ADAPTIVE ALGORITHM FOR OBTAINING IN-PHASE (I) AND QUADRATURE-PHASE (Q) PSEUDO-NOISE (PN) SEQUENCES IN CDMA SYSTEMS

By

KAFI I. HASSAN

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York
2005

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Date	Chair of Examining Committee
Date	Executive Officer

Professor Michael Conner

Professor Tarek Saadawi

Professor Norman Scheinberg

Professor Mohamed M. Zahran

Dr. Nidal Khrais

Supervision Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

ADAPTIVE ALGORITHM FOR OBTAINING IN-PHASE (I) AND QUADRATURE-PHASE (Q) PN SEQUENCES IN CDMA SYSTEMS

by

Kafi I. Hassan

Advisor: Professor Michael Conner

In the CDMA wireless systems, the forward pilot channel is spread in quadrature spreading. The quadrature spreading is a spreading sequence of length 2^{15} pseudo-noise (PN) chips. For spreading rate (SR) 1, each PN chip is about 0.813μs. This sequence is called the pilot PN sequence and consist of a pair of PN sequences called the In-phase (**I**) and quadrature-phase (**Q**) sequences generated by a Linear Feedback Shift Register (LFSR). This pair of PN sequences is used to spread the forward and the reverse CDMA Channel. During the forward Pilot and Sync channel processing, the mobile station acquires and synchronizes to the CDMA system while it is in the initialization state. In the current CDMA systems, to obtain **I** and **Q** Pilot PN sequences for time alignment, a zero is inserted in the LFSR sequences after 14 consecutive "0" outputs. When the mobile station power is turned on, the MS synchronizes with the base station timing using Global Positioning System (GPS) timing reference on the Sync channel and the pilot channel. In this thesis, we propose an algorithm that reduces the computation time to find the **I** and **Q** Pilot PN sequences. We begin by developing evolutionary computation models to find the

initial values (IV) and polynomial feedback coefficients (PFC) for the maximal-length LFSR that is used to generate the CDMA PN sequences. Next, we propose an adaptive algorithm for computing the **I** and **Q** pilot PN sequences in the CDMA systems that can result in a shorter synchronization time between the base station and mobile station. To study the performance of the proposed algorithm, many simulation experiments are carried out under noisy and non-noisy channel conditions. Our simulation test results show that the proposed algorithms cuts the synchronization time by a minimum of 32753 PN chips without degrading the channel performance, and with less computational complexity when compared to the currently used methods.

Acknowledgments

I am very grateful to my advisor, Dr. Michael Conner, for his support, supervision, and assistance. It has been a true privilege for me to work with, and learn from Dr. Conner. Dr. Conner's support and his patience to listen my ideas allowed me to focus and complete this work.

I am also grateful to Professors Saadawi, Scheinberg, and Zahran for serving in my supervisory committee. Their valuable questions, suggestions, and comments strengthened many parts of this research.

In addition, I would like to thank to Dr. Khrais from Lucent Technologies, for serving in my supervisory committee. Dr. Khrais's valuable knowledge in CDMA systems has been very inspiring and helpful.

My heartfelt gratitude goes to my wife Amina, and my two children: Abdifatah and Aisha, who provided me with endless support throughout my PhD education. I am deeply grateful for their encouragement. I am very fortunate to had the continuous support from my parents and my brothers: Mohamed, Abdifatah, and Anwar, in all levels of my education and I will always be grateful to them.

My thanks to my current and previous technical managers and directors at Lucent Technologies for supporting me during my PhD education.

This research was supported in part by Lucent Technologies Bell Labs.

CONTENTS

List of Tables

List of Figures

Glossary

3G	Third Generation Wireless System
3GPP2	Third Generation Partners Project 2
AMPS	Advanced Mobile Phone system
BER	Bit Error Rate
BS	Base Station
c	PN Chip
CDMA	Code Division Multiple Access
CIR	Carrier-to-Interfere Ratio
Clks	Clock
Eb	Bit Energy
$\frac{E_b}{N_o}$	Bit Energy to Noise Spectrum
EIA	Electronic Industry association
FL	Forward link
Flops	Floating operating
GA	Genetic Algorithm
I	In-phase
IDS	Intrusion Detection System
IS	Interim Standard
IS-95A	CDMA Interim Standard 95A

IS-2000	CDMA Interim Standard 2000
IV	Initial Values of Maximal Length LFSR
IVAA	Initial Values – Adaptive Algorithm
IVPFCAA	Initial Values and Polynomial Feedback Coefficient– Adaptive Algorithm
LFSR	Linear Feedback Shift Register
m	Number of shift Flip-flop
MHz	Mega Hertz
ML	Maximal Length
MS	Mobile Station
No	Noise Spectrum
Pc	Probability of Crossover
PCS	Personal Communication Systems
PFC	Polynomial Feedback Coefficients
PFCAA	Polynomial Feedback Coefficients- Adaptive Algorithm
PG	Processing Gain
Pm	Probability of mutation
PN	Pseudo-Noise
PR	Pseudo-Random
PRNG	Pseudo-Random Number Generator
Q	Quadrature-phase
RL	Reverse Link
RX	Receiver
SNR	Signal to Noise Ratio

SOM	Start of Message
SR	Shift Register
SR 1	Spreading Rate 1
TIA	Telecommunications Industry Association
TX	Transmitter
XOR	exclusive-OR

Chapter 1

1 Introduction

The linear feedback shift register (LFSR) has been extensively used in many communication and computer engineering systems as a pseudo-random (pseudo-noise) generator [31]. Primarily, LFSR is used as a pseudo-noise (PN) generator in the wireless Code Division Multiple Access (CDMA) communication systems, and as crypto-analysis in security systems [32]. The LFSR is used more often comparing to the other pseudo-random generators, this is because the LFSR is easier to implement in hardware [45]. The maximal length LFSR with ***m*** cascaded shift registers has a sequence period N equal to $2^m - 1$ [31]. Therefore, the sequence of the LFSR repeats every N period. The prediction of the pattern of LFSR sequences has been active research area. Our research focuses on predicting efficiently the sequences of the maximal-length (ML) LFSR using its output data.

First, we consider a ML LFSR with known polynomial feedback coefficients and unknown initial value contents on the shift registers. We develop an adaptive algorithm that can compute the initial values from the output data of the LFSR using evolutionary computation. We apply the algorithm to the third generation wireless code division multiple access (CDMA) systems [27][80]. CDMA systems based on the interim standards IS-95 [78] and IS-2000 [1][2][3] use a 15-stage LFSR shift with standard defined

polynomial feedback coefficients. Our study attempts to maximize the computation accuracy of the initial value of the shift registers in the shortest clock sequence.

Second, we consider a ML LFSR with known initial value on the shift registers, and unknown polynomial feedback coefficients. We develop an adaptive algorithm for computing the polynomial feedback coefficients using the output data of the LFSR.

Third, we extend the problem to the situations where both the initial values and PFC are unknown. In this case, we develop an adaptive algorithm capable for finding both initial values and polynomial feedback coefficients. This is applicable in the scenarios where a protected communication channel is being intentionally interfered by interference sources from a LFSR pseudo-noise generator.

Simulation test results show that the evolutionary computations such as the Genetic Algorithm (GA) [30][22] have great potential in the area of LFSR sequence computation. Test results also show that our adaptive algorithms can accurately compute the LFSR initial values and polynomial feedback coefficients in a relatively shorter LFSR sequence period.

1.1 Linear Feedback Shift Registers Sequence Computation

Means of generating maximal-length (ML) LFSR predicting schemes has been the focus of our research. Since the sequence of the ML LFSR repeats in certain sequences length, our objective is developing a prediction model that can find the initial values (IV) and the polynomial feedback coefficients (PFC) in short sequences time. The prediction accuracy

within short time is very important since many of the communication applications require it. We decided to break our investigation into three main categories:

1. **Computing the ML LFSR Initial values**: In this part, we developed an adaptive algorithm that finds the initial values of the ML-LFSR (IVAA). We assumed that the polynomial feedback coefficients are known. This scenario is important and is applicable to the wireless CDMA communication systems. Particularly in the base station and mobile station synchronization application in the CDMA systems.

2. **Computing the PFC of the ML LFSR:** In this part, we developed an adaptive algorithm that finds the PFC of the ML-LFSR (PFCAA). We assumed that the initial values of the ML-LFSR are known.

3. **Computing both the IV and PFC:** Here we developed an adaptive algorithm that finds both the IV and PFC (IVPFCAA) simultaneously. In this case, both the initial values and the PFC are unknown.

Our computation models use the output bits generated by the ML-LFSR to compute the IV and the PFC. Genetic algorithm has been shown to be very effective in areas of search and optimization. Genetic algorithm [30][47] has been widely used in many search and optimization applications [44]. In the next sections of this chapter, we will discuss how we implemented our prediction model using GA in each of the three sections that we mentioned. This thesis focuses on developing adaptive algorithms for computing the PN sequences generated by a LFSR in a shorter time with higher accuracy. Then the algorithm

are implemented and tested in CDMA wireless systems to find the start of message (SOM) bit in the Pilot PN sequence to reduce the base station and mobile station synchronization time.

1.2 Thesis Outline

This thesis is divided into seven chapters, including this one. The contents of the remaining chapters are summarized below.

Chapter 2: Background Information

Relevant background information about ML Linear Feedback Shift Registers (LFSR), Genetic Algorithms (GA), and CDMA synchronization are introduced in this chapter. The basic implementation of the ML-LFSR is described. The overview of the foundation of GA as an optimization and search tool are also discussed. Practical implementation and uses of the ML-LFSR in the CDMA wireless systems are also explained. The background information in this chapter about the LFSR, GA, and CDMA are high-level overview composed from the current literature.

Chapter 3: Computing IV of ML LFSR

This chapter proposes an adaptive algorithm that allows predicting the initial value (IV) of ML-LFSR when the polynomial feedback coefficients are known. The steps and input parameters of the adaptive algorithm are described in detail. Performance simulation test results using various polynomial feedback coefficients are presented.

Chapter 4: Adaptive Algorithm for CDMA Synchronization

This chapter examines the practical implementation of the adaptive algorithm developed in chapter 3. The initial value computation algorithm is applied in practical CDMA wireless applications. The algorithm is tested for computing the SOM bit in the I and Q Pilot PN sequences used in the quadrature spreading of CDMA wireless systems. The adaptive algorithm implementation and its performance results are presented in this chapter.

Chapter 5: Computing PFC of ML LFSR

This chapter explains how the adaptive algorithm can be used to find the PFC of a ML-LFSR in situations that the initial values of the ML-LFSR are known. Performance test results for the adaptive algorithm under different initial values are presented and discussed.

