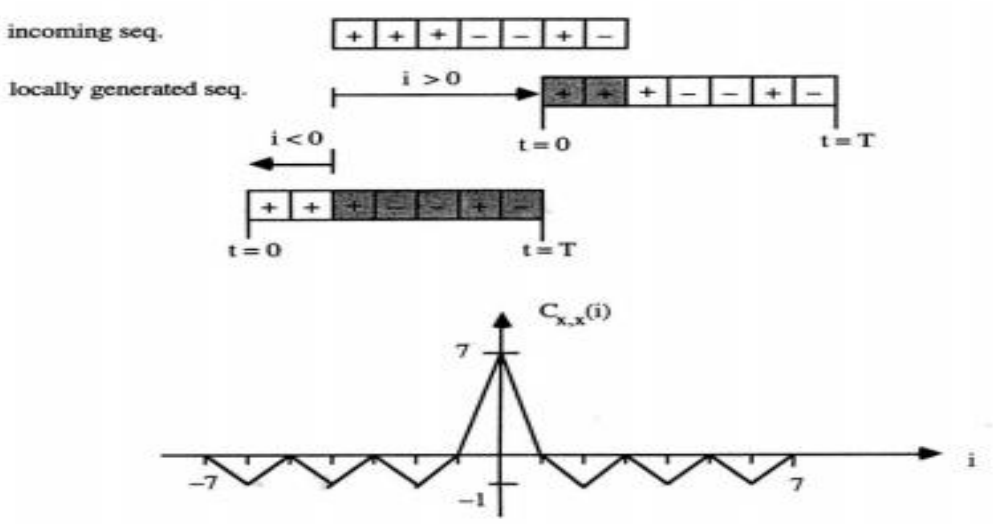


Figure 10. Aperiodic autocorrelation for 7-bit Sequence.

2.4 PROJECT WORK DONE ON PRN-SEQ.

For sequence length of 31 we have used the available m-sequence of order 5. The PN-Sequence generation has been carried out in two ways:- Computer simulation and Hardware implementation.

2.4.1 HARDWARE IMPLEMENTATION

i. We have designed PN sequence generator using generating polynomials.

ii. For generating PN sequence of length 7 the polynomial used is :

$$P(x) = X^3+X+1$$

iii. The design was implemented using **linear feedback shift-register** and **exclusive OR-gate** circuits (for 7 bit sequence only).

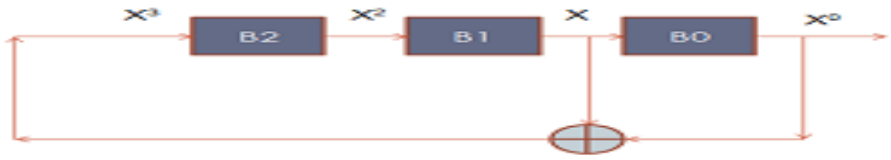
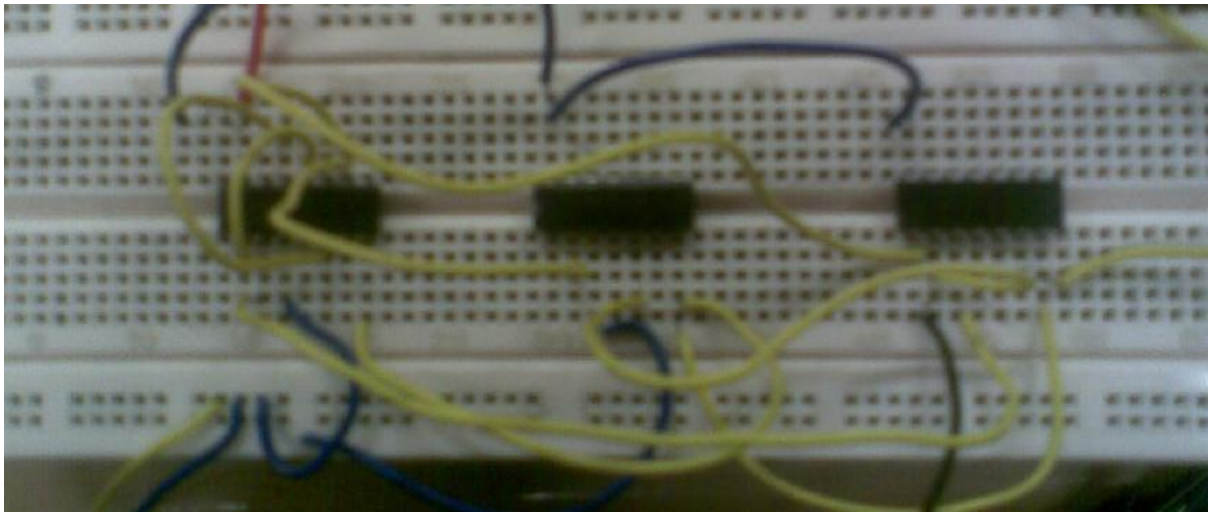
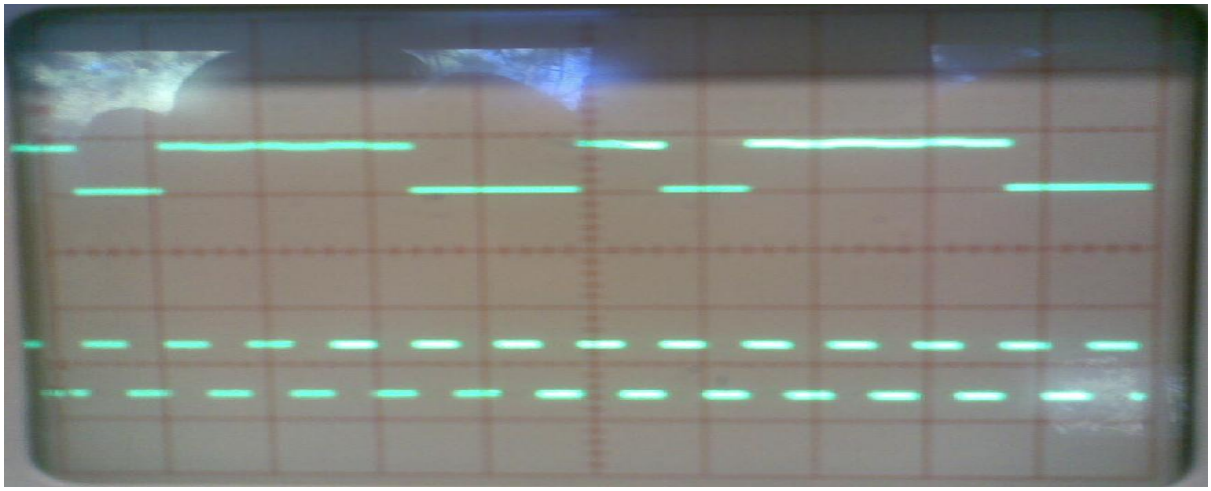


Figure 11: LFSR arrangement for PN-Sequence of length 7.



Snapshot1:- This snapshot shows the implementation of PN-Sequence generation of length 7.



Snapshot2:- This snapshot shows 7 lengths PN-Sequence on CRO.

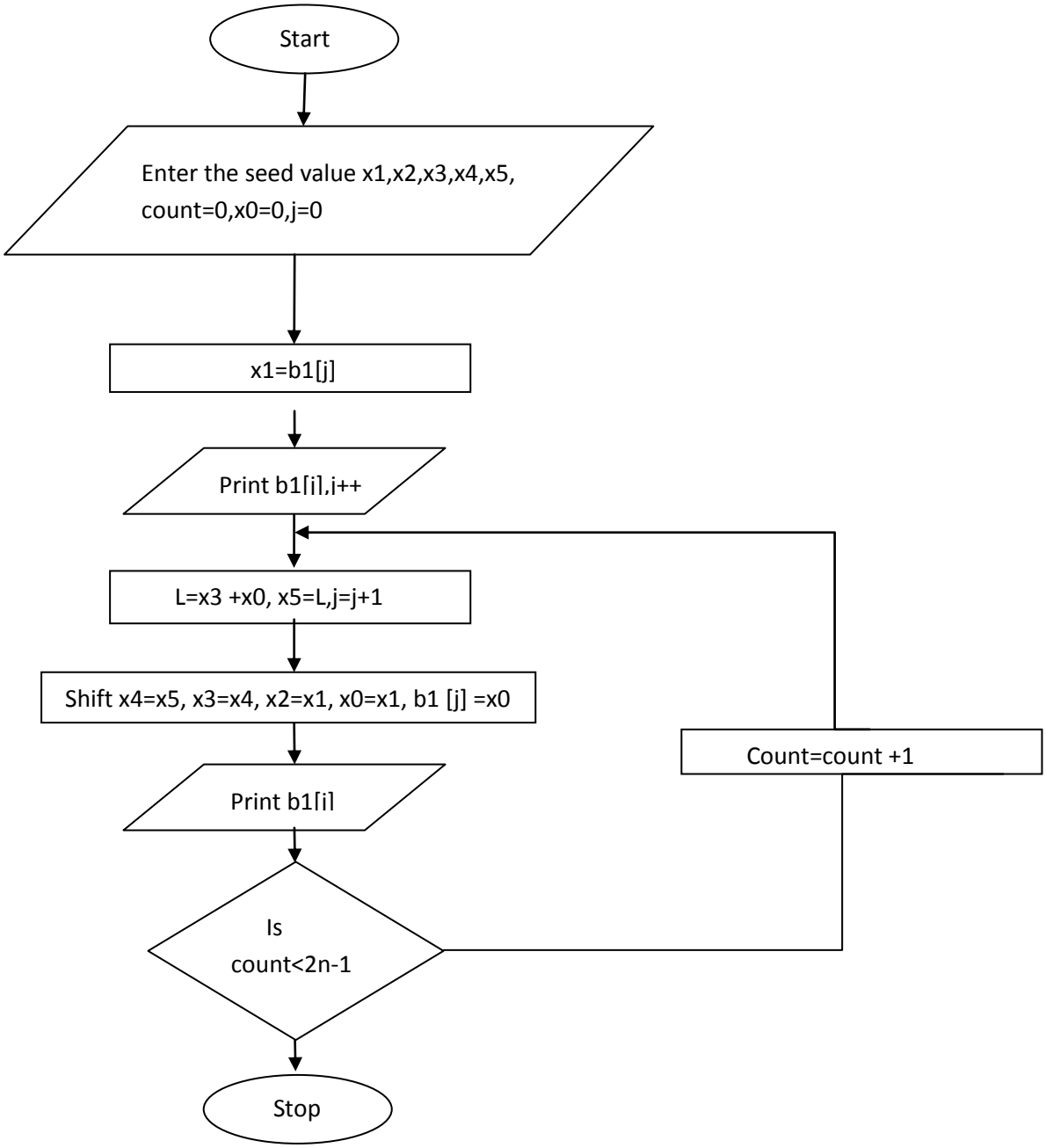
2.4.2 SOFTWARE SIMULATION

a. The PN code sequences of varying lengths (31, 63,255,511) were generated with the help of C++.

b. Waveform plots of PN sequences generated were made using graphics user interface in C++.

c. For generating PN-Sequence of length 31, we use various generating polynomial available to us. In the example given below, we have used $X^5 + X^3 + 1$. The LFSR diagram for this polynomial is same as above; the only difference would come for tap value and number of flip-flops.

d. The Flow chart of the code used for PRN Sequence generator is given below:-

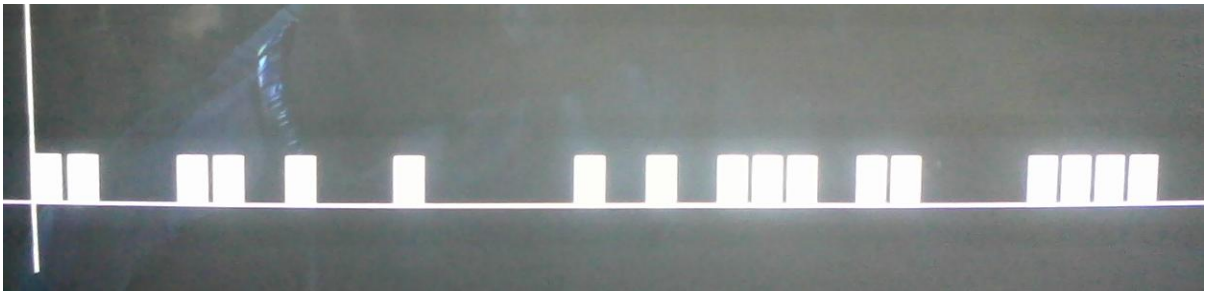


OUTPUT and WAVEFORM

```
C:\TCWIN45\BIN\NONAME00.EXE

 Enter the seed value in 1 bit binary:-
        x5 = 1
        x4 = 0
        x3 = 0
        x2 = 1
        x1 = 1

 the random number sequence is as follows :-
clk     b4      b3      b2      b1      b0      PNSEQ.
0       1       0       0       1       1       1
1       1       1       0       0       1       1
2       0       1       1       0       0       0
3       1       0       1       1       0       0
4       0       1       0       1       1       1
5       0       0       1       0       1       1
6       1       0       0       1       0       0
7       0       1       0       0       1       1
8       0       0       1       0       0       0
9       0       0       0       1       0       0
10      0       0       0       0       1       1
11      1       0       0       0       0       0
12      0       1       0       0       0       0
13      1       0       1       0       0       0
14      0       1       0       1       0       0
15      1       0       1       0       1       1
16      1       1       0       1       0       0
17      1       1       1       0       1       1
18      0       1       1       1       0       0
19      1       0       1       1       1       1
20      1       1       0       1       1       1
21      0       1       1       0       1       1
22      0       0       1       1       0       0
23      0       0       0       1       1       1
24      1       0       0       0       1       1
25      1       1       0       0       0       0
26      1       1       1       0       0       0
27      1       1       1       1       0       0
28      1       1       1       1       1       1
29      0       1       1       1       1       1
30      0       0       1       1       1       1
31      1       0       0       1       1       1
```



Snapshot3:- This snapshot shows 31 lengths PN-Sequence.

2.5 PN-Seq. AUTOCORRELATION

The Autocorrelation of PN-Sequence of Length 7 is calculated on the paper using Dixon's method. Then it is verified by Hardware and Software simulation.

2.5.1 HARDWARE IMPLEMENTATION

a. Then we designed the autocorrelator circuit using linear shift registers (for 7 bit sequence only).
b. For designing of autocorrelator circuit, we used Dixon's method.
c. The circuit diagram used to calculate the Autocorrelation using Inverting Amplifier is:

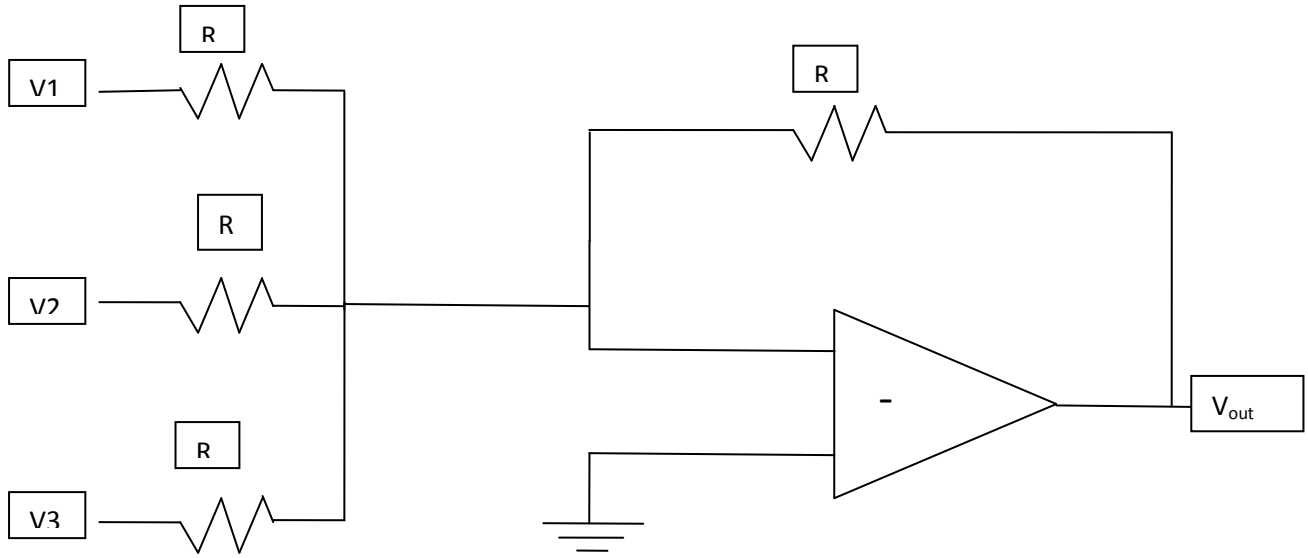
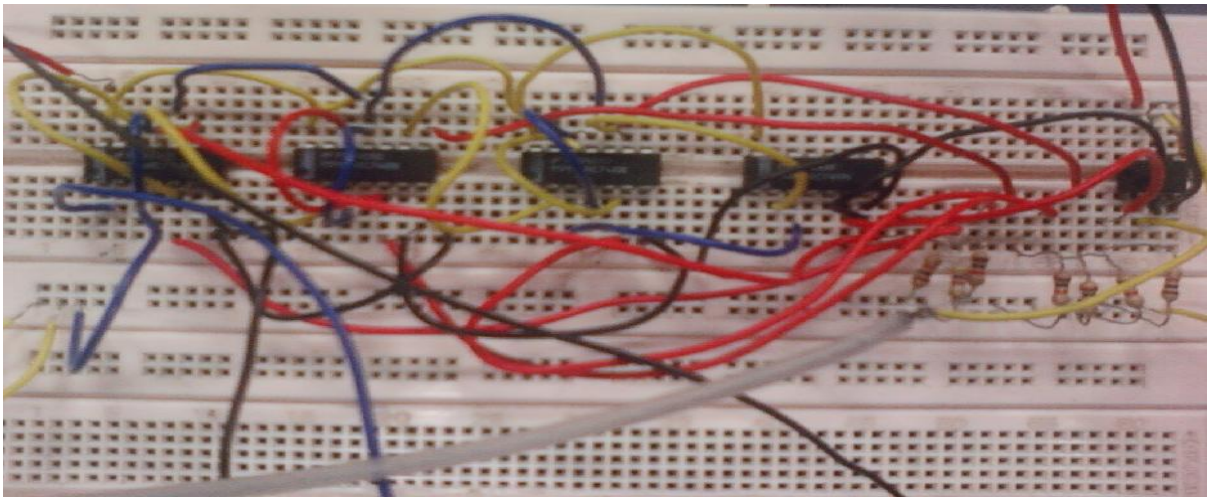