Chapter 6: Computing both IV and PFC of ML LFSR

This chapter extends the application of the adaptive algorithm in scenarios where both the initial values and polynomial feedback coefficients of the ML-LFSR are unknown. In these cases, the adaptive algorithm attempts to predict both initial and polynomial feedback of a ML-LFSR simultaneously. This scenario is applicable to the network security threat identification applications. A wireless ad-hoc network application that uses this algorithm is presented in this chapter

Chapter 7: Conclusion and Future Work

The conclusions and extensions of this thesis are provided in this chapter.

Chapter 2

2 Background Information

The theory of LFSR has found major applications in a wide variety of computer and communication systems [31]. LFSR is used as a spreading pseudo-noise (PN) sequence in the code division multiple access (CDMA) wireless communications [79], and as a key-stream generator in cryptosystems [74]. LFSR has been also extensively implemented and proposed for using as a test pattern generator in circuits under test [4][9][24]. The remaining sections of this chapter provide an overview of the LFSR and discuss how they are used in the wireless CDMA systems based on in-term standard IS-2000 [44] and IS-95 [76]. This chapter also introduces the basic concept of the Genetic Algorithms [47][30] under the evolutionary computation. In this thesis, the GA is used as an optimization and search tool.

2.1 Linear Feedback Shift Register Sequences

ML LFSR creates notably good pseudo-noise sequences [46]. A pseudo-noise sequence is a binary sequence that has an autocorrelation that appears over a time as the autocorrelation of a random binary sequence. Although it is deterministic, a pseudo-noise sequence has many characteristics that are similar to those of true random binary sequences, such as having nearly equal number of 0s and 1s, very low cross-correlation between any two sequences, very low correlation between shifted versions of the sequence [33]. The

pseudo-noise sequence is usually generated using sequential logic circuit that contains a feedback shift register. When the feedback logic consists of exclusive-OR gates, the shift register is called a LFSR. Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. The LFSR can be built by performing exclusive-OR on the outputs of two or more of the flip-flops together and feeding those outputs back into the input of one of the flip-flops [33][45][46]. The typical realization of LFSR is shown in Figure 1.

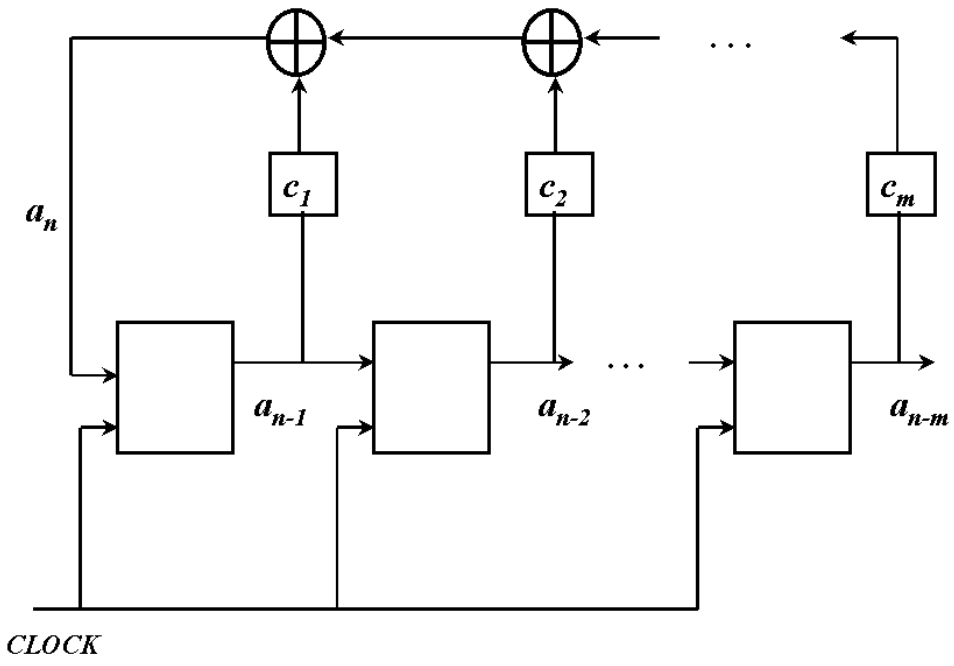

Figure 1. Diagram of LFSR Realization

When the outputs of the flip-flops are loaded with an initial value (anything except all 0s) and when the LFSR is clocked, it will generate a pseudo-noise sequence of 1s and 0s. The clock is the only signal necessary to generate the sequence. When clocked, the register in Figure 1 shifts all contents to the right direction. The sequence a_n (where n is any integer) propagates through with each term generated linearly from the proceeding m terms according to the following formulas [31]:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_m a_{n-m} = \sum_{i=1}^{m} c_i a_{n-i} \tag{1}$$

All terms in here are binary, c_1 to c_m are connection variables where $c = 1$ denotes connection and $c = 0$ is no connection. If the sequence is periodic with period N (i.e., $a_n = a_{n+N}$), Only considering the nonnegative index of the sequence $\{a_n\}$, the generating function of the sequence can be defined as [31]:

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

where D is the delay operator, and the power of D of each term of this polynomial corresponds to the number of units(clock cycles) of delay for that term.

Then G(D) can be written as [31]:

$$G(D) = \sum_{n=0}^{\infty} D^{nN}\left(a_0 + a_1 D + a_2 D^2 + ... + a_{N-1} D^{N-1}\right)$$

$$= \frac{a_0 + a_1 D + a_2 D^2 + ... + a_{N-1} D^{N-1}}{1 + D^N} \qquad (3)$$

The initial contents of the registers and the feedback logic circuit determine the successive contents of the registers. If the LFSR reaches zero state at some time, it would always stay in the zero state. There are exactly $2^m - 1$ in the nonzero states for an ***m***-cell LFSR and the period of the pseudo-noise sequence produced by an ***m***-cell LFSR cannot exceed $2^m - 1$. The sequence of period $2^m - 1$ generated by an ***m***-cell LFSR is known as a ***maximal-length (ML)*** sequence. For all positive integers ***m*** there exist polynomial of degree ***m*** with exponent equal to $2^m - 1$ (largest possible value). Such polynomials are called primitive polynomials. Primitive polynomials are irreducible. Using the Berlekamp-Massey algorithm [59] any LFSR with a feedback polynomial of degree ***m*** can be predicted with the knowledge of only **2*m*** consecutive output bits.

In the CDMA and other digital communications systems, the need exist to use random sequences that can be implemented at the transmitter and receivers. The random sequences that are needed in these applications are finite sequences and no finite sequence is ever truly random [33]. The maximal length linear feedback shift register provides a solution to this problem by offering a simple way to generate binary sequence that look like random.

In general, there are three certain properties that can be associated with randomness of binary sequences when an ideal coin is tossed consecutively, with heads represent as +1 and tails as –1 [31]:

1. The number of heads is approximately equal to the number of tails
2. Runs of consecutive heads or of consecutive tails frequently occur, with short runs being more frequent than long runs.
3. Random sequences posses a special kind of auto-correlation function, peaked in the middle and tapering off rapidly at the ends.

The periodic binary sequences generated by LFSR have some or all of the following randomness characteristics [33] (also referred as the randomness postulates).

R-1. In every period, the number of + l's *is* nearly equal to the number of –1's. The disparity does not to exceed 1. Therefore,

$$\left| \sum_{n=1}^{p} a_n \right| \leq 1 \tag{4}$$

R-2. In every period, half the runs have length one, one-fourth has length two, one-eighth has length three, and so on, as long as the number of runs so indicated exceeds 1. Moreover, for each of these lengths, there are equally many runs of +1's and of -1's. The *total* number of runs of + l's equals the *total* number of runs of - l's, because the runs of these two types alternate.

R-3. The auto-correlation function $C(\tau)$ is two-valued. Explicitly

$$C(\tau) = \sum_{n=1}^{p} a_n a_{n+\tau} = \begin{cases} P & if\ \tau = 0 \\ K & if\ 0 < \tau < p \end{cases} \quad (5)$$

Any sequence with these properties is called a pseudo-random sequence. Since the noises in communication systems are random in nature, pseudo-random sequence is called pseudo-noise (PN) sequence [33]. We will use the term PN sequence for the pseudo random sequence in the remaining of this thesis.

2.1.1 The Use of LFSR in the CDMA Systems

Ideally, the spreading codes in the spread spectrum systems would be true random binary sequences, as might be produced by the consecutive tosses of a fair, memory-less coin. However, this is not practical since both the base station (BS) transmitter and mobile station (MS) receiver require to generate the same sequences with time-aligned for communicating with each other. The receivers thus must perform synchronization searches by changing their offset hypothesis until the transmitter timing is located. High spreading rates (1.2288 Mcps for spreading rate 1 and other higher spreading rates) are also required in order to achieve high capacity in the CDMA environment. Therefore, if truly random sequences were to be used then they would have to be pre-generated and pre-stored in all transmitters, with a matching copy in all receivers. For this reason, deterministic methods of generating the PN sequences are preferable. The CDMA wireless systems based on IS-2000 and IS-95 industry standards use LFSR for generating the pseudo-noise sequences used for the signal spreading and de-spreading in the forward and reverse link channels. The next three sections describe the orthogonal and quadrature spreading used in the IS-2000 CDMA systems as defined in [1][68][79].

2.1.1.1 Orthogonal Spreading

In the CDMA wireless systems based on IS-95 and IS-2000, each of the code channel transmitted on the forward CDMA Channel is spread with a Walsh function at a fixed chip rate of 1.2288 Mcps (for the spreading rate 1) to provide orthogonal channelization among all code channels on a given forward CDMA Channel [1][68]. One of 64 time orthogonal Walsh functions is used. This means that a code channel that is spread using Walsh function ***n*** is assigned to code channel number ***n*** (where *n is from* 0 to 63). Walsh function time alignment is such that the first Walsh chip, designated by 0, begins at an even second time mark referenced to base station transmission time. The Walsh function spreading sequence repeats with a period of 52.083 μs (64/1.2288 Mcps), which is equal to the duration of one forward traffic channel modulation symbol. Code channel number zero is always assigned to the Pilot channel. When the Sync channel is present, it is assigned code channel number 32. If Paging Channels are present, they are assigned to code channel numbers one through seven in sequences. The remaining code channels are available for assignment to the forward traffic channels.

2.1.1.2 Quadrature Spreading

After the orthogonal spreading on the forward CDMA channel, each code channel (pilot, sync, paging, or forward traffic channel) is spread in quadrature [1][68][79]. The spreading sequence is a quadrature sequence of length 2^{15} pseudo-noise (PN) chips in length. For SR 1, 1 chip is equal to 1/1228800 or approximately 813.802 ns. This sequence is called the pilot PN sequence and consist a pair of PN sequences generated by ML LFSR. This pair of PN sequences is used to spread the forward CDMA Channel and the reverse CDMA

Channel. Each different base station is identified by different pilot PN sequence offsets. The pilot PN sequence is based on the following characteristic polynomials [27][68]:

$$P_I(x) = x^{15} + x^{13} + x^9 + x^8 + x^7 + x^5 + 1 \quad (6)$$

for the in-phase (I) sequence

and

$$P_Q(x) = x^{15} + x^{12} + x^{11} + x^{10} + x^6 + x^5 + x^4 + x^3 + 1 \quad (7)$$

for the quadrature-phase (Q) sequence.

2.1.1.3 Pilot PN Sequence Offset

Each base station uses a time offset of the pilot PN sequence to identify a forward CDMA Channel. Time offsets are reused within a CDMA cellular system. Distinct Pilot channels are identified by an offset index (0 through 511 inclusive). This offset index specifies the offset value from the zero offset pilot PN sequence[27][68]. The zero offset pilot PN sequence is such that the start of the sequence is output at the beginning of every even second in time, referenced to base station transmission time. The start of the zero offset pilot PN sequence for either the I or Q sequences are defined as the state of the sequence for which the previous 15 outputs were '0'. Five hundred twelve unique values are possible for the pilot PN sequence offset. The offset (in PN chips) for a given pilot PN sequence from the zero shift pilot PN sequence equals the index value multiplied by 64. For example, if the pilot PN sequence offset index is 15, the pilot PN sequence offset will be 15 x 64 = 960 PN chips or approximately 781.25μs. In this case the pilot PN sequence will start 781.25 μs after the start of every even second of time, referenced to base station

transmission time. The same pilot PN sequence offset is used on all CDMA frequency assignments for a given base station [31].

The result of the LFSR signal spreading in IS-2000 based CDMA system with spreading rate 1 is depicted in Figure 2, and it shows the code channel graphs captured on a code domain analyzer in CDMA forward link channels. Although all of the 64 Walsh codes are shown in the figure 2, the codes that are in use are showing with a power above the noise floor. These codes are Pilot channel (Walsh code 0), Paging channel (Walsh code 1), Sync channel (code channel 32) and six traffic channels. Figure 3 shows how the whole CDMA channels looks after the signals in the forward link channels are spread in 1.2288MHz bandwidth using ML LFSR. Figure 3 depicts the completely occupied bandwidth for spreading rate 1 using frequency channel 475 in the personnel communication systems (PCS).

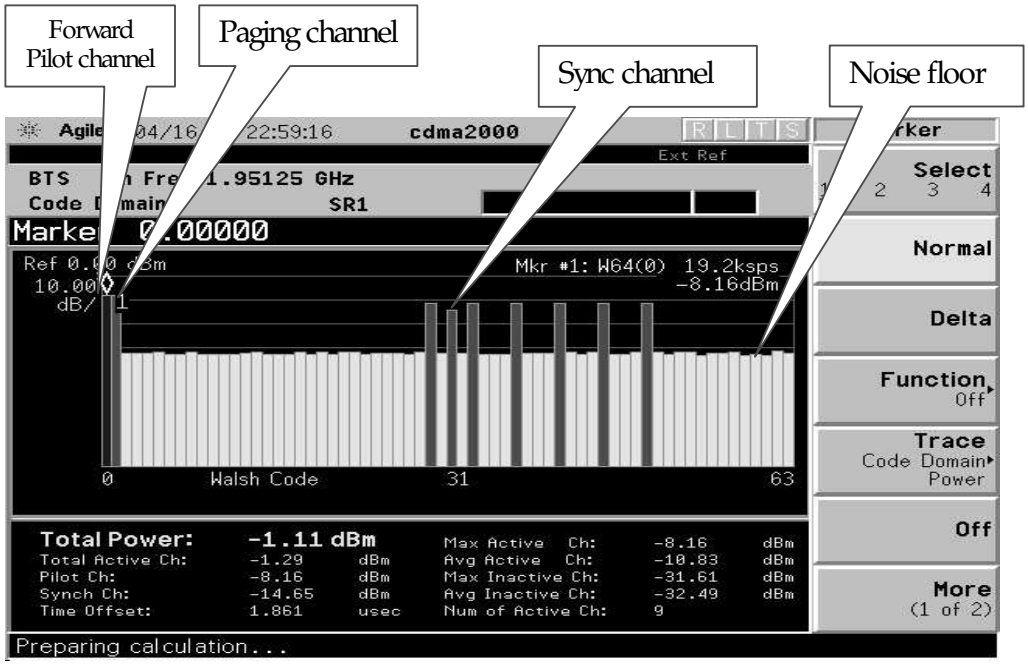

Figure 2. Code Domain Analysis of CDMA Forward Channels

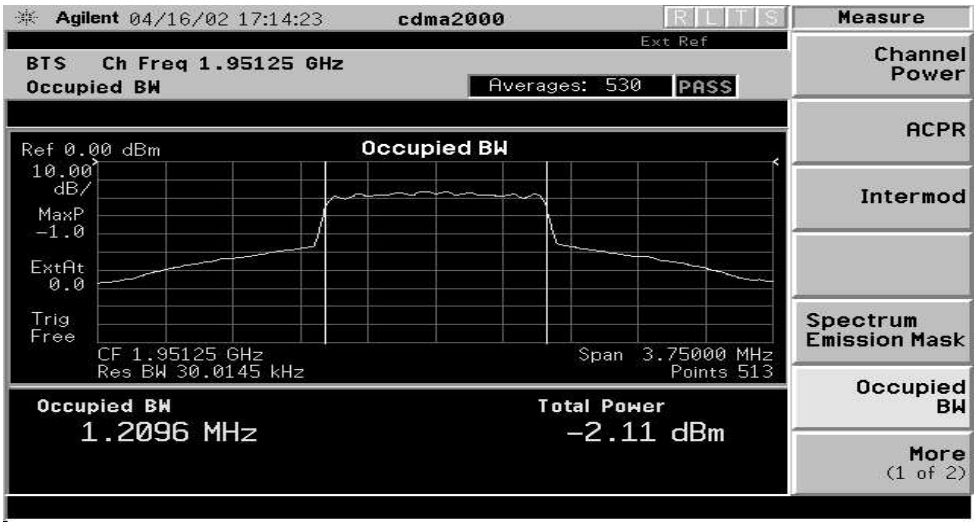

Figure 3. LFSR spread CDMA signal in channel 475 with SR 1

2.2 Genetic Algorithm

GA initiated from the studies of cellular automata, conducted by John Holland and his colleagues at the University of Michigan. Holland's book [47] is generally acknowledged as the beginning of the research of GA [22][23][25][26][30]. GA is under the evolutionary computation umbrella that combines GA, Genetic programming, Classifier systems, Evolutionary programming and Evolutionary strategies. All of these techniques simulate the evolution of individual structures using the processes of selection, mutation and reproduction. Each individual in the population receives a measure of its fitness in the environment. Reproduction focuses attention on high fitness individuals, thus exploiting the available fitness information. Recombination and mutation perturb those individuals, providing general heuristics for exploration. These algorithms sufficiently provide robust and powerful adaptive search mechanisms [14][22][23][25][26][30].

Since the GA is class of stochastic search methods that imitate the character of natural biological evolution, GA operates on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics [14][22][23]. This process leads to the evolution of populations of individuals that are better suited to their environment than the ones that they were created from, just as in natural adaptation. GA model natural processes, such as selection, recombination, mutation, migration, locality and neighborhood. Evolutionary algorithms work on populations of individuals instead of

single solutions. In this way, the search is performed in a parallel manner [14][22][23][25][26][30].

At the beginning of the computation, a number of individuals (the population) are randomly initialized. The objective function is then evaluated for these individuals. The first/initial generation is produced. If the optimization criteria are not met the creation of a new generation starts. Individuals are selected according to their fitness for the production of offspring. Parents are recombined to produce offspring. All offspring will be mutated with a certain probability. The fitness of the offspring is then computed. The offspring are inserted into the population replacing the parents, producing a new generation. This cycle is performed until the optimization criteria are reached. A single population evolutionary algorithm is powerful and performs well on a wide variety of problems [22][23][30][44][63].

Before the early 1980s, the research in GA was mainly theoretical [14][22][44], with few real applications. This period is marked by ample work with fixed length binary representation in the domain of function optimization by, among others, De Jong [23] and Hollstien [48]. Hollstien's work provides a careful and detailed analysis of the effect that different selection and mating strategies have on the performance of a genetic algorithm. De Jong's work attempted to capture the features of the adaptive mechanisms in the family of GA that constitute a robust search procedure. From the early 1980s, the community of GA has experienced an abundance of applications, which spread across a large range of disciplines. Each additional application gave a new perspective to the theory. Furthermore,

in the process of improving performance as much as possible via tuning and specializing the genetic algorithm operators, new and important findings regarding the generality, robustness and applicability of genetic algorithms became available. In the last fifteen years, GA successfully applied to optimization problems in sciences, engineering and the business applications [44].

The evolutionary algorithms differ substantially from more traditional search and optimization methods. The most significant differences are [63]:

a) Evolutionary algorithms search a population of points in parallel, not just a single point.
b) Evolutionary algorithms do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the directions of search.
c) Evolutionary algorithms use probabilistic transition rules, not deterministic ones.
d) Evolutionary algorithms can provide a number of potential solutions to a given problem.

Table 1 shows a brief description of the correspondence between natural and artificial terminology [30][14].

Table 1. GA and Natural Terminology Comparison

Natural	Genetic Algorithm
chromosome	String
Gene	feature, character or detector
Allele	feature value
Locus	string position
genotype	structure, or population
phenotype	parameter set, alternative solution, a decoded structure

The next following sections discuss some of the basic parameters used in the GA, specifically the parameters that are used in our research.

2.2.1 Initial Population

GA starts with a population of strings to be able to generate successive populations of strings afterwards. The initialization is usually done randomly. This means, with binary strings for example, that every allele is set to 0 or 1, with each value having a chance of 50 % to occur. This can be achieved by simulating the tossing of a coin with head = 1 and tail

= 0 [22][30][44][61]. Once a population is generated, all individual in that population has to be evaluated to distinguish between good and bad individuals. This is generally achieved by mapping the objective function to a fitness function [14][30]. In this study, we chose initial population size according to the number of stages in the LFSR.

2.2.2 Selection

The individuals that are chosen for mating (recombination) and how many offspring each individual produces are determined by the selection method. In our study, the following four selection methods were evaluated:

a) Roulette-wheel selection

b) Stochastic universal sampling

c) Truncation selection

d) Tournament selection

2.2.2.1 Roulette wheel selection

Roulette-wheel selection method is a simple stochastic selection that is analogous to a roulette wheel with each slice proportional in size to the fitness [18]. Roulette-wheel selection is also called stochastic sampling with replacement. In this selection, the individuals are mapped to adjacent sections of a line, such that each individual's section is equal in size to its fitness. A random number is generated and the individual whose section spans the random number is selected. The process is repeated until the desired number of individuals is obtained (called mating population) [29][63].