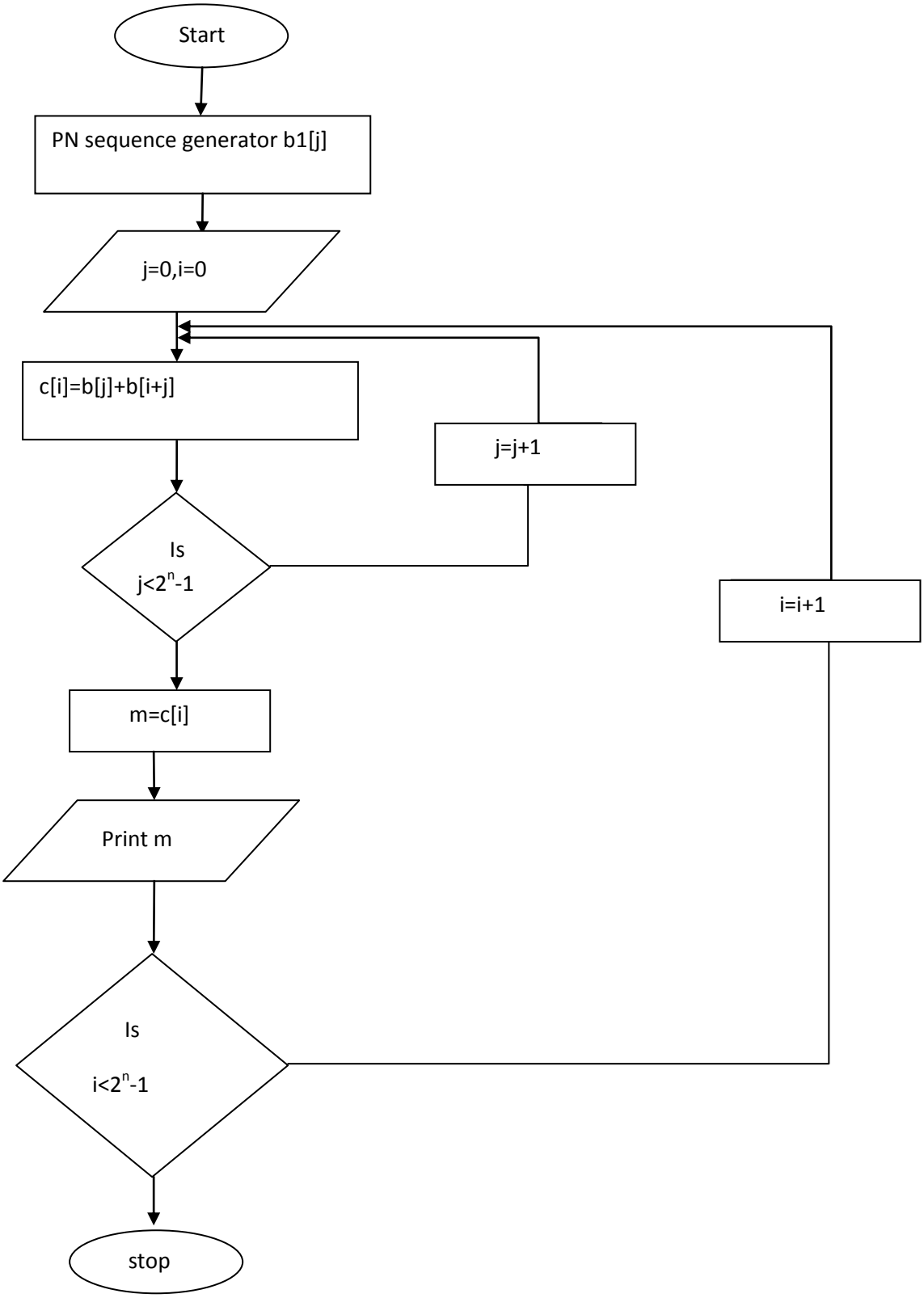
Figure 12:- Inverting Amplifier.



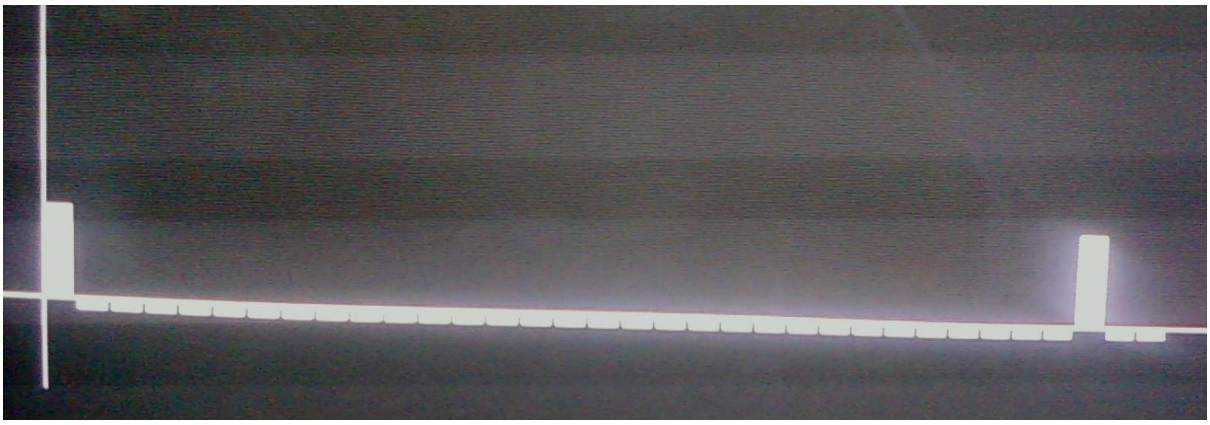
Snapshot4:- This snapshot shows Autocorrelation of PN-Sequence of Length 31.

2.5.2 SOFTWARE SIMULATION

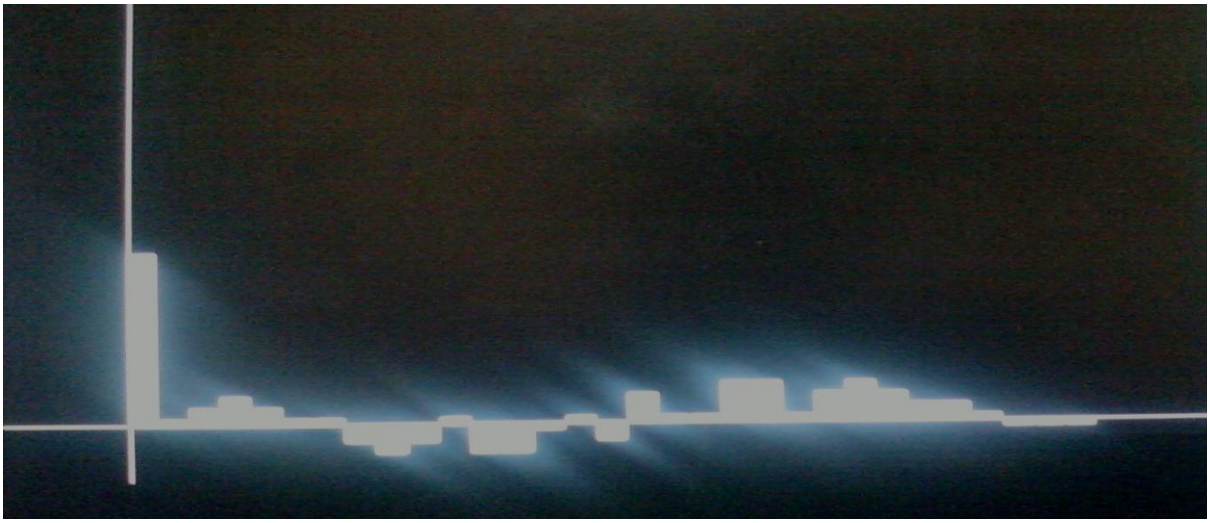
a. Periodic and aperiodic autocorrelation of the sequences were obtained in C++.
b. Waveforms of these Periodic and aperiodic autocorrelation were premeditated.
c. The Flow chart for the code of Autocorrelation is discussed below for better understanding of how it is being calculated.



OUTPUT and WAVEFORM



Snapshot5:- This snapshot shows Periodic Autocorrelation of 31 lengths PN-Sequence.

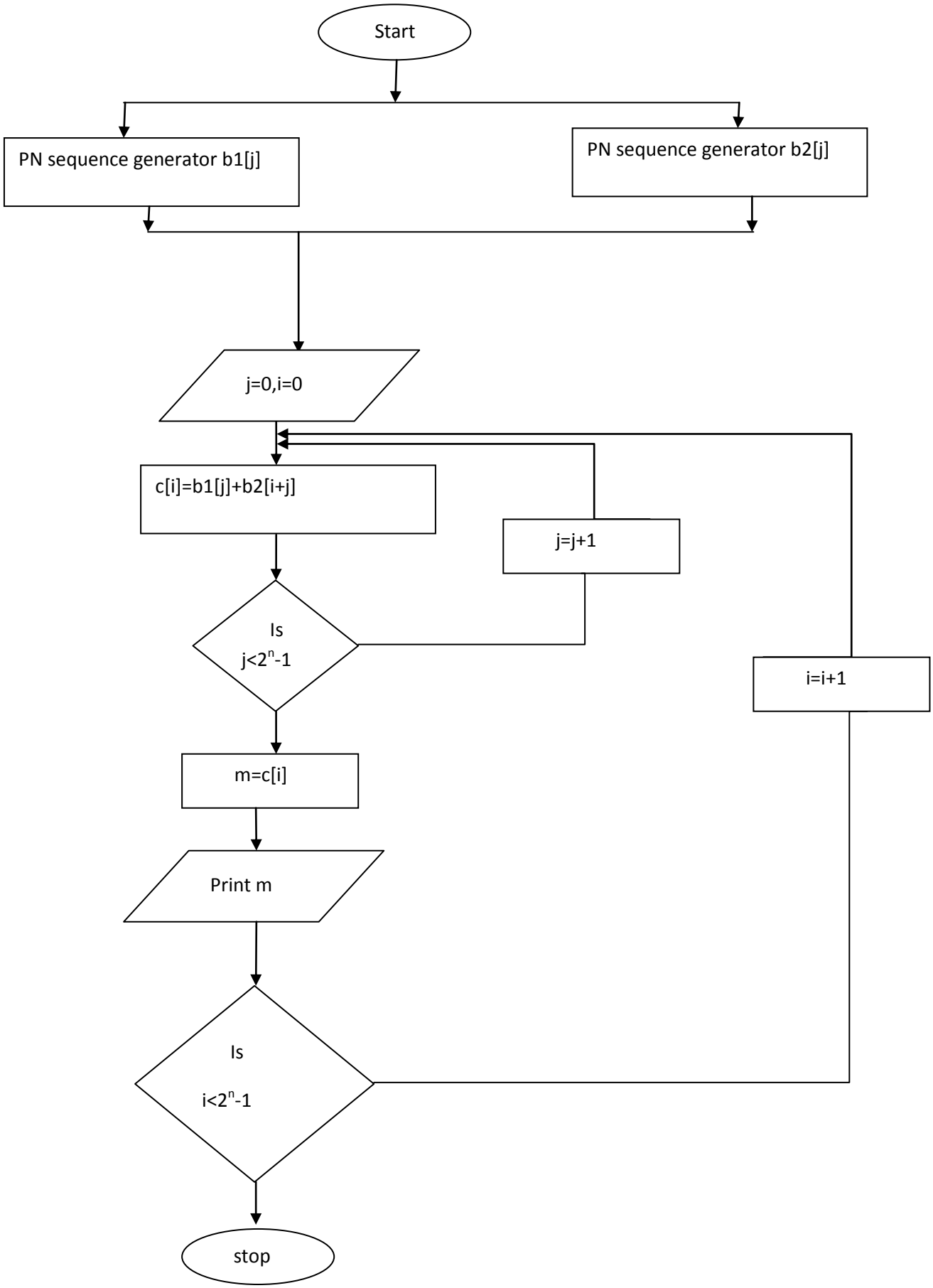


Snapshot6:- This snapshot shows Aperiodic Autocorrelation of 31 lengths PN-Sequence.

2.6 PN-Seq. CROSS-CORRELATION

The Cross-Correlation is calculated using the Dixon's method (for better understanding of Dixon's method refer to book entitled "Communication Systems" by *A. Bruce Carlson , Paul B. Crilly and Janet C. Rutledge* [e].)

Using the concept of Dixon's method and employing them in programming the following Flow chart was prepared, for programming codes look into appendix [c].


Start
PN sequence generator b1[j]
PN sequence generator b2[j]
j=0,i=0
c[i]=b1[j]+b2[i+j]
j=j+1
Is
j<2n-1
i=i+1
m=c[i]
Print m
Is
i<2n-1
stop

The cross-correlation is calculated using Computer simulation and the value is used to draw the below graph on Microsoft Excel.

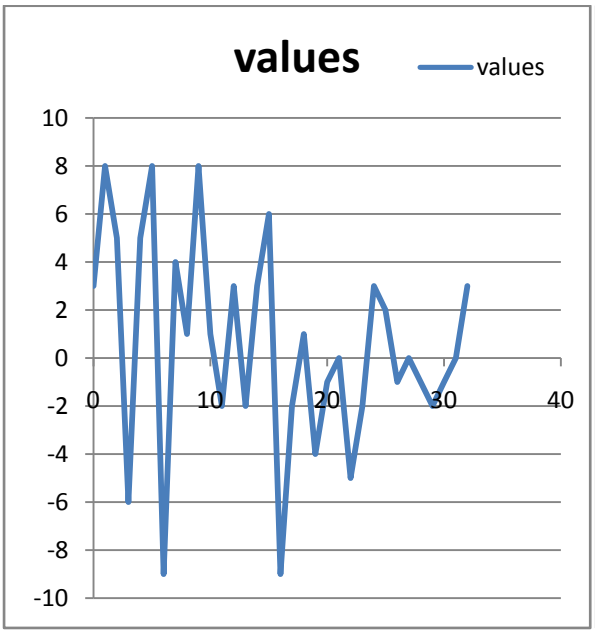


Figure 13:- Cross-Correlation of PN-Seq. generated by (x^5+x^3+1) and (x^5+x^2+1)

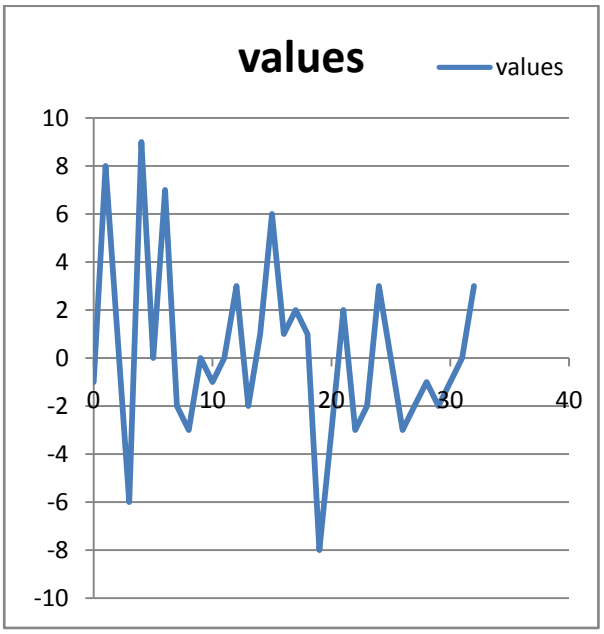


Figure 14:- Cross-Correlation of PN-Seq. generated by (x^5+x^3+1) and (x^5+x^4+x^3+x^2+1)