2.2.2.2 Stochastic universal sampling

Stochastic universal sampling provides zero bias and minimum spread. In the stochastic universal sampling, the individuals are mapped to adjacent parts of a line, such that each individual's part is equal in size to its fitness exactly as in roulette-wheel selection. Then equally spaced pointers are placed over the line as many as there are individuals to be selected. Suppose *NPointer* the number of individuals to be selected, then the distance between the pointers are 1/*Npointer* and the position of the first pointer is given by a randomly generated number in the range [0,1/*NPointer*] [29][63].

2.2.2.3 Truncation selection

The truncation selection is used the breeders for large populations/mass selection [63]. In truncation selection, all individuals are sorted in proportion to their fitness. Then the best individuals are selected for parents. These selected parents produce uniform at random offspring. Selection intensity or Truncation threshold parameter is used in the truncation selection to define the proportion of the population to be selected as parents. Individuals below the truncation threshold do not produce offspring. Truncation threshold uses values ranging from 50%-10% [29][63].

2.2.2.4 Tournament selection

In tournament selection method, a tournament size parameter (tournament size) which takes values ranging from 2 to number of individuals in populations are chosen randomly from the population. Then the best individual from this group is selected as parent. This process is repeated as often as individuals must be chosen. These selected parents produce

uniform at random offspring [29][63]. In this study, the tournament selection method has been found to produce the best performance results for the adaptive algorithm, and therefore, it has been chosen as the selection method.

2.2.3 Recombination (crossover)

The function of the crossover operator is to allow the advantageous qualities to be spread throughout the population in order that the population as a whole may benefit from this chance discovery [14]. The crossover operator randomly chooses a crossover point where two parent chromosomes 'break', and then exchanges chromosome parts after that point. Two new offspring are created from this result [14][29][63]. In our study, we used multi-point crossover algorithms. In multi-point crossover, *i* crossover positions *are* chosen at random with no duplicates and sorted into ascending order. Then, the variables between successive crossover points are exchanged between the two parents to produce two new offspring [14][44][63].

2.2.4 Mutation

After recombination, every offspring undergoes mutation. Offspring variables are mutated by small perturbations (size of the mutation step), with low probability [14][44][63]. We used binary value variables for the mutation in our study.

2.2.5 Reinsertion

After producing offspring, they must be inserted into the population. This is especially important, if less offspring are produced than the size of the original population. Another

case is, when not all offspring are to be used at each generation or if more offspring are generated than needed [14][44][63]. Reinsertion scheme is determined which individuals should be inserted into the new population and which individuals of the population will be replaced by offspring.

This study uses optimized and robust genetic algorithm search methods that have been developed in the last twenty years to produce an adaptive algorithm that can compute the PN sequences of ML LFSR in shorter time.

Chapter 3

3 Computing Initial Value of LFSR

This chapter discusses an adaptive algorithm for computing the initial values (IVAA) of ML LFSR using GA as described in [30]. This method assumes that the PFC of the LFSR is known. First, the SRs are initialized randomly with at least one register having non zero value. Then GA is applied using selection, crossover, and mutation methods. The description of the IVAA process is provided below. The IVAA reads the output of the LFSR and then estimates the sequence bits using known PFC and GA [30] generated sequences [31]. The objective of the algorithm is to find the initial value of the LFSR generated sequences without waiting for the completion of the LFSR cycle period. The repeated period of the ML LFSR is 2^m-1. First, the IVAA reads and stores the output data generated by the LFSR. Then the IVAA initializes its initialization parameters and uses GA to generate binary population. The steps that the IVAA follows are shown in Table 2.

Table 2. IVAA Process Steps

1. Restore the received bit [a_n] $[a_n]_{Received} = [a_1, a_2, a_3, \cdots, a_k]$
2. Initialize the Initialization parameters: • Clock, Clk • Initial population size, N. • Number of shift registers, m. • Probability of crossover, Pc. • Probability of Mutation, Pm.
3. Generate random initial value population $P_{IV} = [N \times m]$
4. Calculate a_n using P_{IV} and known polynomial feedback coefficients: $[a_n]_{Calculated} = \sum_{i=1}^{m} C_i a_{i-1}$, update and restore all registers values.
5. Assign Fitness value of each IV population based on how each population's $[a_n]_{Calculated}$ matches to the $[a_n]_{Received}$. If $[a_n]_{Calculated} = [a_n]_{Received}$ Assign fitness value = 10 Else assign fitness value = 10/2
6. Perform selection process and choose the fittest population. Use Tournament selection method.

7. Perform Recombination (Crossover) Process.

   Use Probability of crossover, Pc = 0.70

8. Perform Mutation process

   Use Probability of mutation, Pm = 0.001

9. Save the new population as the current initial values population, P_{IV}

10. Increment the clock by 1 and restore that number as a sub-clock

11. Restore the received bit [a_n]

    $[a_n]_{\mathrm{Re}ceived} = [a_1, a_2, a_3, \cdots, a_k]$, update and restore all registers values.

12. Repeat step 4 and 5 for sub-clock 1

13. Repeat step 4 for sub-clock 2 using results from 12

14. Assign Fitness value of each IV population based on how each population's $[a_n]_{Calculated}$ matches to the $[a_n]_{\mathrm{Re}ceived}$.

    If $[a_n]_{Calculated} = [a_n]_{\mathrm{Re}ceived}$

    Assign fitness value = $2x[FitnessValue]_{previous\_sub-clok}$

    Else

    Assign fitness value = $\frac{[FitnessValue]_{previous\_sub-clok}}{2}$

15. Repeat step 6 to 9

16. Repeat step 6 to 15 until 100% accuracy achieved.

3.1 Initialization

The initialization starts both the LFSR and the GA algorithm. The values of the ***m*** defined shift registers are initialized with non-null value on at least one shift register. In initialization process, the following parameters are initialized: **m, N, Clks, Pc, Pm,** where:

m = number of shift register (SR) cells.

N = Initial value population size

Clks = Number of clocks that the LFSR will be clocked

Pc = crossover probability

Pm = mutation probability

3.2 Initial Value Population Generation

After initializing the shift registers, the received data value a_R will be restored. The GA algorithm generates initial population for the initial value IV_i using initial parameters defined in section 3.1. An output data a_C will be calculated using the known polynomial feedback coefficients on the shift registers and the randomly generated population of the initial value. This calculation is the XORing and shifting operation of the SR contents according to the PFC connection parameters.

3.3 Fitness Value Assignment

In each clock sequence, the received bit a_R from the LFSR output is restored and that value is compared to the computed value a_C. A fitness value is assigned to each

individual in the population based on how each individual's a_C matches to the current a_R. The rank-based [14][19][30] fitness assignment method is used in the IVAA, where the fitness value that is assigned to each individual depends on its matching to the value on a_R. At the first initialization, if the prediction value on a_C is true, the fitness equal to 10 is assigned to that individual. If it is not true, a fitness value equal to 10/2 is assigned. In the following clocks, the fitness values of the preceding are used as base fitness. Whenever the prediction becomes true, 2 multiply the previous fitness value. When the prediction is not true 2 divides previous fitness value.

3.4 Selection, Recombination, and Mutation Process

Selection process is performed using Roulette Wheel, Universal sampling, Tournament selection and Truncation selection methods of the GA. The Tournament selection performed best in the prediction accuracy and therefore was chosen for the IVAA. The Tournament selection achieved about four clocks faster than other three selection methods. Probability of crossover ***(Pc)*** equal to 0.70 is used during the recombination process. These values are values that are generally used in the literature [61] and worked well for the IVAA. Probability of mutation ***(Pm)*** equal to 0.001, which is in the commonly used range as described in [61] was also used for the IVAA.

3.4.1 Performance comparison for Selection methods

Performances of four GA selection methods were evaluated for the IVAA. This section presents the results of the comparison among the four GA selection methods. A 10 stage LFSR was used with the following initialization parameters:

m = 10, N=100, Clks=40, Pc=0.70, Pm=0.001

The Tournament selection method performed better than other selection method for the IVAA. Figure 4 shows the computation accuracy of all selection methods. Figure 5 shows a plot of average population fitness in each clock period.