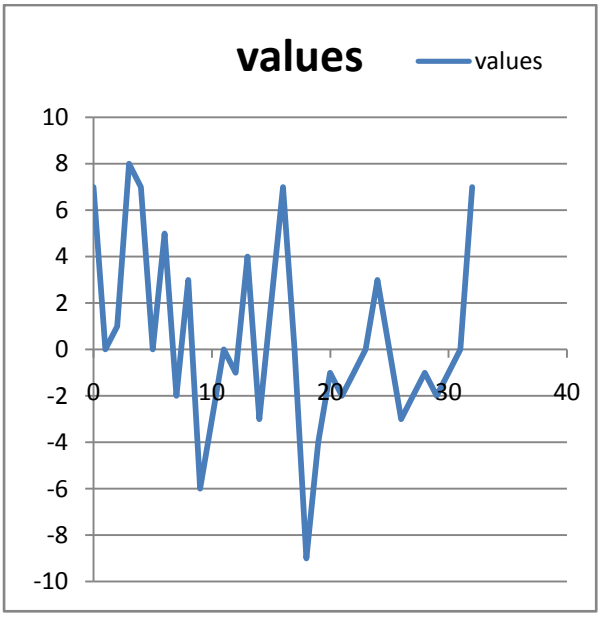


Figure 15:- Cross-Correlation of PN-Seq. generated by (x^5+x^2+1) and (x^5+x^4+x^3+x^2+1)

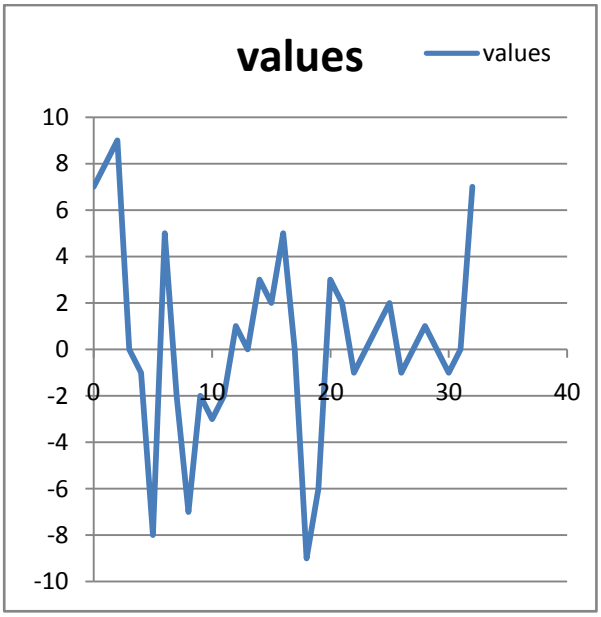


Figure 16:- Cross-Correlation of PN-Seq. generated by (x^5+x^2+1) and (x^5+x^4+x^2+1)

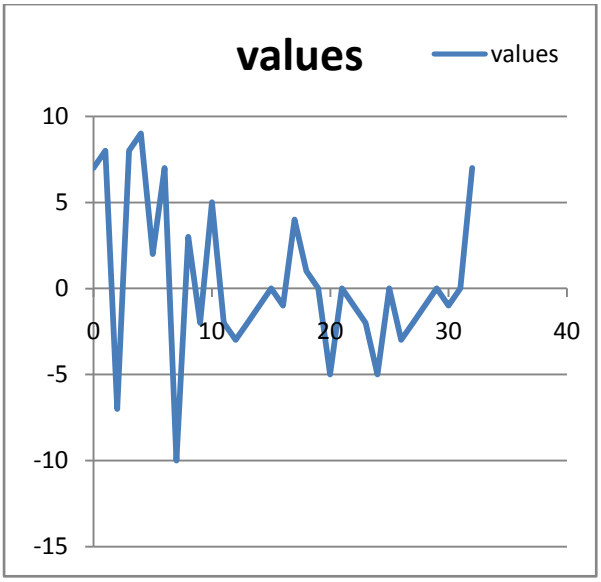


Figure 17:- Cross-Correlation of PN-Seq. generated by (x^5+x^4+x^2+x+1) and (x^5+x^4+x^3+x^2+1)

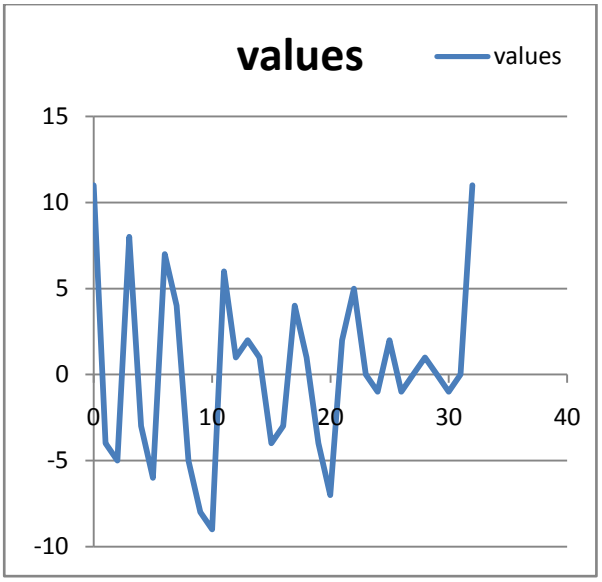


Figure 18:-Cross-Correlation of PN-Seq. generated by (x^5+x^4+x^2+x+1) and (x^5+x^4+x^3+x+1)

CHAPTER 3

GOLD CODES

3.1 INTRODUCTION

A **Gold code**, also known as **Gold sequence**, is a type of binary sequence, used in telecommunication (CDMA) and satellite navigation (GPS). Gold codes are named after Robert Gold (The paper entitled "Characteristic linear sequences and their co-set functions[f]" was published in 1965, gave a brief mathematics behind GOLD CODES). Gold codes have bounded small cross-correlations within a set, which is useful when multiple devices are broadcasting in the same range.

3.2 GENERATION of GOLD CODES

These are constructed by EXOR-ing two *m*-sequences of the same length with each other. Thus, for a Gold sequence of length $m = 2^l-1$, one uses two LFSR, each of length 2^l-1. If the LSFRs are chosen appropriately, Gold sequences have better cross-correlation properties than maximum length LSFR sequences.

Gold sequences help generate more sequences out of a pair of m-sequences giving now many more different sequences to have multiple users. Gold sequences are based on ***preferred pairs*** m-sequences.

Gold showed that for certain well-chosen *m*-sequences, the cross correlation only takes on three possible values, namely -1, $-t$ or t-2. Two such sequences are called preferred sequences. Here t depends solely on the length of the LFSR used. In fact, for a LFSR with l memory elements,

if l is odd, $t = 2^{(l+1)/2} + 1$, and

if l is even, $t = 2^{(l+2)/2} + 1$.

Remember m-sequences gave only one sequence of length 2^l-1. By combining two of these sequences, we can obtain up to 31 (2^l-1) plus the two m-sequences themselves, hence we can generate upto 33 (2^l+1) sequences (each one of length 2^l-1) that can be used to spread

different input messages (different users CDMA). The m-sequence pair plus the Gold sequences form the available sequences to use in DSSS. The wanted property about Gold codes is that they are balanced (i.e. same number of 1 and -1s).

a) For example, let us take two preferred pair of m-sequence of length 31; $1+x^2+x^5$ and $1+x+x^2+x^4+x^5$. This example is taken from "*CDMA-Principles of Spread Spectrum Communication*" *by Andrew J. Veterbi* [g].

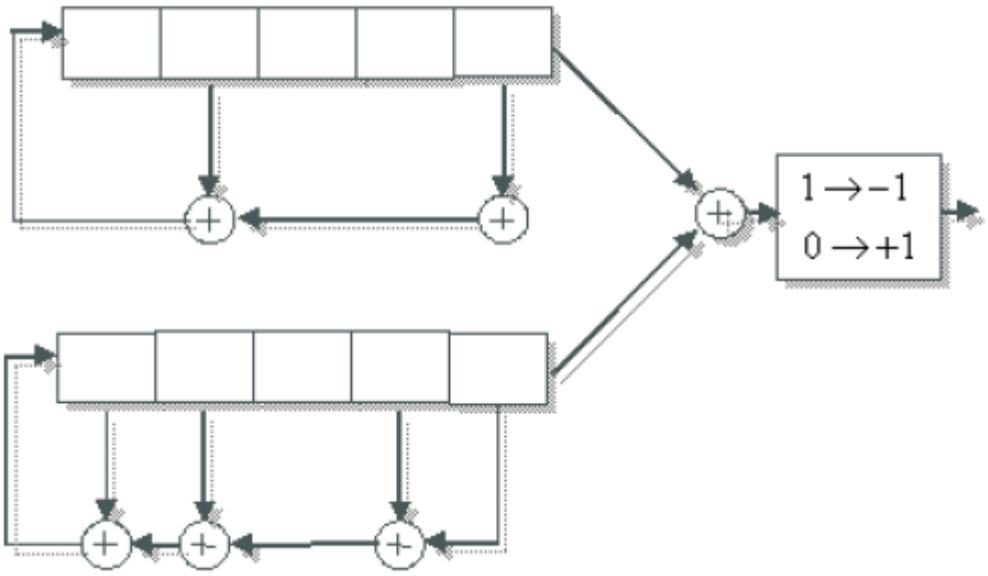


***Figure 19**. Block diagram of GOLD codes generation*

3.3 Comparison between PRN-Seq. and GOLD code

i. The cross-correlation of different PN-Seq. and different GOLD codes is calculated.

ii. The normalized value of cross-correlation is noted in a tabular form. This shows that at the smaller value of sequence length the normalised value of both (PN-Seq. and GOLD codes) are nearly same. But, for larger value of sequence length, the

normalised value for GOLD codes is less as compared to PN-Seq., as for PN-Seq. the normalised value keeps on increasing as the sequence length increase.

iii. The table is shown for clear verification. This table is taken from a paper entitled "Code Sequences for Direct Sequence CDMA" *by Frank Kamperman"* [h].

Table for comparing GOLD and PN Sequences

No. of LFSR elements, l	Sequence Length, m	No. of m-sequence	Max. cross-correlation of m-sequence Normalised	Cross-Correlation of GOLD Codes, t	Normalised Cross-Correlation $t/(2^l-1)$
3	7	2	0.71	5	0.71
4	15	2	0.69	9	0.69
5	31	6	0.35	9	0.29
6	63	6	0.36	17	0.27
7	127	18	0.32	17	0.13
8	255	16	0.37	33	0.13
10	1023	60	0.37	65	0.03

The above table shows that, with increase in length of sequences the cross-correlation decreases for both PRN and GOLD. But GOLD has upper hand as it provides the designer a larger family of sequences for the preferred length.

Thus, we can say that GOLD codes are a better option when used for large number of users, because it can give more number of sequences so that we can allot them to multiple users.

3.4 PROJECT WORK DONE ON GOLD CODES

The GOLD codes are generated using preferred pairs. The preferred pairs are formed from the available m-sequences. For length of 31 there are 6 m-sequences possible. Hence we can have 36 pairs. But all of them are not suitable to generate gold sequences as they do not

satisfy the basic properties of gold sequences. Hence only those pairs are preferred which provide sequences with good balanced property (equal number of 1s and 0s).

The preferred pairs used for generating GOLD sequences are :-

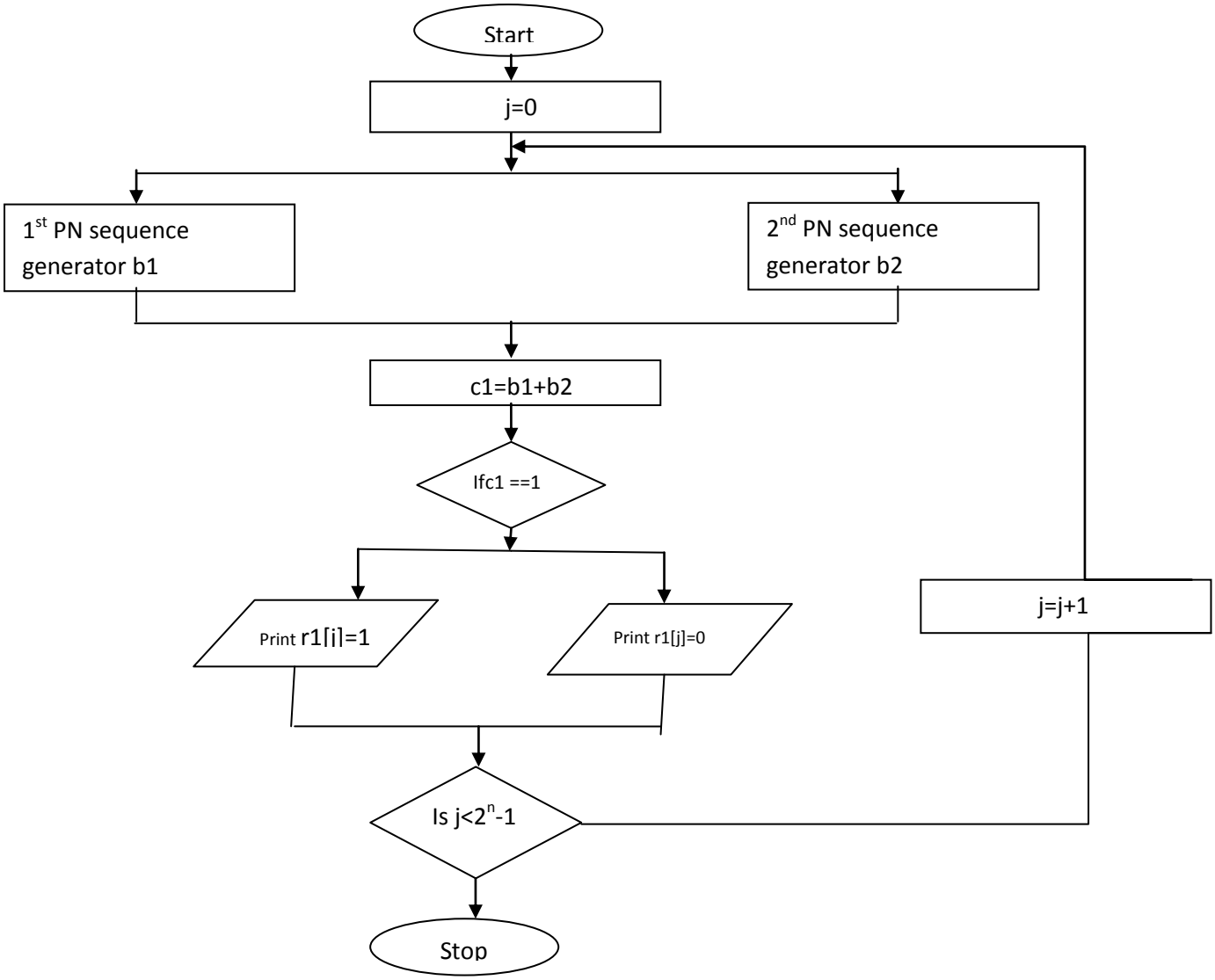
Code **A** :- $X^5 + X^3 + 1$ and $X^5 + X^2 + 1$.

Code **B** :- $X^5 + X^3 + 1$ and $X^5 + X^4 + X^3 + X^2 + 1$.

Code **C** :- $X^5 + X^2 + 1$ and $X^5 + X^3 + X^2 + X + 1$.

Code **D** :- $X^5 + X^4 + X^3 + X^2 + 1$ and $X^5 + X^3 + X^2 + X + 1$.

Using any of the preferred pairs we can generate the GOLD Codes in C++, the Flow chart for the Generating Code is:-



OUTPUT

Using the above preferred pairs we generated different gold codes of length 31.