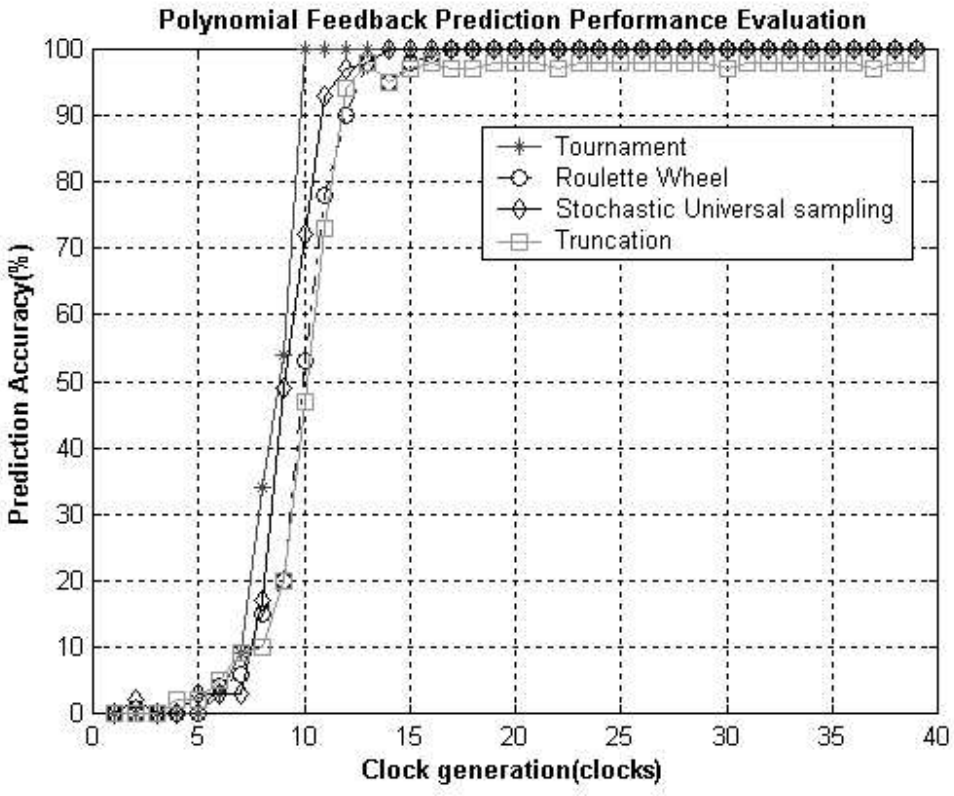

Figure 4. Performance comparison of GA selection methods

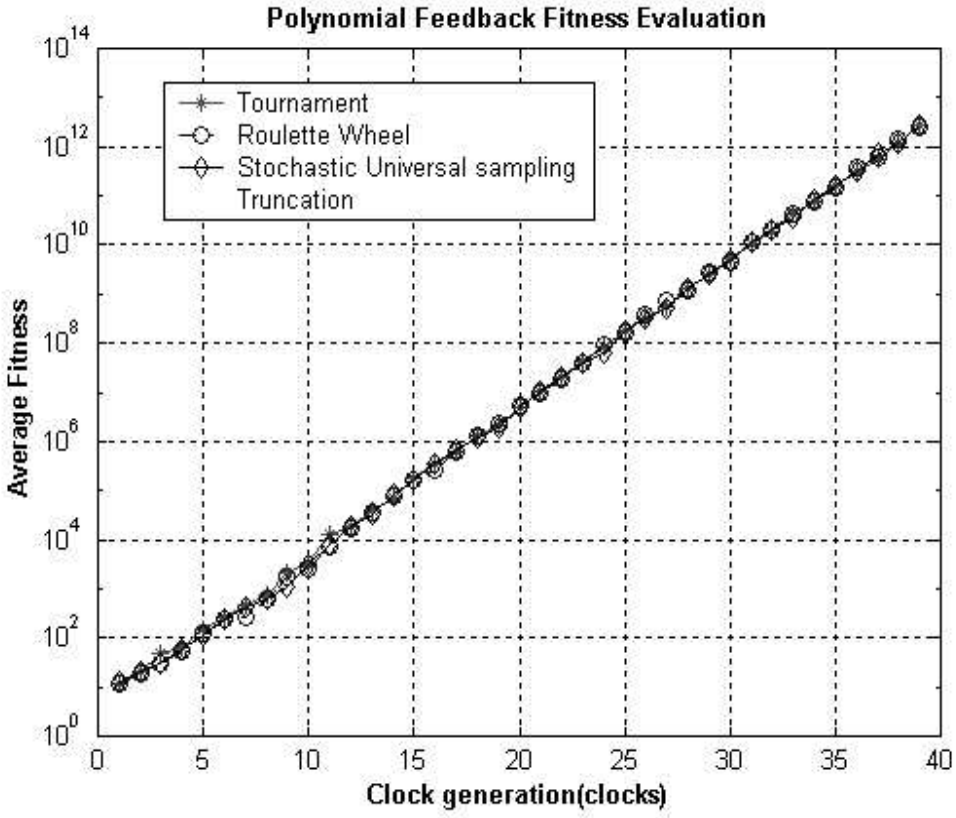

Figure 5. GA selection methods fitness comparison for 10 Stage LFSR

3.5 Performance Test Results

To verify the performance of the IVAA for computing the IVs of the ML LFSR, we tested many different test scenarios using various PFC with MATLAB simulation. In this section, we present the test results from different PFC. Performance of the IVAA is measured in two important categories: computation accuracy and the time (i.e. time it takes the IVAA to compute all IVAA in 100% accuracy). Table 3 lists the test cases that we used to verify the performance of the IVAA, the PFC, the number of shift registers (m), number of initial

population size (N), probability of crossover (Pc) value, probability of mutation (Pm) value. In each test case, we show a plot showing the results of the performance evaluation (i.e. prediction accuracy), and another plot showing the genetic algorithm average fitness performance.

Table 3. IVAA Performance Test Configurations

Test Case	**f(x)**	**m**	**N**	**Pc**	**Pm**
1	$1+x+x^3$	3	30	0.70	0.001
2	$1+x^2+x^4+x^5$	5	100	0.70	0.001
3	$1+x^3+x^5+x^7+x^9+x^{10}$	10	100	0.70	0.001
4	$1+x^4+x^6+x^8+x^{11}+x^{13}+x^{15}$	15	100	0.70	0.001
5	$1+x^4+x^6+x^8+x^{10}+x^{14}+x^{17}+x^{19}+x^{20}$	20	150	0.70	0.001

Test Case 1: IVAA performance

$f(x) = 1 + x + x^3$: m=3, N=30, Pc=0.7, Pm=0.001

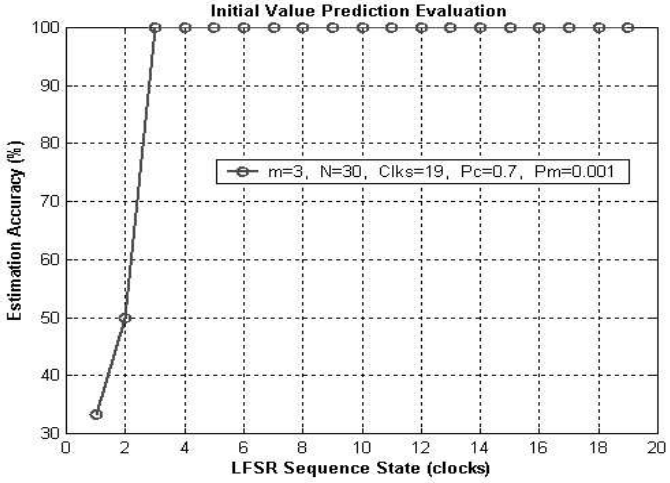

Figure 6. LFSR initial value prediction Performance with m=3 and N=30.

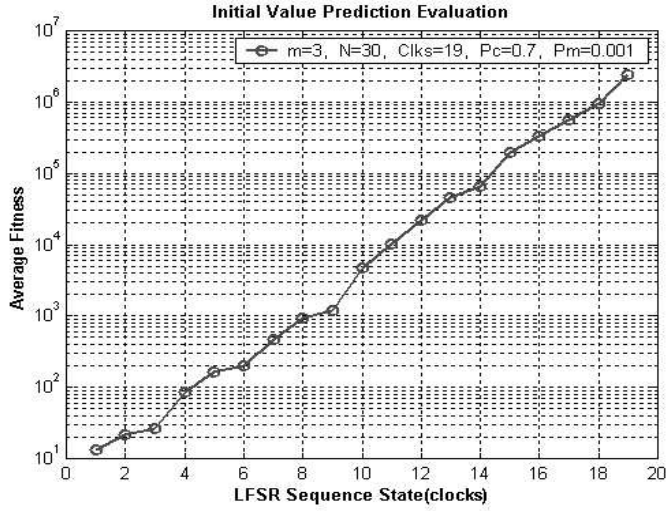

Figure 7. Average Fitness Evaluation with m=3, N=30

Test Case 3: IVAA Performance

$f(x) = 1 + x^2 + x^4 + x^5$: m=10, N=100, Pc=0.7, Pm=0.001

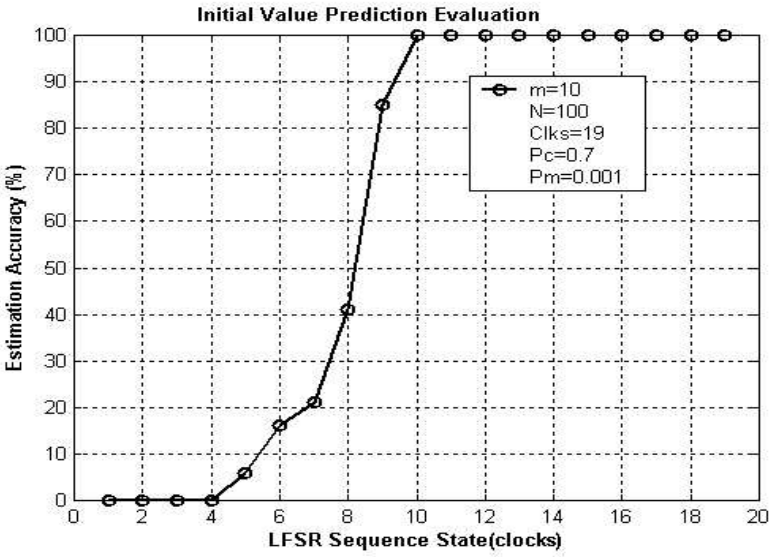

Figure 8. Initial Value performance Evaluation with m=10, N=100.

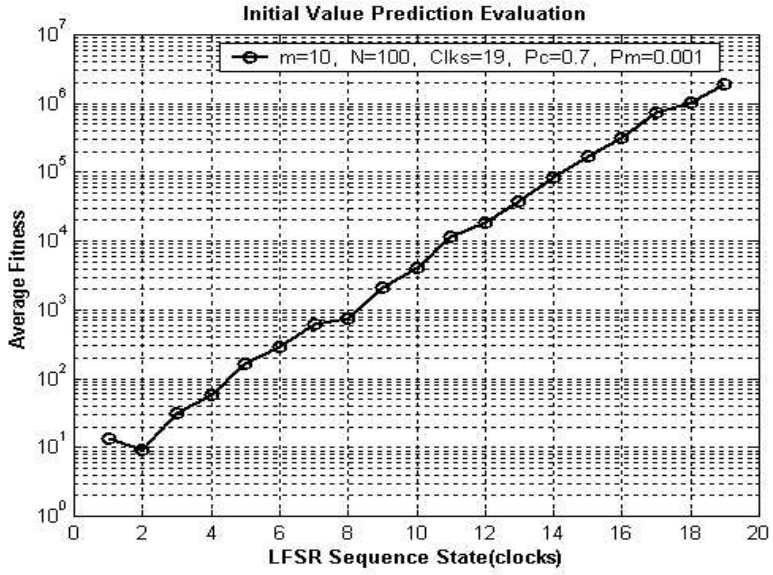

Figure 9. Average Fitness Evaluation with m=10, N=100

Test Case 4: IVAA Performance

$f(x) = 1 + x^3 + x^5 + x^7 + x^9 + x^{10}$: m=15, N=100, Pc=0.7, Pm=0.001

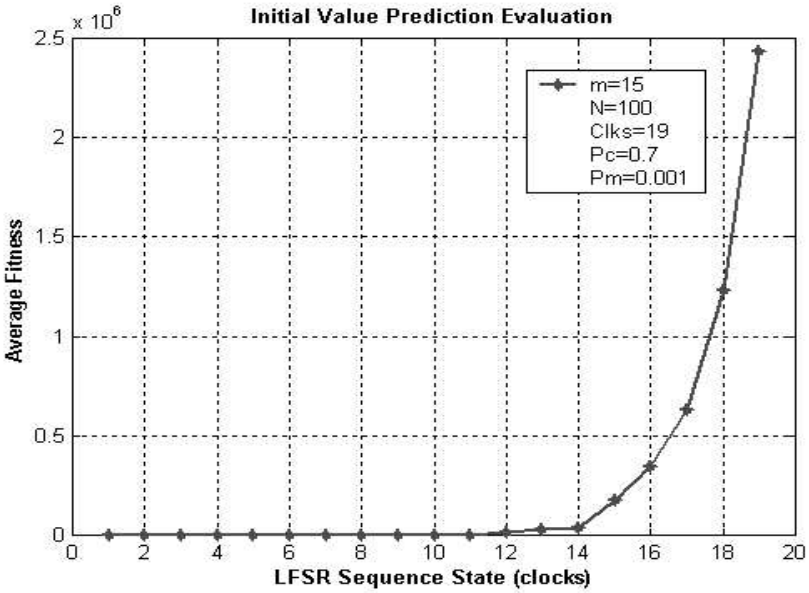

Figure 10. Average Fitness Evaluation with m=3, N=30

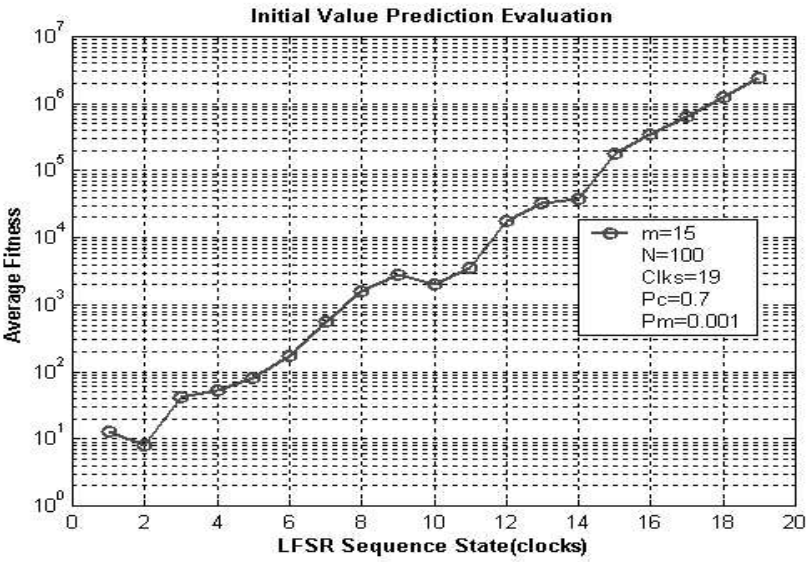

Figure 11. Average Fitness Evaluation with m=15, N=100

Test Case 5: IVAA Performance

$f(x) = 1 + x^4 + x^6 + x^8 + x^{11} + x^{13} + x^{15}$ m=20, N=100, Pc=0.7, Pm=0.001

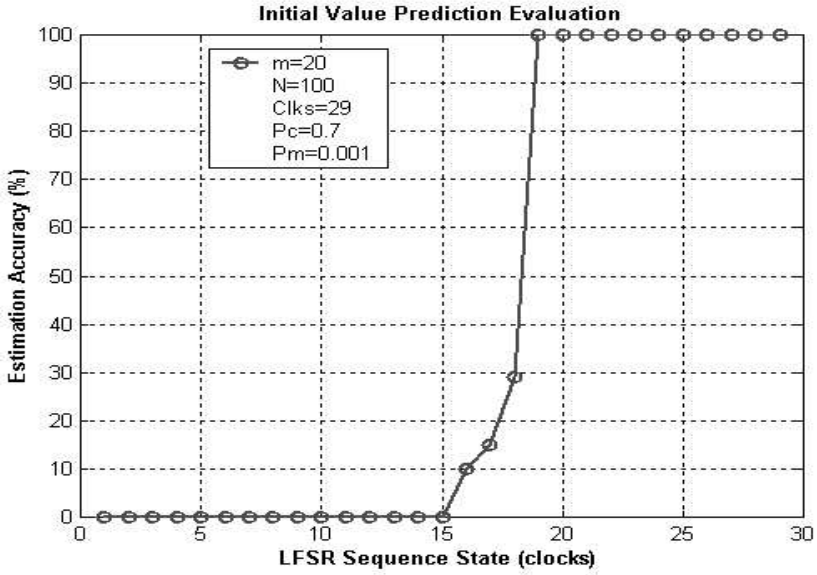

Figure 12. Initial value Accuracy Evaluation with m=20, N=100

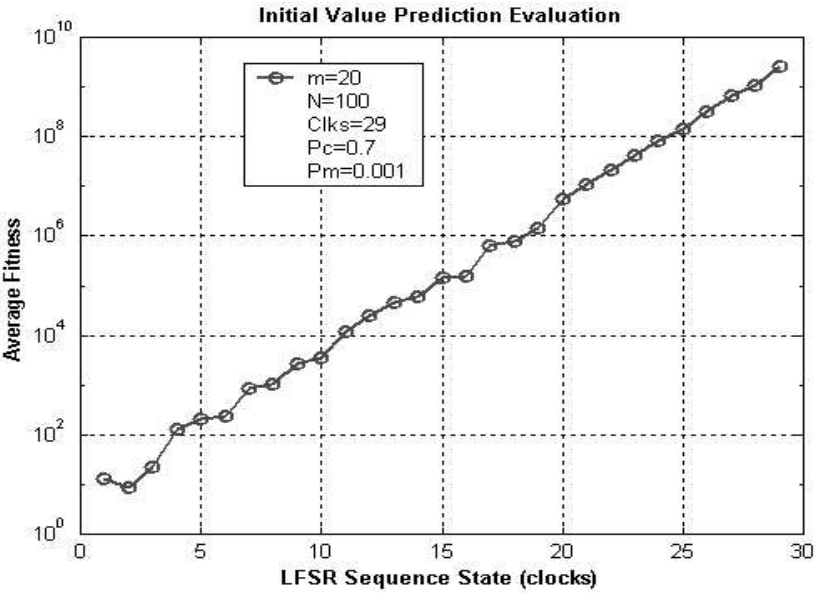

Figure 13. Initial value Fitness Evaluation with m=20, N=100

All of the test results in Figure 6 to Figure 13 show that the IVAA computed the correct initial value of the ML-LFSR on around the $\boldsymbol{m}^{th}$ clock shift of each test case (where $\boldsymbol{m}$ is the number of the shift register).

Chapter 4

4 Computing I and Q Pilot PN Sequences in CDMA Systems

In chapter 3, we presented an IVAA that computed the IV of ML-LFSR when the PFC was known. In this chapter, we will show an implementation of the IVAA in a practical application in wireless Code Division Multiple Access (CDMA) system.

The first time that the power of the Code Division Multiple Access (CDMA) MS is turned on, the MS goes through a call processing states. The first state is called the initialization state. During this state, the MS synchronizes with the BS timing using the Global Positioning System (GPS) timing reference on the CDMA Sync channel and the forward Pilot channel. The MS acquires and synchronizes to the CDMA system while it is in the initialization state. In the current IS-95 [78] and IS-2000 [1] standard based CDMA systems, to obtain In-phase (I) and Quadrature-phase (Q) Pilot PN sequences for time alignment, a zero is inserted in the LFSR sequences after 14 consecutive "0" outputs. We have implemented and tested the IVAA that was presented in chapter 3 to obtain the I and Q Pilot PN sequences during the quadrature spreading. IVAA reduces the computation time to obtain the I and Q pilot PN sequences in CDMA systems, and results faster CDMA MS synchronization time.

When a user turns on the power of the Code Division Multiple Access (CDMA) mobile station, the station acquires and synchronizes to the CDMA system. First, the station

determines the system, and then it acquires the Pilot and Sync channels. In General, the MS has 15 seconds to find a Pilot channel and another 1 second to receive valid Synch message before it attempts again to determine the system. In non-fading radio frequency (RF) environment, we found that it may take a time between two to five seconds from the time the user releases the power button to the time that phone is fully ready (idle mode) to originate a successful calls. In the second and third generation CDMA, in forward link (from base station to mobile station), each of the CDMA code channel (Pilot, Sync, Paging, and forward traffic channel) is spread using orthogonal spreading followed by quadrature spreading. The quadrature spreading is a quadrature sequence of length 2^{15}-1 PN chips (c). The chip rate in the spreading rate (SR) one is 1228800 chips per second (cps). Therefore, the chip duration is approximately is $1/1.2288\times10^{6} \approx 0.814\mu s$ [1][27][68]. These sequence are called the Pilot PN sequences and consist of pair of PN sequences generated by ML LFSR [31][32]. In the current CDMA wireless systems, the I and Q is obtained using algebraic computation. In this approach, to obtain I and Q Pilot PN sequences for time alignment, a zero is inserted in the LFSR sequences after 14 consecutive "0" outputs. The IVAA discussed in chapter 3 computes the I and Q Pilot PN sequences and reduces the base station and MS initial synchronization time. The IVAA finds the forward Pilot Q and I phases using GA adaptive synthesis in a shorter period of time comparing the to the currently used approach [27][68][80]. The IVAA application focuses the computation of the I and Q in the quadrature spreading during the CDMA phone and system synchronization process. In particularly, the quadrature spreading sequence of length $2^{15}-1$ PN sequences (short code) in the spreading rate 1. The long code is another longer

PN sequence $(2^{41}-1)$ that is used for scrambling on the forward CDMA Channel and spreading on the reverse CDMA Channel. On the forward and the reverse traffic channel, the long code provides limited privacy. In addition, the long code uniquely identifies a MS on the reverse traffic channel [27][68][80].

The first section of this chapter provides an overview of the CDMA system quadrature spreading. The forward and reverse link call processing procedures are briefly explained to show the call processing states that are associated with the MS and base station synchronization process. The characteristic polynomials of LFSR that are used to generate the PFC of the I and Q are described in this section. In addition, the forward Pilot channel, the Sync channel, and CDMA system timing are explained in the first section. Section 2 presents the IVAA that computes I and Q Pilot PN sequences. Section 3 presents simulation test results and performance evaluation of the IVAA in channels with present of noise and channels without noise. This section also presents simulation test results of computational complexity of the algorithm to assess the practical implementation of the algorithm.

4.1 CDMA Call Processing States

The MS and BS synchronization process [1][68][27] in the CDMA systems takes place in the call processing states. The call processing can be grouped into two parts: the MS call processing and BS call processing. Brief descriptions of the two call processing states are provided below. After the CDMA MS gets turn on, it goes through different call processing states. The first state is the MS initialization state. In this state, the mobile determines the

CDMA system and attempts to synchronize after locking into the CDMA base station Pilot PN offset. Figure 14 shows the forward and reverse link CDMA channels.