Code **A**: 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0

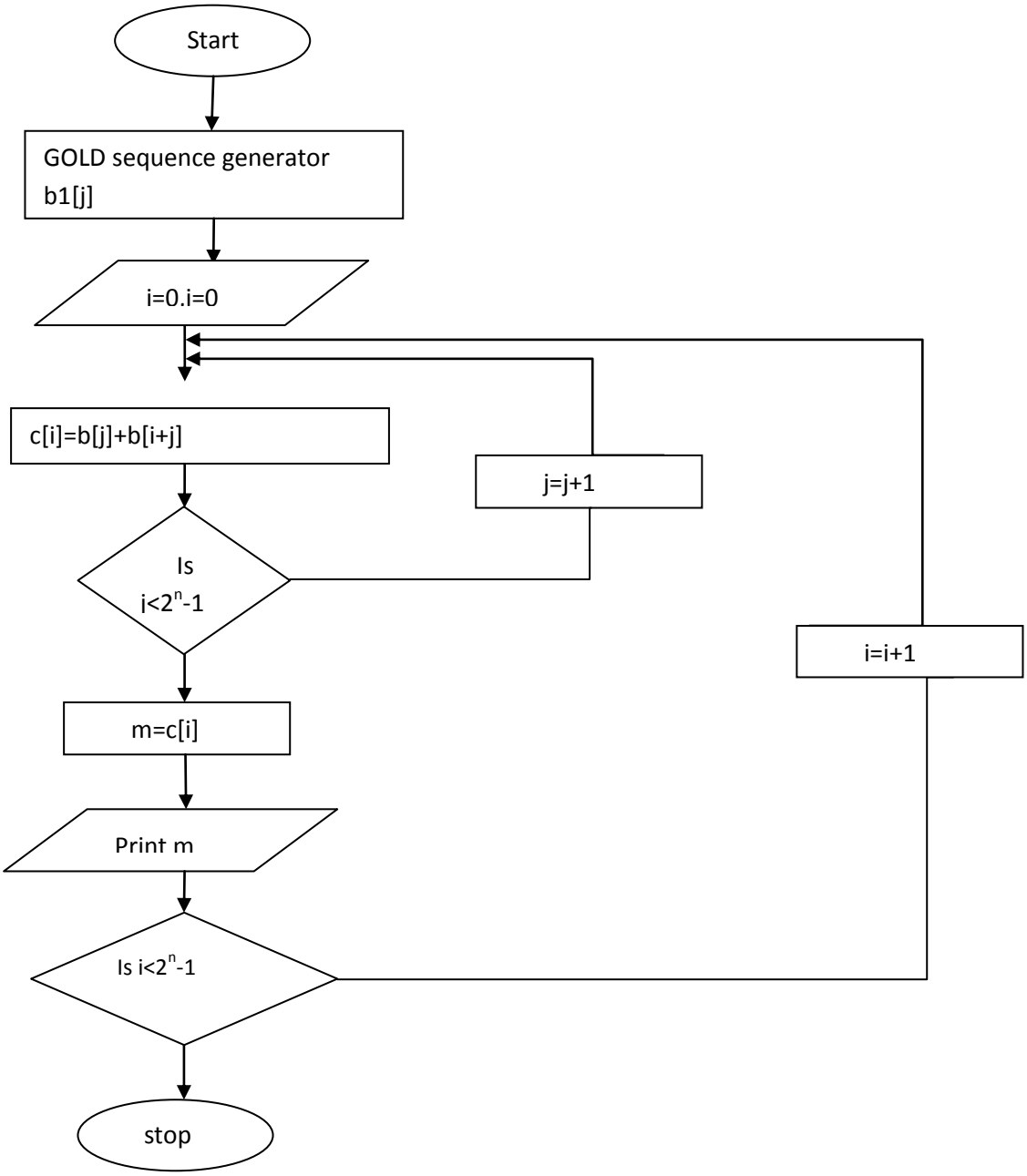
Code **B**: 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 0 0

Code **C**: 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 0 0 1 0

Code **D**: 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 0 1 1 1 0 1 0 1 0

2.5 GOLD Codes autocorrelation

The Flow chart for the GOLD Code autocorrelation is:-



The result of Computer simulation using the codes in Appendix and plotting them in Excel we get the following graphs:-

OUTPUT

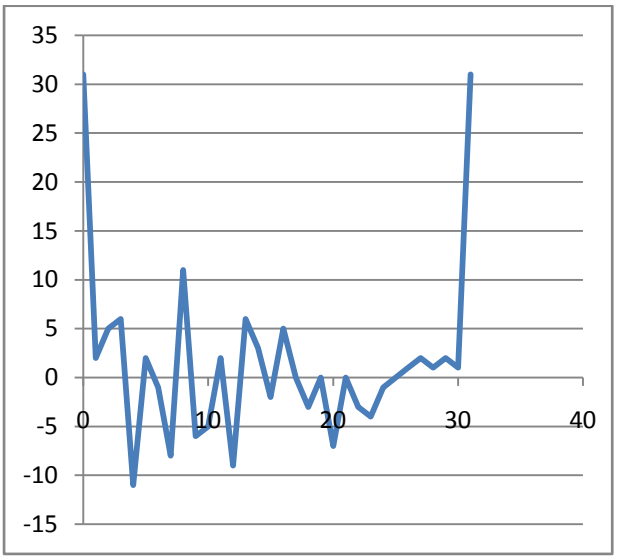


Figure 20: Autocorrelation of Code A ($X^5 + X^3 + 1$ and $X^5 + X^2 + 1$)

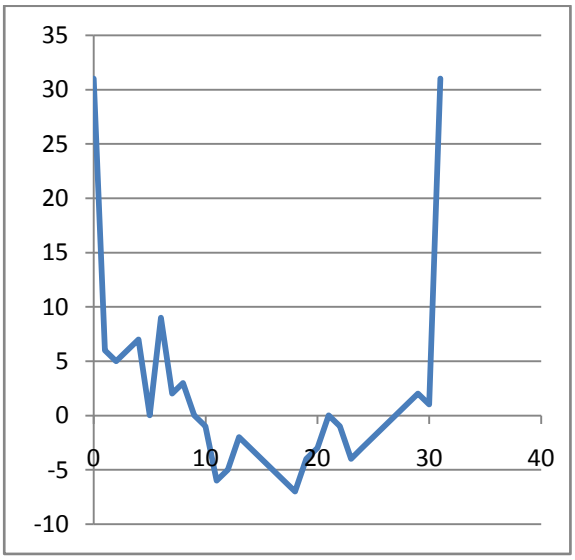


Figure 21: Autocorrelation of Code B ($X^5 + X^3 + 1$ and $X^5 + X^4 + X^3 + X^2 + 1$)

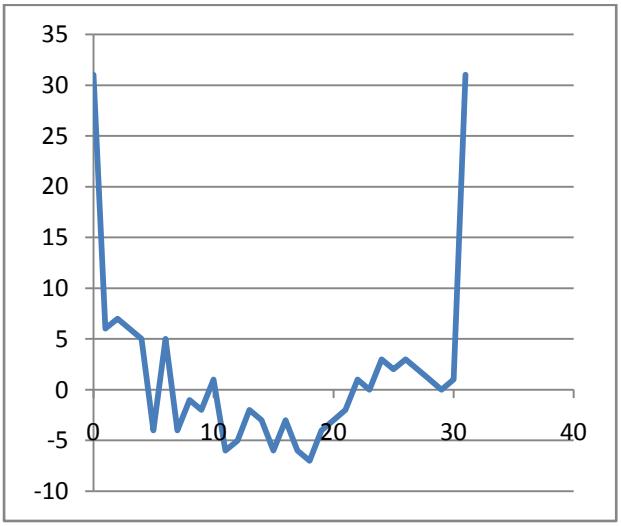


Figure 22: Autocorrelation of Code C ($X^5 + X^2 + 1$ and $X^5 + X^3 + X^2 + X + 1$)

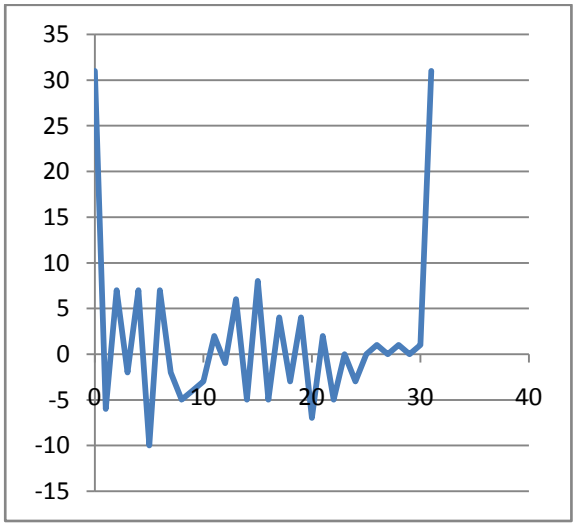
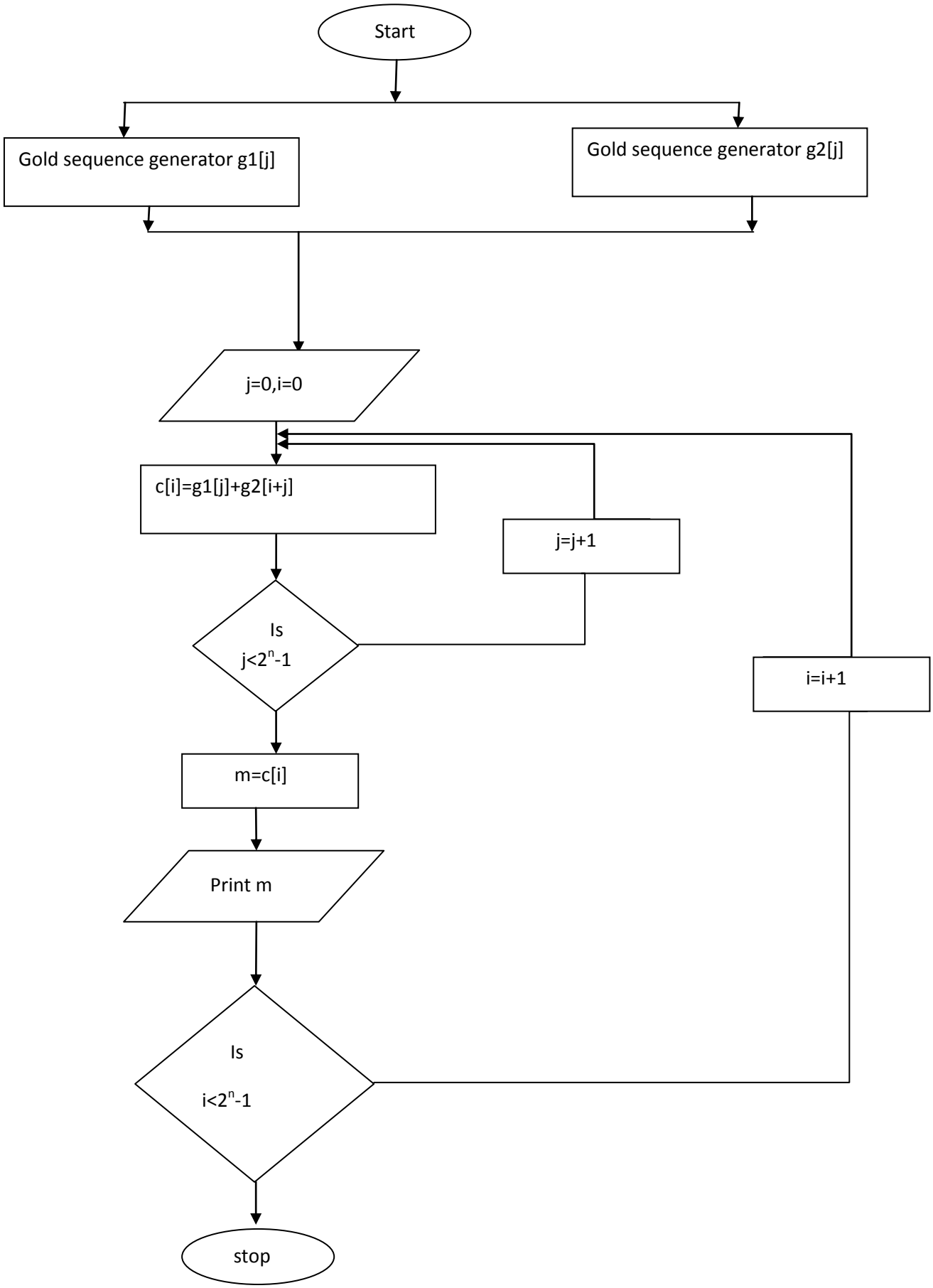


Figure 23: Autocorrelation of Code D ($X^5 + X^4 + X^3 + X^2 + 1$ and $X^5 + X^3 + X^2 + X + 1$)

2.5 GOLD Codes cross-correlation

The method used for finding the Cross-Correlation is same as PRN Sequence and the Flow chart for the GOLD Code cross-correlation used is:-


Start
Gold sequence generator g1[j]
Gold sequence generator g2[j]
j=0,i=0
c[i]=g1[j]+g2[i+j]
j=j+1
Is
j<2^n-1
i=i+1
m=c[i]
Print m
Is
i<2^n-1
stop

The result of Computer simulation using the codes in Appendix is :-

OUTPUT

Using the codes generated by Flow chart, and finally simulating them on C++. We get the values for the Cross-Correlation and it is plotted on Microsoft Excel.