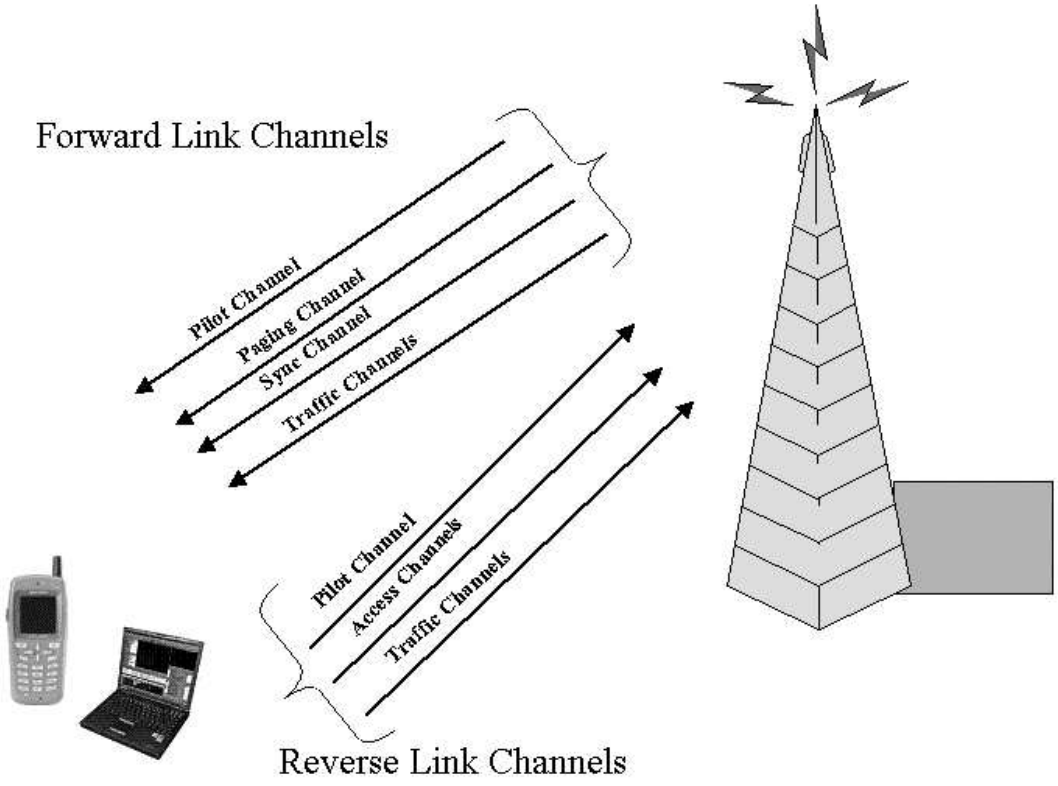

Figure 14. Forward and Reverse Link CDMA Channels

4.1.1 MS Call Processing

After a user presses the power-up button on the mobile station, the call setup goes through different call processing states. The CDMA MS call processing states are summarized below [68]:

1. **Mobile Station Initialization State:**

   During this state, the MS selects and acquires the CDMA system. The MS initialization states consists of the following sub-states:

   a) System determination:

      In this sub-state, the MS selects which system to use.

   b) Pilot Channels Acquisition:

      In this sub-state, the MS acquires the Pilot Channel of a CDMA system.

   c) Sync Channel Acquisition:

      In this sub-state, the MS obtains system configuration and timing information for a CDMA system.

   d) Timing change:

In this sub-state [68], the MS synchronizes its timing to that of a CDMA system.

2. **Mobile Station Idle State:**

   During this state, the MS monitors signaling messages on the Paging channels.

3. **System Access State:**

   During this state, MS sends origination attempt messages to the base station using the access channel.

4. **Mobile Station Control on the Traffic Channel State:**

   During this state, the MS communicates with the BS using forward and reverse traffic channels.

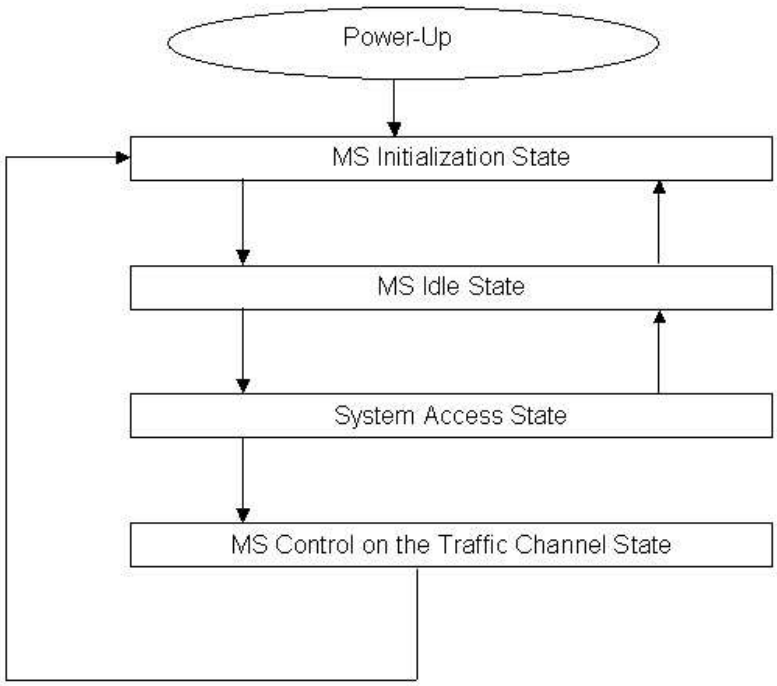

Figure 15. MS Call Processing

4.1.2 BS Call Processing

The BS call processing [68] consists of the following call processing types:

1. Pilot and Sync channel processing
2. Paging channel processing
3. Access channel processing
4. Traffic channel processing

4.2 CDMA MS and BS Time Alignment

A CDMA MS within the coverage area of a BS uses the Pilot channel for synchronization. The pilot channel is a reference channel that allows a MS to acquire the timing of the forward CDMA channel and as a result provides a phase reference for consistent demodulation. The CDMA forward link Pilot channel is an un-modulated spread spectrum signal. Each CDMA BS is identified by a different forward Pilot PN sequence offsets. The Pilot channel signals are transmitted continuously by each of CDMA BS on forward link. Generally, about 20% of the BS forward power is assigned to the Pilot channel [76]. Figure 16 shows the spreading of the CDMA Pilot channel [68].

The MS monitors the Pilot channel for every CDMA channel at all times except when not receiving in the slotted mode [1][27][68]. Slotted mode is an operation mode of the MS in which the MS monitors only selected slots on the Paging channel. During the Pilot and Sync channel processing, the MS acquires and synchronizes to the CDMA system while it is in the MS initialization state. In order to obtain the I and Q pilot PN sequences (of period 2^{15}-1), a '0' is inserted in the LFSR sequence after 14 consecutive '0' outputs. This occurs only once in each cycle period. Therefore, the Pilot PN sequences have one run of 15 consecutive '0' outputs instead of 14. The chip rate (RC) for the pilot PN sequence is 1228800 chips for the spreading rate (SR) 1.

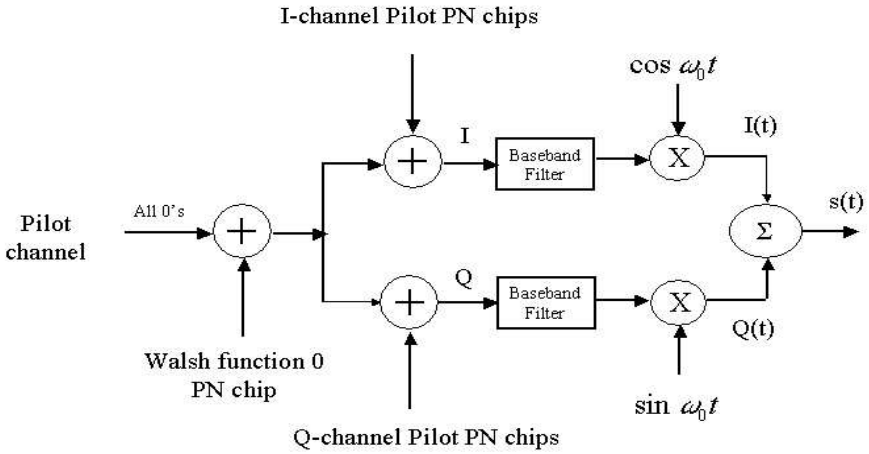

Figure 16. Forward CDMA Pilot Channel Spreading

The pilot PN sequence period (Tps) is defined as:

$$Tps = 32768/1228800HZ \approx 26.666ms \quad (8)$$

In the current CDMA implementation, in every 2 seconds the Pilot PN sequence recurs 75 times. Figure 17 illustrates the nominal relationship between the MS and BS's transmit and receive time reference [1][78]. The MS establishes a time reference that is utilized to derive timing for the transmitted chips, symbols, frame slots, and system timing. Under steady state conditions, the MS time reference is within 1μs of the time of occurrence of the earliest multi-path component being used for demodulation as measured at the MS antenna connector [1][27]. If another multi-path component belonging to the same Pilot channel or

to a different Pilot channel becomes the earliest arriving multi-path component to be used, the MS time reference tracks to the new component. When receiving the forward traffic channel, the MS time reference is used as the transmit time of the reverse traffic channel.

All BS digital signals are transmitted using common CDMA system-wide time reference. This time reference uses the Global Positioning System (GPS) time measurement and it is traceable to, and synchronous with, Universal Coordinated Time (UTC) [1]. GPS and UTC are different by an integer number of seconds (in particularly the number of leap second corrections added to UTC since January 6, 1980. The start of CDMA System Time is January 6, 1980 00:00:00 UTC, which corresponds with the start of GPS time [1]. System time keeps track of leap second corrections to UTC but does not use these corrections for physical adjustments to the system time clocks. The MS aligns the I and Q PN sequences such that the first chip on every even second mark as referenced to the transmit time reference is the '1' after the 15 consecutive '0's. Time measurements are made at the antennas of BSs and the RF connectors of the MSs. A pair of 2^{15}-1 length PN sequences is used for the forward CDMA channels and the spreading rate 1 reverse CDMA channel. The initial state of the 2^{15} PN sequence, for both I and Q channels, is the state in which the output of the PN sequence generator is the first '1' output following 15 consecutive '0' outputs. Figure 17 shows simple sketch of the CDMA system time [1][78].