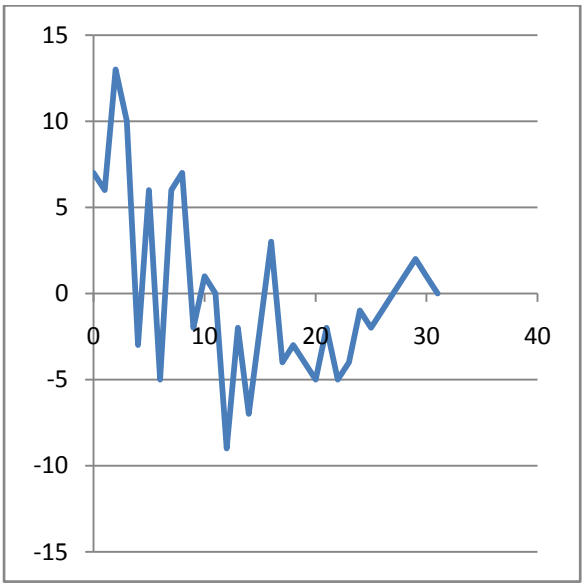


Figure 24: Cross-correlation of Code A&B

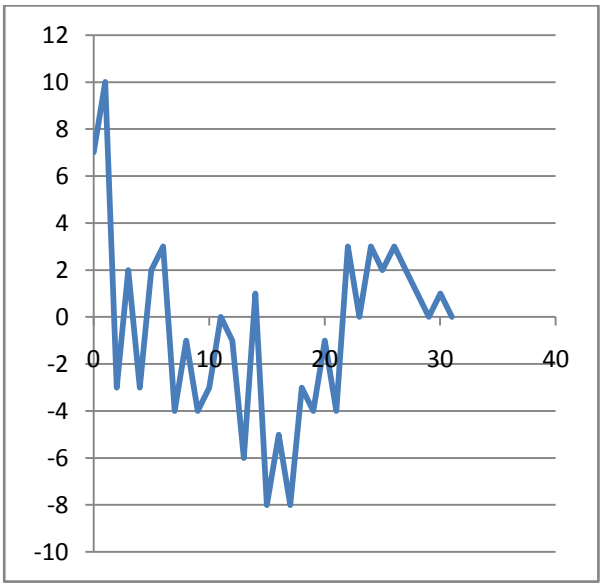


Figure 25: Cross-correlation of Code A&C

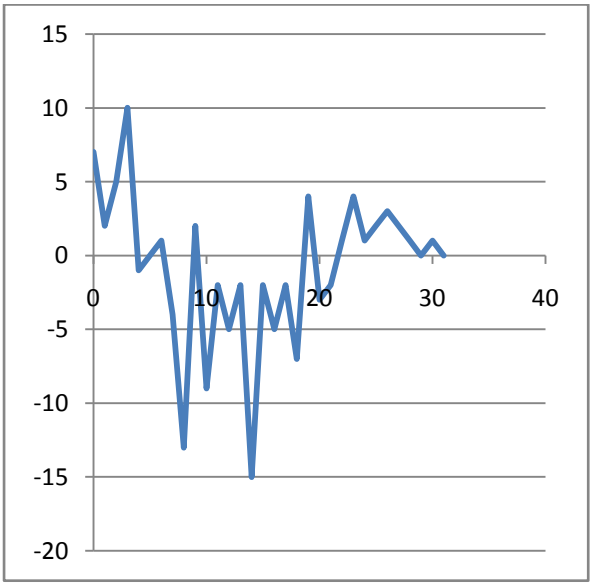


Figure 26: Cross-correlation of Code C&B

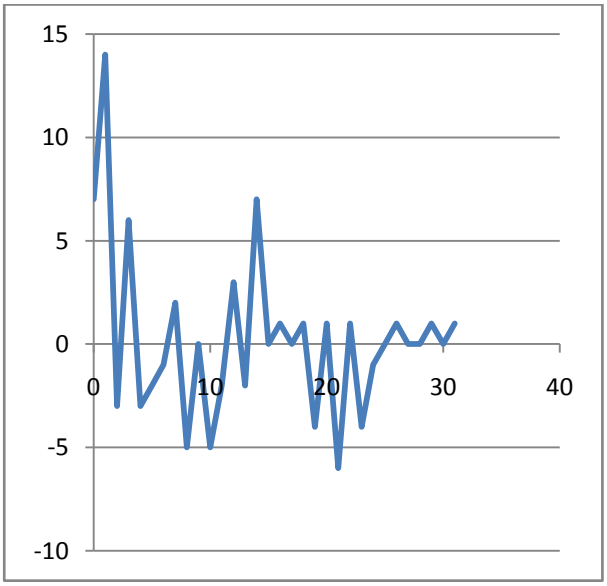


Figure 27: Cross-correlation of Code A&D

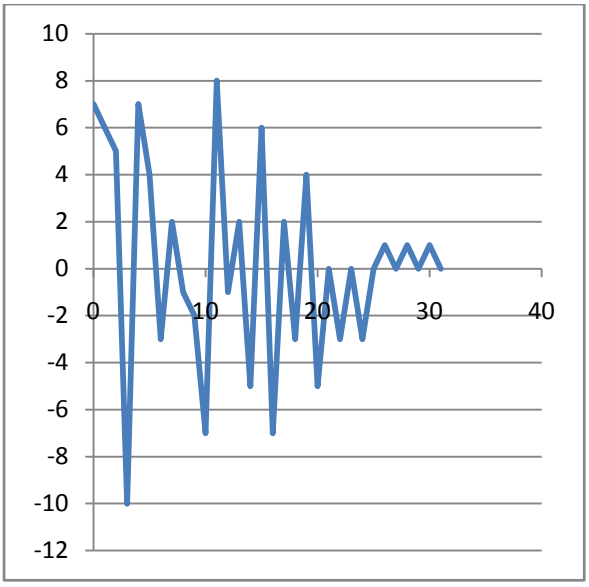


Figure 28: Cross-correlation of Code D&B

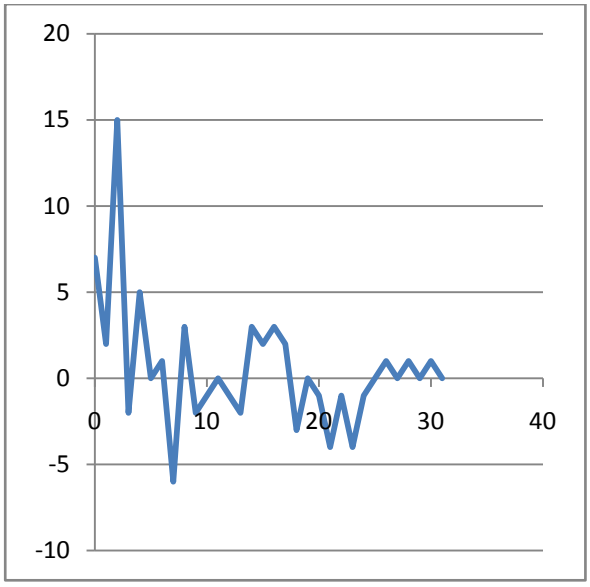


Figure 29: Cross-correlation of Code D&C

CHAPTER 4

TRANSMITTER and RECEIVER for DSSS

4.1 INTRODUCTION

Bandwidth spreading by direct modulation of signals by a wideband spread signal (also called code) is called direct sequence spread spectrum (DS SS). The DSSS signal is then modulated by a carrier before final transmission. In DSSS, the base band signals are usually called bits, and the code bits are called chips. Typically, the baseband signal bandwidth is multiplies several times by the spreading signals. In other words, the chip rate is much higher than the bit rate. The spreading signal sequence is unique for a transmitter, and the same chip sequence is used at the receiver to re-construct the signals (data bits). Correlation is used to synchronize the received spread signals (that contain data) with the locally generated code. At maximum received signal strength, correlation said to have occurred. The receiver then enters the tracking mode, such that the spread signal modulated signals are received without interruption.

4.2 DSSS Transmitter

In the DSSS transmitter, a code generator is a pseudo random generator that generates a known pseudo noise code sequence. Normally, the code has finite length (say 1024 chips), and repeats periodically (For more detail on the topic one can refer to the Book entitled "Principles of spread spectrum" by *Don j. Torrieri* [j]). The requirements for a good PN code was already discussed in the previous chapter named PRN-Sequences.

An XOR gate can be used for spreading the data bits. The input, code, and the resulting output are displayed in the **figure 30** on the next page :

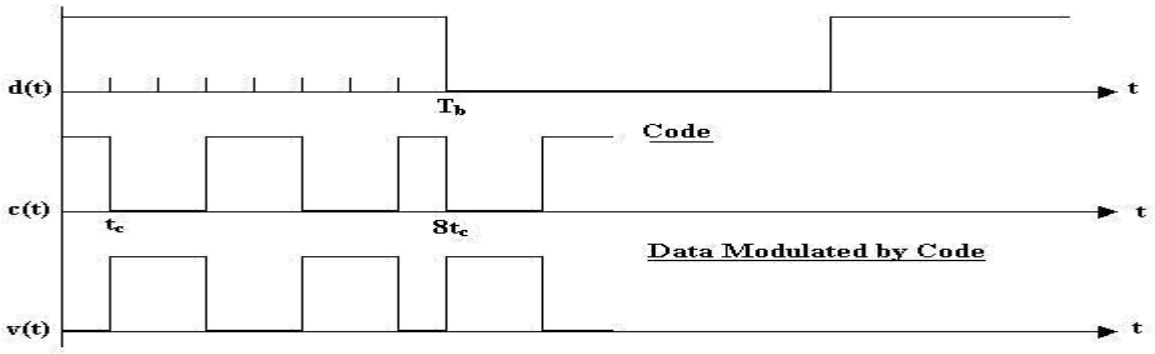


Figure 30*.Generation of Modulated DATA*

In the figure shown above, each data bit is coded with 8 chips. In practice, this would be much higher, of the order of 1024 or even more (For more detail on the topic one can refer to the Book entitled "Principles of spread spectrum" by *Don j. Torrieri* [j]). Higher the number of chips per bit, higher will be the processing gain. Processing gain is defined below:

Processing Gain: One important parameter of DS SS receiver is the processing gain. Consider a data rate of 10KBPS, and Chip rate of 1MBPS. The processing gain is given by 10 log [rc/rb], where rc is the chip rate, and rb is the data rate. For a chip rate of 1MBPS, and a data rate of 1KBPS, the processing gain is 10log[1000] or 30dB. The processing gain is a measure of immunity to noise, and jamming signals. Higher the processing gain, more the band spread of the signals.

Higher processing gain results in greater immunity to noise, and interfering signals.

After spreading, the signals are unconverted and transmitted.

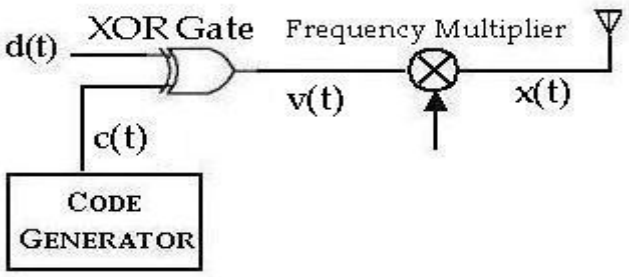


Figure 31: DSSS Transmitter

Where, d(t) is the input data bits, c(t) is the code bits, x(t) is the frequency converted signal, ready for transmission.

A note about why frequency up conversion is required for radio transmission: Base band, and very low frequencies are susceptible to heavy attenuation during transmission. In addition, imagine every transmitter transmitting in the base band frequencies. It is practically impossible for everyone to transmit in base band frequencies (A base band frequency is the frequency spectrum that is occupied by the un-modulated signals). Hence up-conversion of frequency is normally done to comply with the transmission requirements.

4.2.1 Implementation of DSSS Transmitter

a. First of all, we generated the PRN-Sequence of desired length (in our case length was 31).
b. Then the DATA was XOR-ed with PRN-Sequence. Hence, resulting into the spread signal.
c. Each bit of DATA was XOR-ed with all the 31 bit of sequence. Hence, the resulting signal spreads up to the length equals to product of No. of DATA bits and length of PRN-Seq.(31).

For example:- Let us take an example of 7-bit PRN-Seq. and a 2-bit DATA.

7-bit PRN-Seq. used here is:-

1 1 1 0 0 1 0

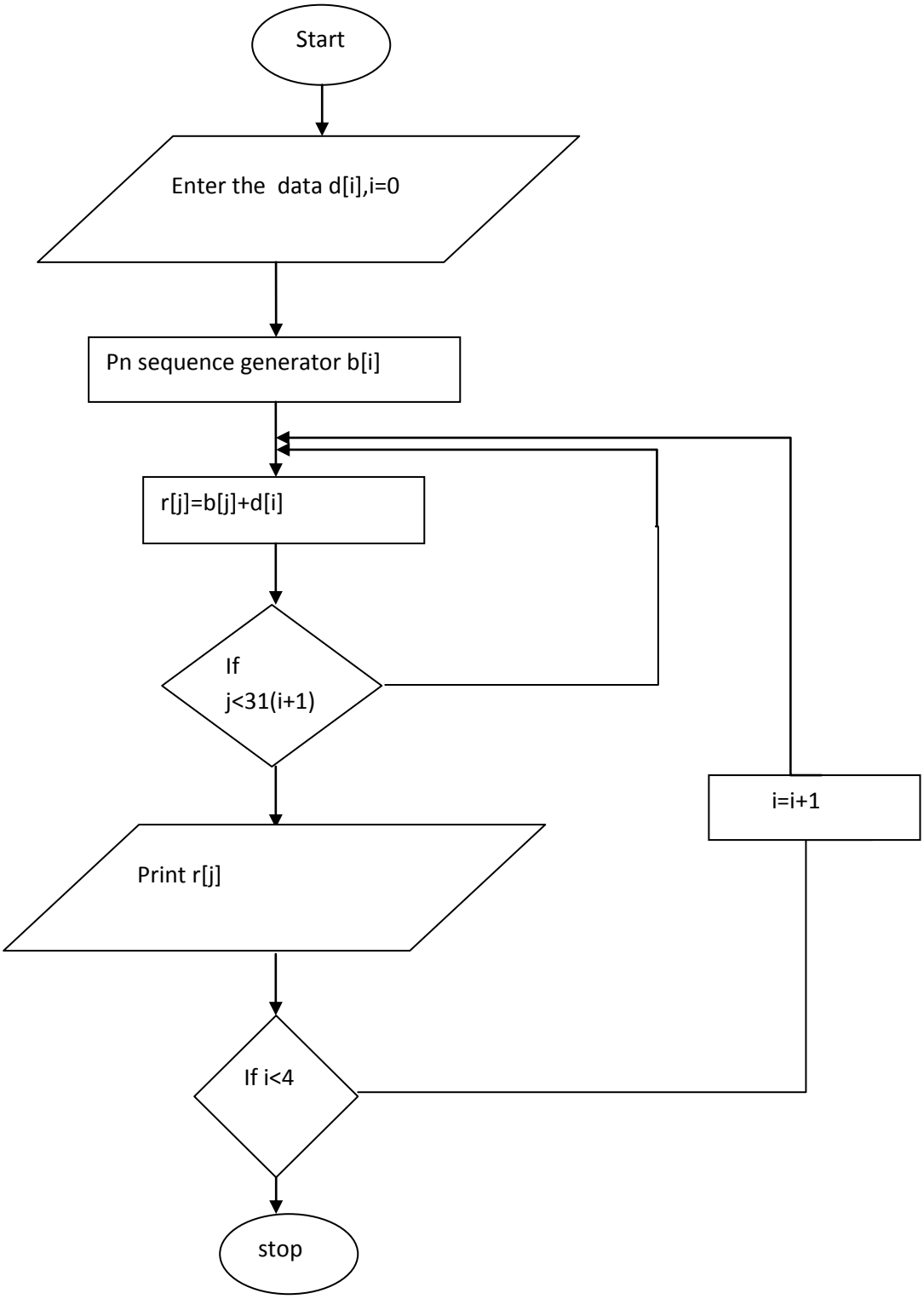
2-bit DATA used here is:-

1 0

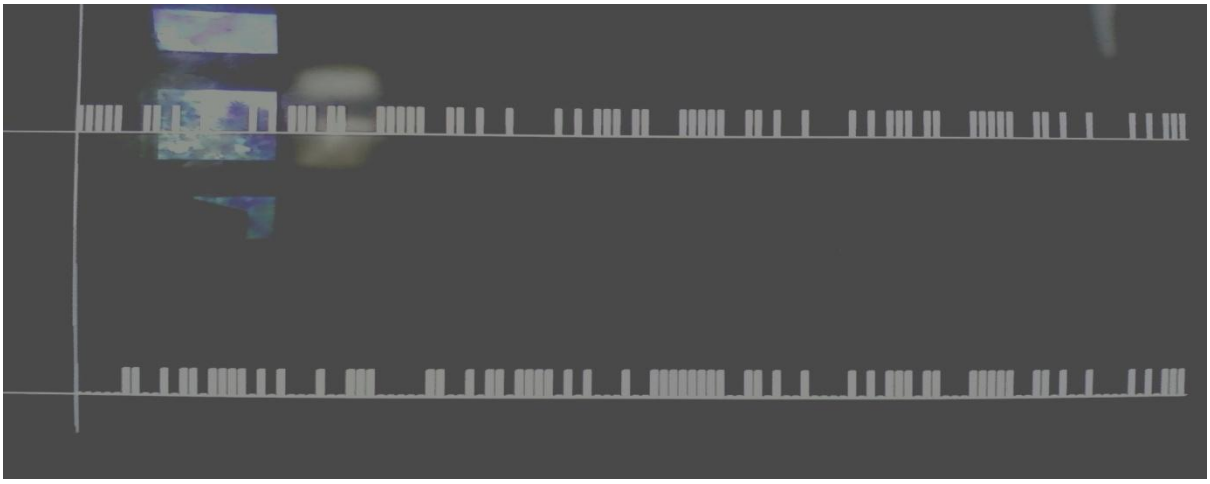
The spread signal is calculated as follows:-

PN-Seq.	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**0**
DATA	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
SPREAD SIGNAL	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**0**

The flow chart for the generation of Spreading Sequence used is:-



OUTPUT and WAVEFORM



Snapshot 7:- This snapshot shows Modulation of PN-Seq. and DATA **1 1 0 0**.

4.3 DSSS Receiver

A simplified DSSS receiver block diagram is shown below.

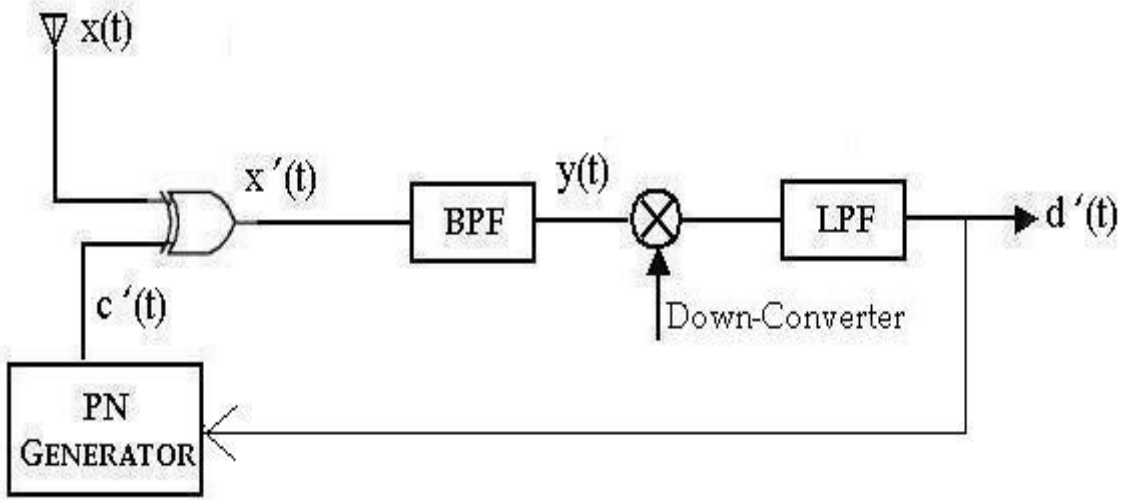


Figure 32: DSSS Receiver

It consists of a PRN generator that feeds the matching chip sequence to an XOR gate to reproduce the original bit sequence. The PRN generator is driven by an error signal from the output of the LPF, so that chip timing is adjusted to produce maximum signal threshold. Normally, the acquisition of the data is done through a two step process. The first is acquisition, and the second is tracking. Acquisition refers to acquiring the chip timing of the received signals. This may further be sub-divided into course acquisition, and fine acquisition. The two are differentiated by the amount of chip timing adjustment. Once the acquisition is achieved, then the received signals must be tracked properly. Otherwise, you may lose the lock, resulting in loss of data bits. As with conventional receiver operation, an error voltage at the output of the LPF (or an Integrator) provides necessary correction to the PRN Generator.

4.3.1 Implementation of DSSS Receiver

a. On reception of spread signal, the signal is matched with the locally generated PN-sequence.

b. The matching is done with a matched filter, which matches the local PN-sequence with the received signal sequence.

c. On full matching of both received signal sequence and local PN-sequence we get a large positive or a large negative value, in our case we got +31 if 1 is received or -31 if 0 is received.

For example:- Let us take an example of 7-bit PN-Seq. and a 2-bit DATA.

7-bit PN-Seq. used here is:- **1 1 1 0 0 1 0**

2-bit DATA used here is:- **1 0**

Spread signal is:- **0 0 0 1 1 0 1 1 1 1 0 0 1 0**

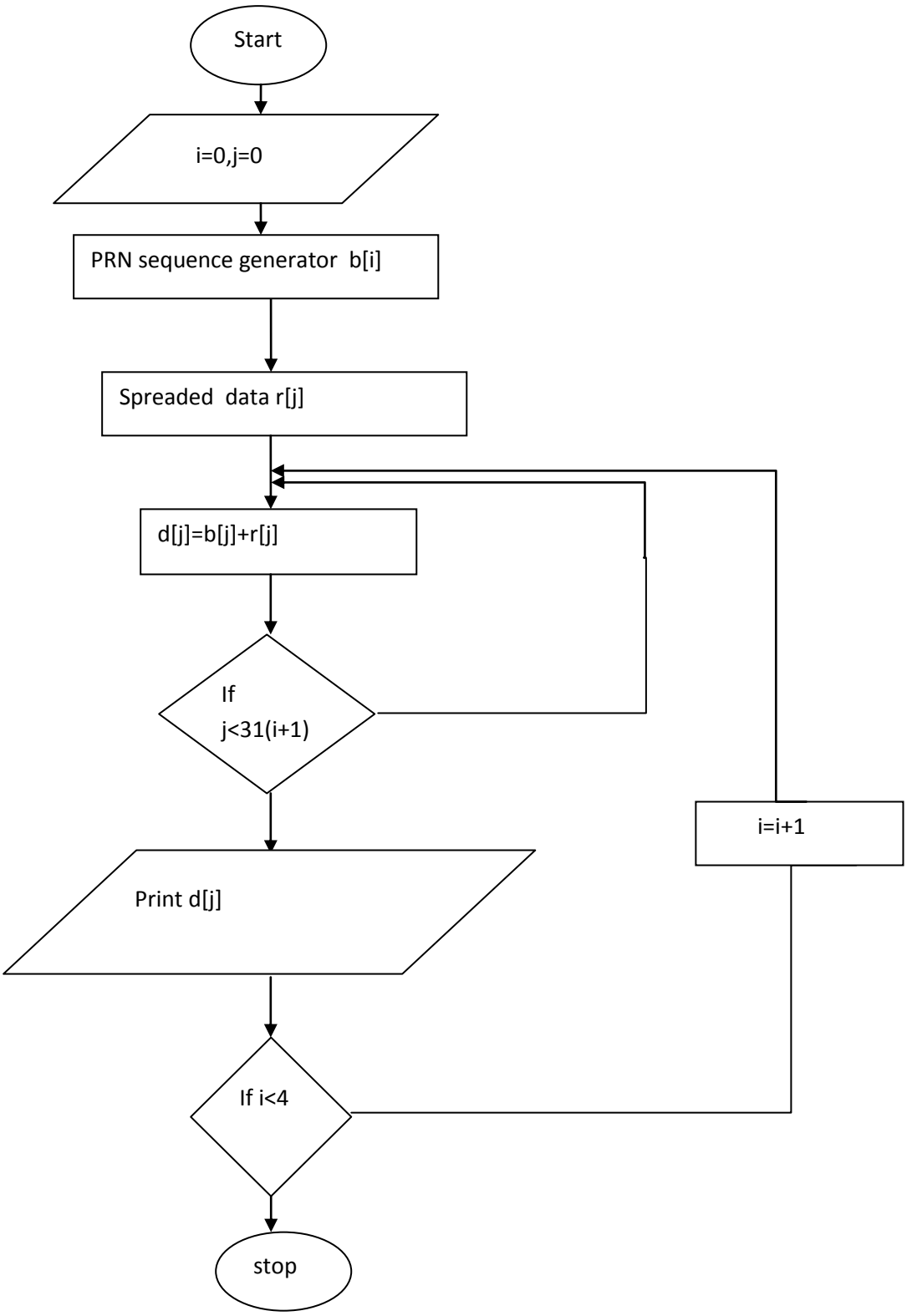
Note:- for calculating zero is replaced by -1 and 1 remains 1.

	0	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**0**
1	0													
0	1	0												

1	0	1	0											
0	0	0	1	0										
2	1	0	0	1	0									
0	1	1	0	0	1	0								
7	1	1	1	0	0	1	0							
-1		1	1	1	0	0	1	0						
1			1	1	1	0	0	1	0					
-1				1	1	1	0	0	1	0				
1					1	1	1	0	0	1	0			
2						1	1	1	0	0	1	0		
-1							1	1	1	0	0	1	0	
-7								1	1	1	0	0	1	0
0									1	1	1	0	0	1
-2										1	1	1	0	0
0											1	1	1	0
-1												1	1	1
0													1	1
-1														1

And plotting these values we get the following graph, which shows that, the two peaks +7 corresponds to DATA bit 1 and -7 corresponds to DATA bit 0.

Using the above concept in C++, we draw the de-spreading part, the Flow chart for the codes is:-



And plotting these values we get the following graph, which shows that, the two peaks +7 corresponds to DATA bit 1 and -7 corresponds to DATA bit 0.

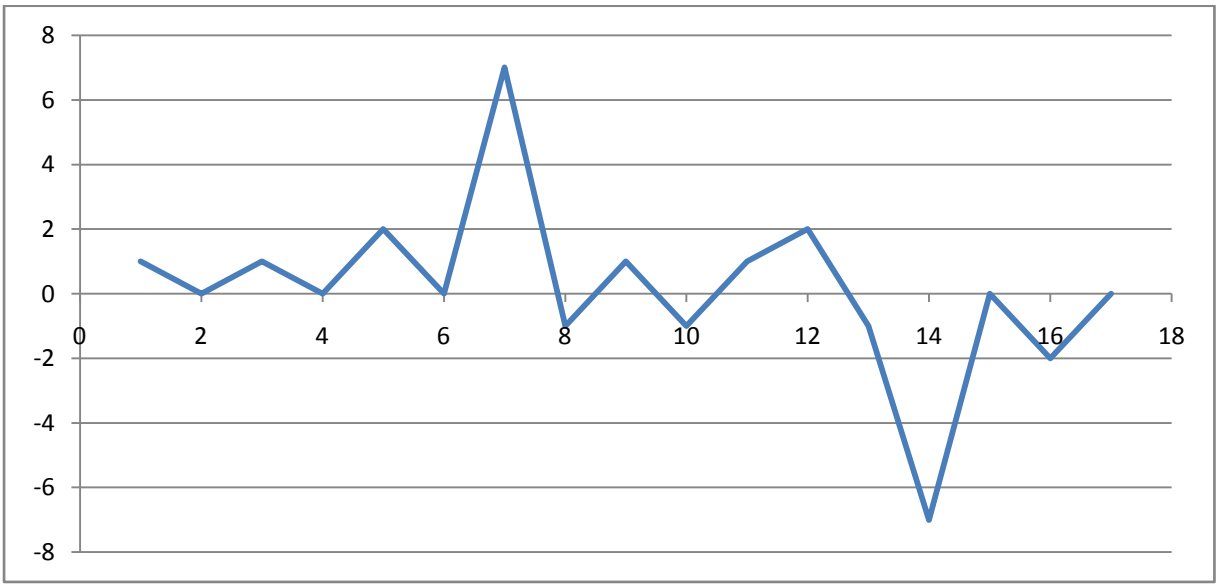
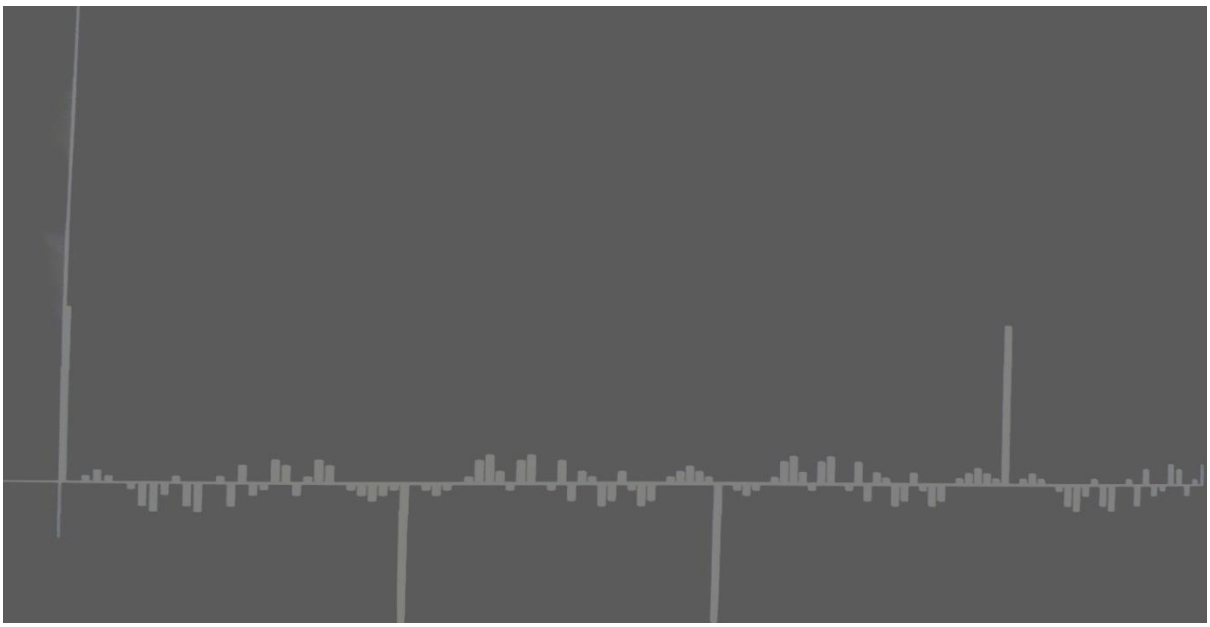


Figure 33: Matched filter output of 7 length sequence with DATA 1 and 0.

OUTPUT and WAVEFORM



Snapshot 8:- This snapshot shows Matched filter result for DATA 1 0 0 1.

4.4 Recovery of Spread Spectrum Signals: Timing Signals

In all cases of SS receivers, faithful recovery of the transmitted signals require the following:

i. **Correlation Interval Synchronization**: Receiving bits is achieved by proper correlator (or integrator) timing. Proper start/stop times for correlator are required for minimizing the received bit errors.
ii. **SS Generator Synchronization**: Timing signals are required to control the SS wave form generator signals. Direct Sequence systems employ a clock ticking at the chip rate $1/t_c$, and FH systems have a clock operating at the hopping rate $1/t_h$.
iii. **Carrier Synchronization**: Faithful reproduction of the transmitted signals to baseband requires down-conversion, and demodulation. This can be achieved only if the locally generated frequency and phase are in sync with the received carrier frequency.

4.6 Data Retrieval for Multiple Users

In a system with multiple users transmitting at the same time. The signals are overlaid. In other words, if one does not know the code, then it hears some signal like random noise. However, for the receiver with the correct code, it will be able to extract from the noise-like data its expected information.

Each chipping sequence, or code, is a sequence of 0's and 1's. We will represent that by a vector. Essentially replace 1 by a corresponding 1 and replace 0 by a corresponding -1. For example, the Baker code 10110111000 is represented by a vector of dimension 11,
represented by (1,-1, 1, 1,-1, 1, 1, 1,-1,-1,-1).Similarly, we also represent a signal bit of 1 by 1 and a data bit 0 by -1. Now the XOR of two bits is exactly multiplication in this new system. The XOR of the chipping sequence of the input signal is exactly the bit-wise product in the vector space. The extraction of the data is the *dot product* of two vectors: take the bit-wise product and sum up all the values.
The sender *A* has data *Ad* = 1 (stands for bit 1). It will take the product of *Ad* with the chipping sequence *Ak* = 010011 = (-1, 1,-1,-1, 1, 1) *Ad* . *Ak* = *Ad* * (-1,1,-1,-1- 1- 1)

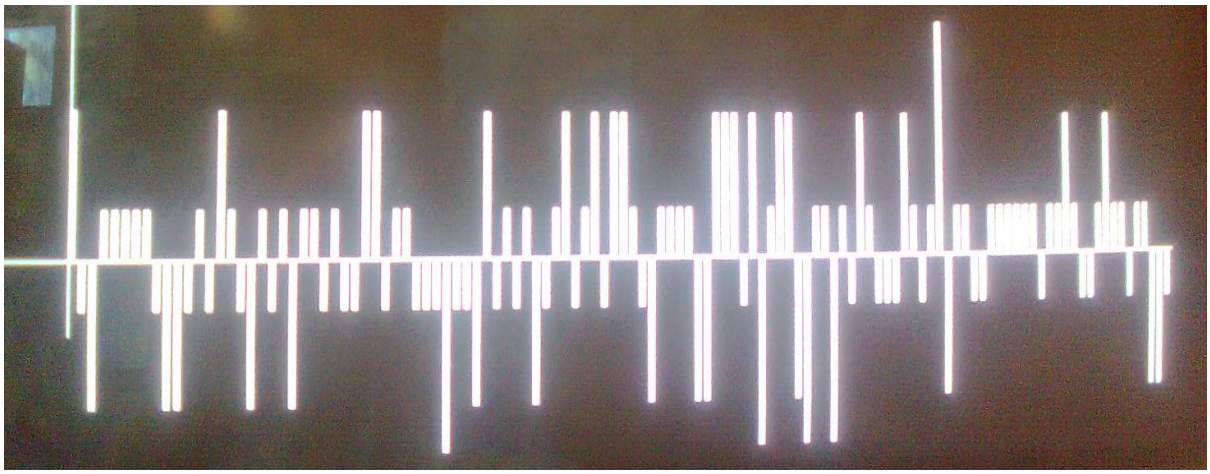
The sender *B* has data *Bd* = -1 (stands for bit 0), and takes takes the product of *Bd* with the chipping sequence *Bk* = 110101 = (1, 1,-1, 1,-1,1) which is: *Bd* . *Bk* = *Bd* * (1,1,-1, 1,-1, 1) Since both transmission happen at the same time, the two signals will superimpose on each other. Thus the receiver of *A* and *B* will hear the following signal: *S* = *Ad* **Ak* + *Bd* * *Bk* = (0, 2,-2, 0, 0, 2).Now here is the key point. With the code *Ak*, a user can decipher from the superimposed signal the data sent by *A*. Basically it XOR its code with the signal, or in other words, performs a dot product of the code *Ak* with the signal *S*: *Ak* . *S* = (-1, 1,-1,-1, 1, 1). (0, 2,-2, 0, 0, 2)= 6 With the code of *Bk*, a user can decipher the data sent by *B*, again, by doing a dot product of *Bk* with *S*: *Bk* . *S* = (1, 1,-1, 1,-1,1) .(0, 2,-2, 0, 0, 2)= -6. A value larger than 0 translates to 1 and a value smaller than 0 translates to 0.

OUTPUT and WAVEFORM

The code given in appendix is run on C++ compiler and following values for the signal of 5 users were obtained, it is also been plotted in C++.

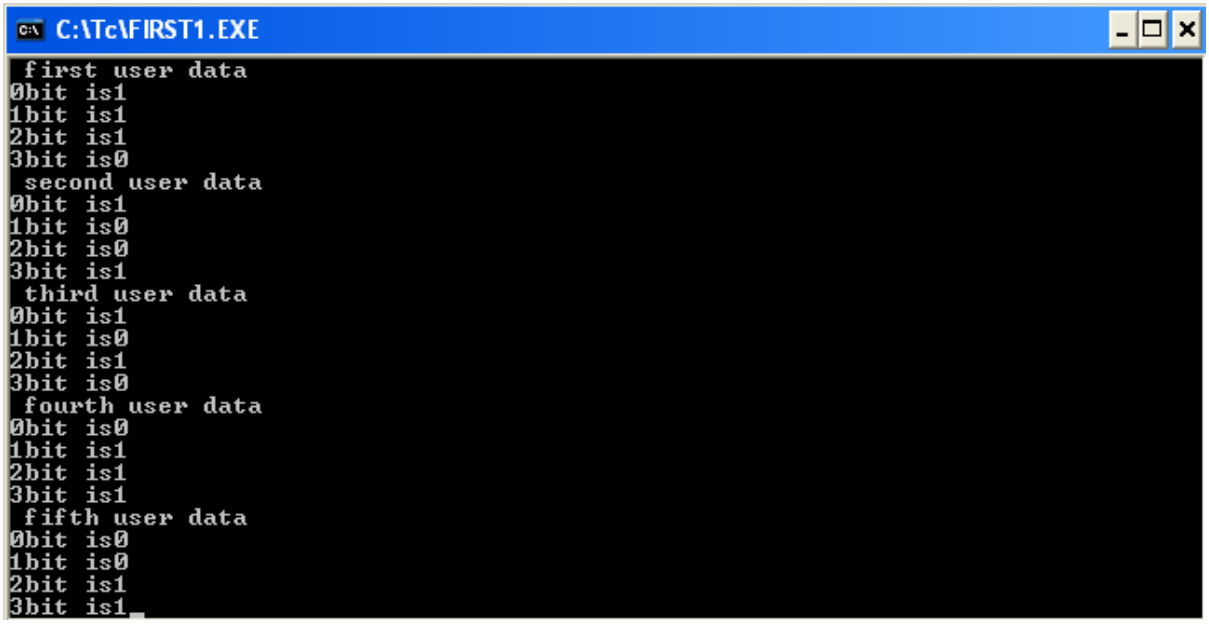
```
        c2 = -1 c2 = -1 c2 = 1  c2 = 1  c2 = 1  c2 = 3  c2 = 1  c2 = -1 c2 = -1
c2 = -3 c2 = 1  c2 = -1 c2 = 1  c2 = 1  c2 = 1  c2 = 3  c2 = -5 c2 = -1 c2 = -3
c2 = -1 c2 = -1 c2 = -3 c2 = 1  c2 = -1 c2 = 3  c2 = 1  c2 = 1  c2 = -1 c2 = 1
c2 = 5  c2 = -3 c2 = 1  c2 = 1  c2 = -1 c2 = -1 c2 = -1 c2 = -3 c2 = -1 c2 = 1
c2 = -3 c2 = 3  c2 = -1 c2 = 1  c2 = 3  c2 = -1 c2 = -5 c2 = 1  c2 = 1  c2 = 1
c2 = -1 c2 = -3 c2 = 5  c2 = -1 c2 = 3  c2 = 1  c2 = 1  c2 = 3  c2 = -1 c2 = -3
c2 = 3  c2 = -1 c2 = -1 c2 = -3 c2 = -3 c2 = 3  c2 = 3  c2 = 3  c2 = 1  c2 = 3
c2 = -3 c2 = 1  c2 = 3  c2 = 3  c2 = -3 c2 = -1 c2 = -1 c2 = -1 c2 = -3 c2 = 1
c2 = -3 c2 = 3  c2 = -3 c2 = 1  c2 = -1 c2 = -1 c2 = 1  c2 = 1  c2 = 3  c2 = 3
c2 = 1  c2 = -1 c2 = -1 c2 = -1 c2 = -1 c2 = -1 c2 = 1  c2 = 1  c2 = 1  c2 = -1
c2 = 1  c2 = -1 c2 = -1 c2 = 1  c2 = 1  c2 = 3  c2 = -3 c2 = 1  c2 = 1  c2 = -1
c2 = 3  c2 = 3  c2 = 1  c2 = 3  c2 = -1 c2 = 1  c2 = -3 c2 = -1 c2 = -1 c2 = -3
c2 = -3 c2 = 3  c2 = -3 c2 = -3 c2 = 5
```

Snapshot 9:- This snapshot shows signal values of 5 signals overlaying in a channel.



Snapshot 10:- This snapshot shows waveform of 5 signal overlaying in a channel.

The DATA was received successfully, when the PRN Sequence was generated locally.



Snapshot 11:- This snapshot shows DATA retrieval for 5 users.

4.6 Role of Processing gain in multi user system and interference

We consider a figure to show the concept of processing gain. The figure used below has been taken from a book entitled "Processing gain in Spread Spectrum Signals" by *John Fakatselis and Harris Semiconductor* [c]. The study of this book gave a better understanding of DSSS transmitter in view of INTERFERE.

The spread signal is with the addition of the wideband modulation utilizing the PRN code. It can be seen in the **figure 34** that the spread signal is wider in frequency BW but with lower power spectral density per Hz. The spread signal is shown to be close to the noise floor. PG for a DS system can be visualized as the margin that exists as the difference between the un-spread and spread waveforms. The PRN code spreads the transmitted signal in bandwidth and makes it less susceptible to narrowband interference.

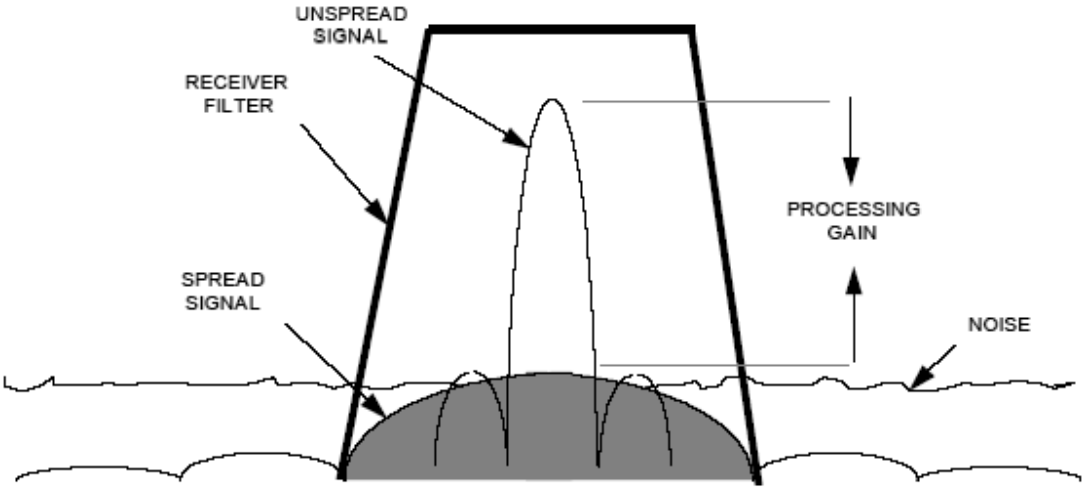
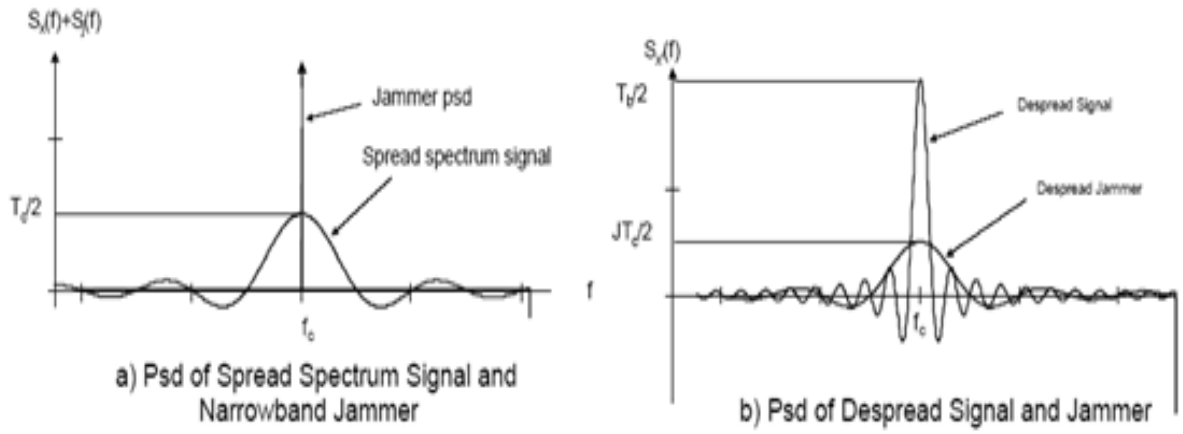


Figure 34 :- Showing processing gain concept at receiver

The receiver of a DSSS system can be viewed as un-spreading the intended signal and at the same time spreading the interfering waveform. This operation is best illustrated in **Figure 35**. **Figure 35** depicts the power spectral density (psd) functions of the signals at the receiver input, the de-spread signal, the bandpass filter power transfer function, and the band pass filter output. **Figure 35** graphically describes the effect of the processing gain on a jammer. The jammer is narrow, and has a highly peaked psd, while the psd of the DSSS is wide and low. The de-spreading operation spreads the jammer power psd and lowers its peak, and the BPF output shows the effect on the signal to jammer ratio.



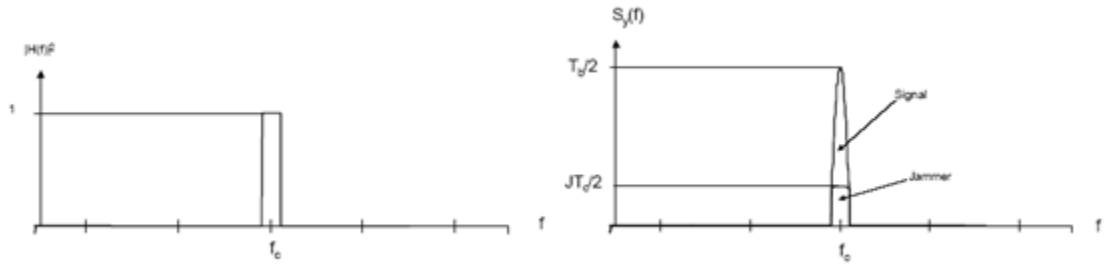
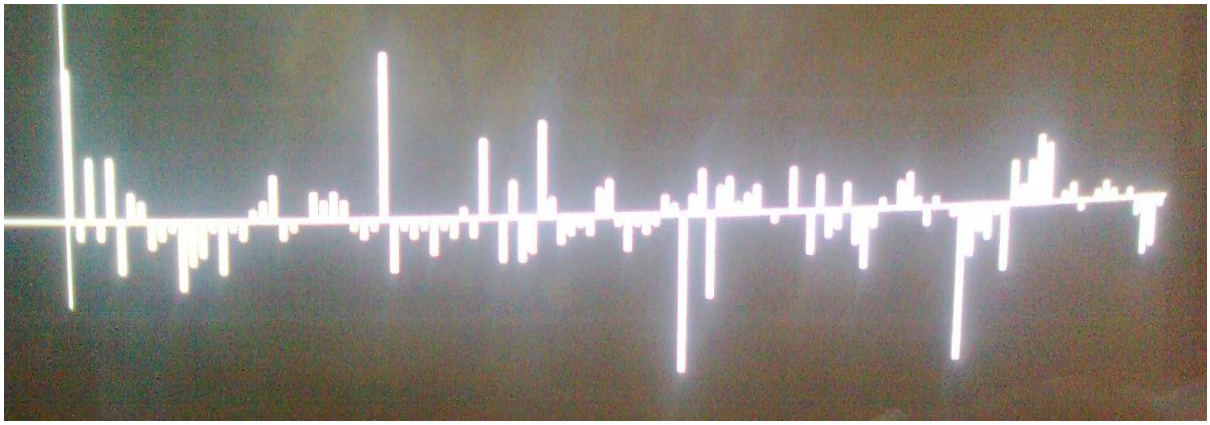


Figure 35:- processing gain effect on narrow band interference

4.7 Data Retrieval for Multiple Users in presence of interference

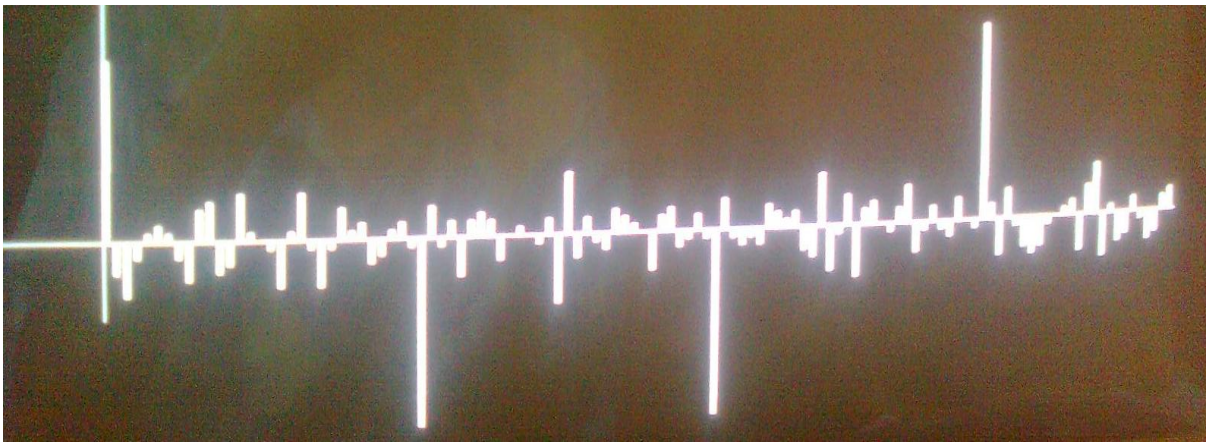
In our project it was little difficult for us to introduce the actual scenario of interference. Because, we were doing the project on Software. So, to introduce the interference in picture, we manually changed some bits in our transmitted spread signal. We use two cases :-

a. In one case we changed 1-bit of the spread signal after every 5-bit. In this way, we not only introduce interference but the effect of NOISE also comes into picture. The retrieved signal after this is:-



Snapshot 12:- This snapshot shows waveform in presence of interference, in which we can see decrease in value of peaks. But then also we can retrieve the data because of sufficient difference between main peaks and noise peaks.

b. In this case we changed 1-bit of the spread signal after every 10-bit. The retrieved signal after this is:-



Snapshot 13:- This snapshot shows waveform in presence of interference, in which we can see decrease in value of peaks. But then also we can retrieve the data because of sufficient difference between main peaks and noise peaks.

CONCLUSION

In our project we studied about the spread spectrum system whose main characteristic is the presence of a code or key, which must be known in advance by the transmitter and receiver(s).The spread spectrum system is mainly classified into DSSS and FHSS. We worked on DSSS. In Spread Spectrum our main concern was to know the type of code we are going to use. In modern communications the codes are digital sequences that must be as long and as random as possible to appear as "noise-like" as possible. But in any case, the codes must remain reproducible, or the receiver cannot extract the message that has been sent. Thus, the sequence is "nearly random." Such a code is called a pseudo-random number (PRN) or sequence. The method most frequently used to generate pseudo-random codes is based on a feedback shift register.

The construction or selection of proper sequences, or sets of sequences, are not trivial. To guarantee efficient spread-spectrum communications, the PRN sequences must respect certain rules, such as length, autocorrelation, cross-correlation and bits balancing. We worked on *PRN sequences* and *Gold Codes*. Keep in mind that a more complex sequence set provides a more robust spread-spectrum link. But there is a cost to this: more complex electronics both in speed and behaviour, mainly for the spread-spectrum de-spreading operations.

During our project we worked on PRN sequence generation and studied their properties such as length, autocorrelation, cross-correlation and bits balancing. We studied PRN sequence of different lengths and found some sequences are more suitable for spreading than others. Then we also studied Gold sequences and their autocorrelation, cross-correlation, and bits balancing properties.

On comparing PRN-sequences and gold sequences on basis of their correlation properties we found that both PRN and gold sequences are suitable for spreading, also the cross-correlation decreases with increasing length thereby giving good results in presence of interference and multi user systems, as the processing gain is high. Gold sequences are better if we have to deal with large number of users, as large number of gold codes are available for a particular length as compared to PRN-sequences.

APPENDICES

A. CODE for PN-Sequence Generation

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
    int x5,x4,x3,x2,x1,a,j=0,x0,b[32];
    cout<<"\n Enter the seed value in 1 bit binary:- \n";
    cout<<"\tx5 = ";
    cin>>x5;
    cout<<"\tx4 = ";
    cin>>x4;
    cout<<"\tx3 = ";
    cin>>x3;
    cout<<"\tx2 = ";
    cin>>x2;
    cout<<"\tx1 = ";
    cin>>x1;
    cout<<"\n the random number sequence is as follows :- ";
    cout<<"\n"<<"clk"<<"\tb4"<<"\tb3"<<"\tb2"<<"\tb1"<<"\tb0"<<"\tPNSEQ.;
    cout<<"\n"<<j<<"\t"<<x5<<"\t"<<x4<<"\t"<<x3<<"\t"<<x2<<"\t"<<x1<<"\t<<x1 ;
    b[0]=x1;
    \\ this block implements the basic operation of LFSR for generating PN-Seq. of length 31
     for(j=1;j<32;j++)
    {
            x0 = x1;
            x1 = x2;
            x2 =  x3;
            x3 =  x4;
```

```
        x4 =  x5;
        a  =  x0 + x3;
        b[j]=x1;
        if(a == 0 || a == 2)
                x5 = 0;
        else if(a == 1)
                x5 = 1;
        cout<<"\n"<<j<<"\t"<<x5<<"\t"<<x4<<"\t"<<x3<<"\t"<<x2<<"\t"<<x1<<"\t"<<x1 ;
}
getch();
```

\\this block represents the PN-Seq. in graphical form using GRAPHIC USER INTERFACE

```
        int i,k=50;
                int gd,gm;
                gd=DETECT;
                initgraph(&gd,&gm," ");
                line(0,400,900,400);
                line(50,430,50,50);
                for(i=0;i<32;i++)
                {
                        if(b[i]==1)
                                bar(k,400,k+10,380);
                        else
                                bar(k,400,k+10,400);
                k = k+12;
                }
                getch();
                closegraph();
        }
```

B. CODE for PN-Sequence Autocorrelation

I. PERIODIC AUTOCORRELATION

```
 int p=0,n=0,q=0,m=0,v[63],h=50;
```

```
cout<<"\n";
for(n = 0; n<34; n++)
{
        for(p = 0; p<62 ;p++)
        {
                if(b[p] == 0)
                        d[p] = -1;
                else
                        d[p] = 1 ;
                        e[p] = d[p];
                        c[p] = d[p]*e[p+n];
        }
        m = c[0];
        for(p = 1; p<31 ;p++)
        {
                m = m + c[p];
        }
        cout<<"\t"<<m;
        v[n] = m;
        m = 0;
}
getch();
gd = DETECT;
initgraph(&gd , &gm, " ");
line(0,400,900,400);
line(50,430,50,50);
for (i = 0; i<33; i++)
{
        float  l = (v[i]);
        float z = (400-l);
        bar (h,400,h+7,z);
        h = h+9;
}
 getch();
```

```
closegraph();
```

II. APERIODIC AUTOCORRELATION

```
int p = 0, n = 0, q = 0, m = 0;
cout<<"\n\n";
for(n = 0; n<511; n++)
{
        for(p = 0; p<1022; p++)
        {
                if(b[p] == 0)
                        d[p] = -1;
                else
                        d[p] = 1 ;
                        e[p] = d[p];
                        c[p] = d[p] * e[p + n];
        }
        m  = c[0];
        for(p =1 ; p<511; p++)
        {
                m  = m + c[p];
        }
        cout<<" "<<m;
        v[n] = m;
}
getch();
int i; float k = 50;
int gd,gm;
gd = DETECT;
initgraph(&gd, &gm, " ");
line(0,400,900,400);
line(50,430,50,50);
for(i = 0; i<511; i++)
{
        float  l = (1 * v[i]);
```

```
            float z = (400 - l);
            bar(k, 400, k+1, z);
            k  =  k + 1.5;
        }
        getch();
        closegraph();
```

C. CODE for GOLD CODES Generation

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
void main()
{
        int x5,x4,x3,x2,x1,l,j = 0,x0,c1,c4,b[124],b1[124],r1[124],q1[124],i = 0;
        for(int a = 0; a<124; a++)
        {
            b[a]=0;
            b1[a]=0;
            q1[a]=0;
        }
        cout<<"\n Enter the seed value in 1 bit binary:- \n";
        cout<<"\tx5 = ";
        cin>>x5;
        cout<<"\tx4 = ";
        cin>>x4;
        cout<<"\tx3 = ";
        cin>>x3;
        cout<<"\tx2 = ";
        cin>>x2;
        cout<<"\tx1 = ";
        cin>>x1;
        cout<<"\n the random number sequence is as follows :- ";
        cout<<"\n"<<"clk"<<"\tb4"<<"\tb3"<<"\tb2"<<"\tb1"<<"\tb0"<<"\tPNSEQ.";
        cout<<j<<"\t"<<x5<<"\t"<<x4<<"\t"<<x3<<"\t"<<x2<<"\t"<<x1<<"\t"<<x1 ;
```

```
b[0] = x1;
for(j = 1; j<124; j++)
{
        x0  =  x1;
        x1  =  x2;
        x2  =  x3;
        x3  =  x4;
        x4  =  x5;
        l  =  x0 + x2;
b[j] = x1;
if(l == 0 || l == 2)
        x5 = 0;
else if(l == 1)
        x5  =  1;
cout<<j<<"\t"<<x5<<"\t"<<x4<<"\t"<<x3<<"\t"<<x2<<"\t"<<x1<<"\t"<<x1 ;
}
getch();
cout<<"\n Enter the seed value in 1 bit binary:- \n";
cout<<"\tx5 = ";
cin>>x5;
cout<<"\tx4 = ";
cin>>x4;
cout<<"\tx3 = ";
cin>>x3;
cout<<"\tx2 = ";
cin>>x2;
cout<<"\tx1 = ";
cin>>x1;
cout<<"\n the random number sequence is as follows :- ";
cout<<"\n"<<"clk"<<"\tb4"<<"\tb3"<<"\tb2"<<"\tb1"<<"\tb0"<<"\tPNSEQ.";
cout<<j<<"\t"<<x5<<"\t"<<x4<<"\t"<<x3<<"\t"<<x2<<"\t"<<x1<<"\t"<<x1 ;
b1[0] = x1;
for(j = 1; j<124; j++)
{
```

```
            x0 = x1;
            x1 = x2;
            x2 =  x3;
            x3 =  x4;
            x4 =  x5;
            l  =  x0 + x3;
        b1[j] = x1;
        if(l == 0 || l == 2)
            x5 = 0;
        else if(l == 1)
            x5 = 1;
        cout<<j<<"\t"<<x5<<"\t"<<x4<<"\t"<<x3<<"\t"<<x2<<"\t"<<x1<<"\t"<<x1 ;
        }
        getch();
        for(i = 0; i<4; i++)
        {
            for(j = i*31; j<(i*31) + 31; j++)
            {
                c1 = b[i] + b1[j];
                if(c1 == 0 || c1 == 2)
                    r1[j] = 0;
                else if(c1 == 1)
                    r1[j]=1;
                cout<<"\t r="<<r1[j];
            }
        }
        getch();
```

OUTPUT

Using the above code we generated different gold codes of length 31.

Code **A**: 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0

Code **B**: 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 0 0

Code **C**: 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 0 0 1 0

Code **D**: 0 0 0 0 0 0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 0 1 1 1 0 1 0 1 0

D. CODE for Autocorrelation GOLD CODES

```
for(int h = 0; h<124; h = h + 31)
{
        int x = 31, u = 0, y = h, f = 0,g = h;
       while(x >= 0)
       {
              for(j = h,f = y; j<(h + x), f<g+31; j++,f++)
              {
                     c4 = r1[j] + r1[f];
                     if(c4 == 0 | | c4 == 2)
                            q1[y] = q1[y] + 1;
                     else if(c4 == 1)
                            q1[y] = q1[y]-1;
              }
              cout<<"\t q= "<<q1[y];
              x--,y++;
       }
    }
    getch();
    int z,q, k  = 50;
    int gd,gm;
    gd = DETECT;
    initgraph(&gd,&gm," ");
    line(0,400,900,400);
    line(50,430,50,50);
    for(z = 0; z<124; z++)
   {
              float  l = (3*q1[z]);
              float b1 = (400-l);
              bar(k,400,k+2,b1);
              k = k + 5;
}
getch();
```

```
closegraph();
```

E. CODE for Cross-Correlation GOLD CODES

```
for(int h = 0; h<124; h = h + 31)
{
        int x = 31, u = 0, y = h, f = 0, g = h;
        while(x >= 0)
        {
                for(j = h, f = y; j<(h + x), f<g+31; j++, f++)
                {
                        c4 = r1[j] + r2[f];
                        if(c4 == 0 | | c4 == 2)
                                q1[y] = q1[y] + 1;
                        else if(c4 == 1)
                                q1[y] = q1[y] - 1;
                }
                cout<<"\t q= "<<q1[y];
                x--,y++;
        }
 }
getch();
int z,q,k = 50;
int gd,gm;
        gd  =  DETECT;
        initgraph(&gd, &gm, " ");
        line(0,400,900,400);
        line(50,430,50,50);
        for(z = 0; z<124; z++)
        {
                float  l = (3*q1[z]);
                float b1 = (400 - l);
                bar(k,400, k + 2, b1);
                k = k + 5;
```

```
    }
  getch();
  closegraph();
```

F. CODE for Spread Signal Generation

```
cout<<"\n Enter the data\n";
for(i = 0; i<4; i++)
{
        cin>>d[i];
}
for(i = 0; i<4; i++)
{
    for(j = 0; j<31; j++)
    {
        c = d[i] + b[j] ;
        if(c == 0 | | c == 2)
            r[q] = 0;
        else if(c == 1)
            r[q] = 1;
        cout<<"\t r=  "<<r[q];
        q++;
    }
}
```

G. CODE for Match Filter Operation

```
    for(int h = 0; h<124; h = h + 31)
    {
        int x =31,u = 0,y = h,k = 0,l = h,t = 0;
        for(t=0;t<150;t++)
        {
            p[t] = 0;
        }
        t = 0;
```

```
while(x>=0)
{
for(j = h, k = y; j<(h+x),k<l+31;j++,k++)
{
        u=r[k]+b[j];
        if(u == 0||u == 2)
                p[y] = p[y]-1;
        else if(u == 1)
                p[y] = p[y] + 1;
        cout<<"\t p=  "<<p[y];
        x--,y++;
}
}
```

H. CODE for Data Retrieval for Multiple User

```
cout<<"\n"<<"\n";
int c5[128];
for(j = 0; j < = 123; j++)
{
        c5[j] = c[j] + c1[j] + c2[j] + c3[j] + c4[j];
        cout<<"\tc2 = "<<c5[j];
}
getch();
cout<<"\n first user data" ;
for(i=0 ; i<4 ; i++)
{
        t  = 0;
        for(q = i*31, j = 0; q < (i+1)*31, j < 31; q++ , j++)
        {
                int g;
                g = c5[q] * o[j];
                t = t + g;
        }
        if(t>0)
```

```
                v  = 1;
        else if(t<0)
                v  = 0;
        cout<<"\n"<<i<<"bit is"<<v;
}
cout<<"\n second user data" ;
for( i=0; i<4; i++)
{
        t = 0;
        for( q = i*31,j = 0; q < (i+1)*31, j < 31; q++, j++)
        {
                int g;
                g = c5[q] * o1[j];
                t = t + g;
        }
        if( t > 0)
                v  = 1;
        else if( t < 0 )
                v  = 0;
        cout<<"\n"<<i<<"bit is"<<v;
 }
 cout<<"\n third user data" ;
 for( i = 0; i < 4; i++)
{
        t  = 0;
        for( q = i*31, j=0; q < (i+1)*31, j < 31; q++, j++)
        {
                int g;
                g = c5[q] * o2[j];
                t = t + g;
        }
        if(t>0)
                v  = 1;
        else if(t<0)
```

```
                v  = 0;
        cout<<"\n"<<i<<"bit is"<<v;
}
cout<<"\n fourth user data" ;
for( i = 0 ; i < 4; i++)
{
        t = 0;
        for( q=i*31, j=0; q<(i+1)*31, j < 31; q++, j++)
        {
                int g;
                g = c5[q] * o3[j];
                t = t + g;
        }
        if( t > 0)
                v = 1;
        else if( t < 0)
                v  = 0;
        cout<<"\n"<<i<<"bit is"<<v;
}
cout<<"\n fifth user data" ;
for( i = 0; i < 4; i++)
{
        t  = 0;
        for( q = i*31, j = 0; q < (i+1)*31, j < 31; q++, j++)
        {
                int g;
                g = c5[q] * o4[j];
                t = t + g;
        }
        if(t > 0)
                v  = 1;
        else if( t < 0 )
                v = 0;
        cout<<"\n"<<i<<"bit is"<<v;
```

```
}
        getch();
}
        int p,k = 50;
        int gd,gm;
        gd = DETECT;
        initgraph(&gd, &gm, " f");
        line(0,400,900,400);
        line(50,430,50,50);
        for( p = 0;  p<123; p++)
        {
                float l = (20*c2[p]);
                float z = (400-l);
                bar(k,400,k+2,z);
                 k = k+5;
        }
        closegraph();
        getch();
}
```

I. Collection of Definitions, Useful in understanding of Project

Code Division Multiple Access (**CDMA**) is a channel access method used by various radio communication technologies. CDMA employs spread-spectrum technology and a special coding scheme (where each transmitter is assigned a code) to allow multiple users to be multiplexed over the same physical channel.

Direct-Sequence Spread Spectrum (**DSSS**) is a modulation technique. DSSS uses a signal structure in which the sequence of chips produced by the transmitter is known *a priori* by the receiver. The receiver can then use the same *PRN sequence* to counteract the effect of the PRN sequence on the received signal in order to reconstruct the information signal.

Frequency-Hopped Spread Spectrum (**FHSS**) is a method of transmitting radio signals by rapidly switching a carrier among many frequency channels, using a pseudorandom sequence known to both transmitter and receiver.

A **Pseudo-Random Sequence (PRNS)** is one that appears to be perfectly random for K output symbols but then repeats, i.e. it is periodic with a cycle time of K symbols. If K can be made large enough, this is not a great limitation; any interval up to K symbols will appear to be perfectly random.

Binary phase shift keying (BPSK) (also sometimes called PRK, Phase Reversal Keying, or 2PSK) is the simplest form of phase shift keying (PSK). It uses two phases which are separated by 180° and so can also be termed 2-PSK. It does not particularly matter exactly where the constellation points are positioned, and in this figure they are shown on the real axis, at 0° and 180°.

The **Chip Rate** of a code is the number of pulses per second (chips per second) at which the code is transmitted (or received).

The ratio of chip rate to symbol rate is known as the **Spreading Factor (SF)** or **Processing Gain**. The chip rate is larger than the symbol rate, meaning that one symbol is represented by multiple chips.

Fading is deviation of the attenuation that a carrier-modulated telecommunication signal experiences over certain propagation media.

A **linear feedback shift register** (LFSR) is a shift register whose input bit is a linear function of its previous state.

A **maximum length sequence** (**MLS**) is a type of pseudorandom binary sequence. They are bit sequences generated using maximal linear feedback shift registers and are so called because they are periodic and reproduce every binary sequence that can be reproduced by the shift registers (i.e., for length-*m* registers they produce a sequence of length $2^m - 1$). A MLS is also sometimes called a **n-sequence** or a **m-sequence**.

Autocorrelation is the cross-correlation of a signal with itself. Informally, it is the similarity between observations as a function of the time separation between them.

Cross-Correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. This is also known as a *sliding dot product* or *inner-product*. The cross-correlation is similar in nature to the convolution of two functions. Whereas convolution involves reversing a signal, then shifting it and multiplying by another signal, correlation only involves shifting it and multiplying (no reversing).

A **Gold code**, also known as **Gold sequence**, is a type of binary sequence. Gold codes have bounded small cross-correlations within a set, which is useful when multiple devices are broadcasting in the same range. A set of Gold code sequences consists of $2^n - 1$ sequences each one with a period of $2^n - 1$.

The **XOR gate** (sometimes **EOR gate** or **EXOR gate**) is a digital logic gate that implements an exclusive disjunction; that is, it behaves according to the truth table shown on the right. A true output (1) results if one, and only one, of the inputs to the gate is true (1). If both inputs are false (0) and both are true (1), a false output (0) results. A way to remember XOR is "one or the other but not both". It represents the inequality function, i.e., the output is HIGH (1) if the inputs are not alike otherwise the output is LOW (0).

Frequency Multiplier is an electronic circuit that generates an output signal whose output frequency is a harmonic of its input frequency. Frequency multipliers consist of a nonlinear circuit that distorts the input signal and consequently generates harmonics of the input signal.

A **band-pass filter** is a device that passes frequencies within a certain range and rejects (attenuates) frequencies outside that range. These filters can also be created by combining a low-pass filter with a high-pass filter.

A **Low-pass filter** is a filter that passes low-frequency signals but attenuates (reduces the amplitude of) signals with frequencies higher than the cutoff frequency. The actual amount of attenuation for each frequency varies from filter to filter.

Synchronization is timekeeping which requires the coordination of events to operate a system in unison.

A **Matched filter** is obtained by correlating a known signal, or template, with an unknown signal to detect the presence of the template in the unknown signal. This is equivalent to convolving the unknown signal with a conjugated time-reversed version of the template. The

matched filter is the optimal linear filter for maximizing the signal to noise ratio (SNR) in the presence of additive stochastic noise.

Jamming is the transmission of those signals that disrupt communications by decreasing the signal to noise ratio. Unintentional jamming occurs when an operator transmits on a busy frequency without first checking whether it is in use, or without being able to hear stations using the frequency.

References and Bibliography

a) Wireless communication and Networks
   - *William Stallings*.

b) Error Correcting Codes
   - *W.W. Peterson*

c) Processing Gain in Spread Spectrum Signals
   -*John Fakatselis and Harris Semiconductor*

d) Maxim Integrated Products
   -*Application note 1890*

e) Communication Systems
   -*A. Bruce Carlson, Paul B. Crilly and Janet C. Rutledge*

f) Characteristic Linear Sequences and their Co-set functions.
   -*R. Gold* (accepted for publication J. Soc Ind. Appl. Math, May 1965.)

g) CDMA-Principles of Spread Spectrum Communication
   - *Andrew J. Veterbi*.

h) Wireless Communication
   - *Andrea Goldsmith*

i) Principles of spread spectrum
   – *Don j. Torrieri*

j) CDMA with direct sequence spread spectrum
   – *Jie Gao February* 13, 2007

k) Spread spectrum and CDMA
   - *Tan & Fong*

g) Spread spectrum techniques and technology

– *Mark a. sturza, 3C Systems Company*

h) Spread spectrum systems

– *Robert C. Dixon*

i) *Yu. V. Stasev* , *A. A. Kuznetsov*, and *A. M. Nosik* ,"Formation of Pseudorandom sequences with improved autocorrelation properties" UDC 621.396.

j) Spread spectrum communications Handbook

– *Marvin K.simon, Jim K. omura, Robert A. scholtz , Barry K. levitt*, Mc Graw Hill.

k) Introduction to Spread Spectrum Communications

– *Robert L. Peterson, Rodger E. ziemer, David E. Borth*, Prentice Hall

l) Coherent Spread spectrum systems

– *Jack K. Holmes , Wiley –Interscience*

m) Introduction to Spread-Spectrum Communications

- By *Roger L. Peterson* (Motorola), *Rodger E. Ziemer* (University of Co. at Colorado Springs), and *David E. Borth* (Motorola),Prentice Hall, 1995(Navtech order #2430).

n) Frame , bit and chip error rate evaluation for a DSSS communication system

– *F.R. Castillo –Soria , D. Pacheco-Bautista y M. Sanchez – Meraz* , Universidad Del Istmo, Campus Ixtepec , Oaxaca, Mexico y Departemento de Telecomunicaciones SEPI-ESMI-IPN , Unidad Profesional "Adolfo Lopez Mateos",Mexico DF , Ingenieria Investigacion Y Techno logia IX 271-277 , 2008.