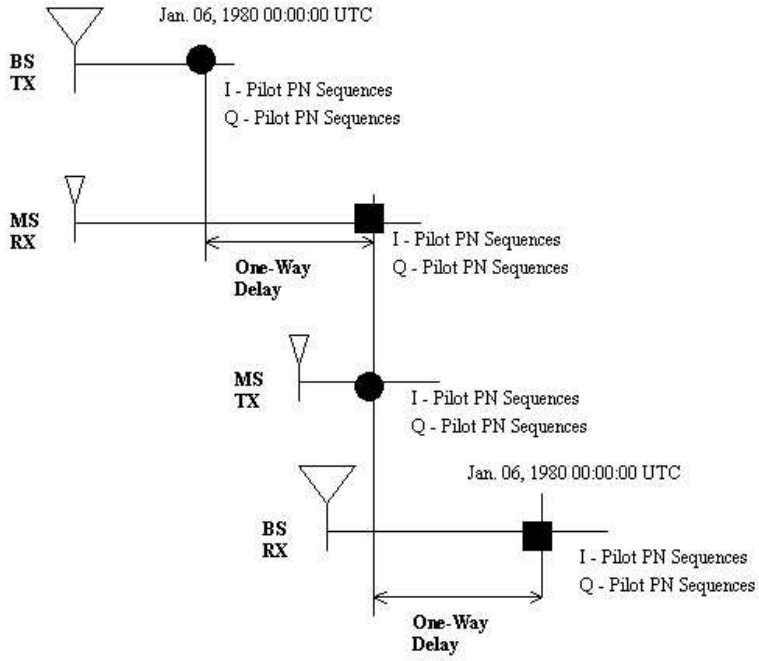

Figure 17. CDMA System Timing Diagram

4.2.1 SYNC Channel

The Sync channel carries important time information that is used by MSs operating within the coverage area of the BS to acquire initial time synchronization [1]. A Sync Channel frame is about 26.667 ms in duration. Within the same BS, the I and Q channel pilot PN sequences for the Sync Channel use the same pilot PN sequence offset as for the forward

Pilot channel. The Sync channel (and all other channels) is spread with the same pilot PN sequence, and the frame and inter-leaver timing on the Sync Channel are aligned with the pilot PN sequence. Therefore, once the MS achieves pilot PN sequence synchronization by acquiring the forward Pilot channel, the synchronization for the Sync channel is immediately known. The start of the inter-leaver block and the frame of the Sync channel align with the start of the Pilot PN sequence being used to spread the forward CDMA channel. Table 4 shows Sync channel message parameters [27] that the CDMA BS sends to the MS over the air.

Table 4. CDMA Sync Channel Message Parameters

Sync Channel Message
MSG_TYPE - Message type.
P_REV - Protocol revision level.
MIN_P_REV - Minimum protocol revision level.
SID - System identification.
NID - Network identification.
PILOT_PN - Pilot PN sequence offset index.
LC_STATE - Long code state.
SYS_TIME - System time.
LP_SEC - The number of leap seconds that have occurred since the start of System Time:
LTM_OFF - Offset of local time from System Time.
DAYLT - Daylight savings time indicator.
PRAT - Paging Channel data rate.

4.3 Adaptive Algorithm Approach

The Current CDMA systems use ML-LFSR with a known and standard defined PFCs. Therefore, the IVAA that we described in chapter 3 can be used for computing the I and Q of the forward Pilot PN sequences during the time alignment. The IVAA utilizes the GA as a search and optimization tool. As described in section 4.1, the algebraic approach [1] that

is currently used in the CDMA systems adds additional zero to the PN sequence. This added zero is used for the SOM reference to time align the start of the BS and MS PN sequence. The current approach requires a minimum of one LFSR cycle period (2^{15} PN chip). The IVAA uses the received data with the known coefficients of the LFSR to compute the initial values (SOM) bit of the LFSR. The IVAA performs well in finding the SOM bit, and the test results presented in next few sections show that it maintains good performance in both noisy and non-noisy channel environments. Figure 18 shows the forward Pilot channel quadrature spreading [27][68].

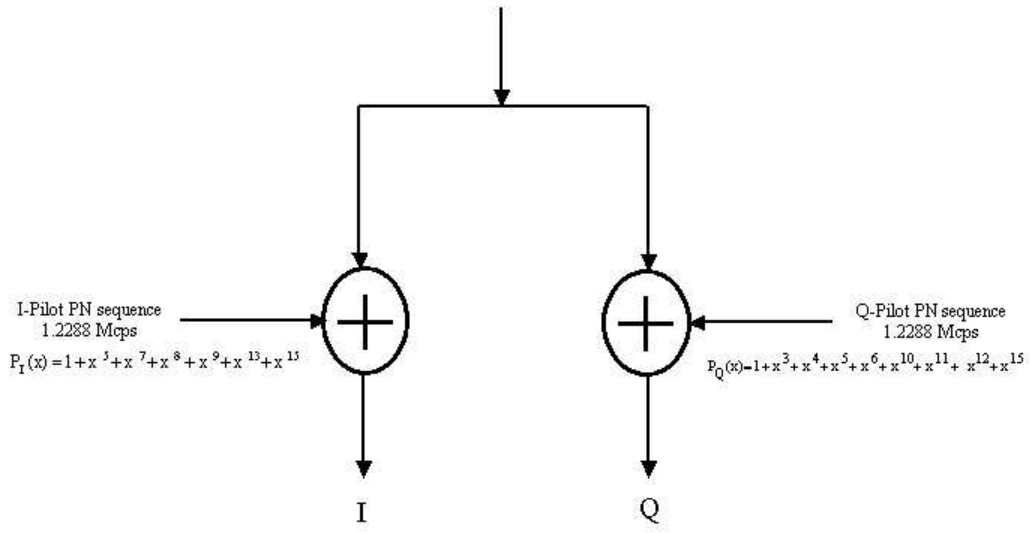

Figure 18. Forward Pilot Channel Quadrature Spreading

The pilot PN sequence is based on the following ML LFSR characteristic polynomials [1] [27][68][78]. For the I sequence, the characteristic polynomial is:

$$F_I(x) = x^{15} + x^{13} + x^9 + x^8 + x^7 + x^5 + 1 \quad (9)$$

or F_I= (1000010111000101)

and for the Q sequence is:

$$F_Q(x) = x^{15} + x^{12} + x^{11} + x^{10} + x^6 + x^5 + x^4 + x^3 + 1 \quad (10)$$

or F_Q = (1001111000111001).

The pilot PN sequences are generated from the characteristic polynomials $F_I(x)$ and $F_Q(x)$, respectively. The ML-LFSR sequences I(n) and Q(n), based on the polynomials $F_I(x)$ and $F_Q(x)$ have a period of $2^{15} - 1$. These I and Q pilot PN sequences can be generated using the following reciprocal polynomials [27][68][78]:

$$\begin{aligned} i(x) &= x^n F_I(x^{-1}) \\ &= 1 + x^2 + x^6 + x^7 + x^8 + x^{10} + x^{15} \end{aligned} \quad (11)$$

and

$$\begin{aligned} q(x) &= x^n F_Q(x^{-1}) \\ &= 1 + x^3 + x^4 + x^5 + x^9 + x^{10} + x^{11} + x^{12} + x^{15} \end{aligned} \quad (12)$$

The binary sequences of the LFSR sequences based on the equations 11 and 12 polynomials can be generated using the following linear recursive operations [68]:

$$i(n) = i(n-15) \oplus i(n-10) \oplus i(n-8) \\ \oplus i(n-7) \oplus i(n-6) \oplus i(n-2) \quad (13)$$

and

$$q(n) = i(n-15) \oplus i(n-12) \oplus i(n-11) \oplus i(n-10) \\ \oplus i(n-9) \oplus i(n-5) \oplus i(n-4) \oplus i(n-3) \quad (14)$$

where $i(n)$ and $q(n)$ for $1 \le n \le 32767$ represent binary-valued 0 or 1, and the additions are module-2. A high-level description of the steps performed by the IVAA to compute the I and Q phases is listed in Table 5. After initialization, a random population of chromosomes of size N is generated and used to estimated the output sequences of the LFSR containing the I and Q pilot sequences. Then the GA is applied using selection, crossover, and mutation methods.

Table 5. High-level View of the adaptive algorithm

Step	Task Description
1	Initialize the LFSR and GA
2	Generate a population of chromosomes of size N
3	Calculate I and Q pilot PN sequences
4	Calculate the fitness of each chromosomes
5	Perform selection process
6	Apply crossover using crossover probability (Pc).
7	Apply mutation using mutation probability (Pm)
8	Place the resulting chromosomes in the new population
9	Replace the current chromosome population with the new population
10	Repeat steps 3-9 for each PN chip until the 100% of the population converges to the IVs including the SOM bit.

4.3.1 Initialization Procedure

Initialization starts both the LFSR[1] and the GA algorithm. The initialization parameters used in the algorithm are: ***m, N, Clks, Pc, Pm,*** where:

m identifies the number of shift register (SR) cells (15).

N identifies initial values chromosome population size.

[1] Initialization of the LFSR is needed for only simulation test purposes.

Clks identifies the number of sequences that the LFSR uses.

Pc identifies crossover probability.

Pm identifies mutation probability

4.3.2 Generating Initial Population

In this practical application, we take in to the consideration the size of the initial population. We need to have an optimized population size that can converge properly with minimum computational complexity. As discussed in [44], a small population size causes the GA to quickly converge on local minimum, because it insufficiently samples the parameter space, and large population size takes too long to find the optimum solution. In our algorithm, we decided to find the correct population size by evaluating both computational complexity and convergence concerns.

4.3.3 In-phase and Q-phase Sequence computation

The IS-95 and IS-2000 standards define PFCs of the 15-stage LFSR used to generate the CDMA forward Pilot PN sequences. Since the PFCs of the LFSRs are known, the IVAA uses randomly generated initial value to compute the I and Q sequences. The IVAA loads the IVs on the LFSRs and computes the output bit using the PFCs. The feedback bit in the ML-LFSR follows the following formula:

$$a_n = \sum_{i=1}^{15} c_i a_{n-i} = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_{15} a_{n-15} \tag{15}$$

4.3.4 Selection Process

Chromosomes are selected through natural selection or survival of the fittest process. In this process, the rank-based fitness assignment method is used, where the fitness value that is assigned to each individual depends on its matching to the value of the I and Q output sequence bit of the LFSR. At the first generation of the process, natural selection method known as the *thresholding* [44] is applied to the initial population. In this approach, all chromosomes that have a lower than defined fitness value are discarded. The thresholding allows the algorithm to get a very good start at the first generation. After initial fitness assignment, the next PN chips use fitness assignment that proceeds after the initial assignment. A short outline of the procedure is shown below:

if computation is true

*fitness = 2 * previous fitness*

else

*fitness = 1/2 * previous fitness*

end

end

The selection process is performed using Tournament selection method. Tournament size equal to the size of population function is used in the Tournament selection. Probability of crossover (Pc) equal to 0.70 is used during the recombination process. Probability of mutation (Pm) equal to 0.001 is used. These values are generally used in the literature [61].

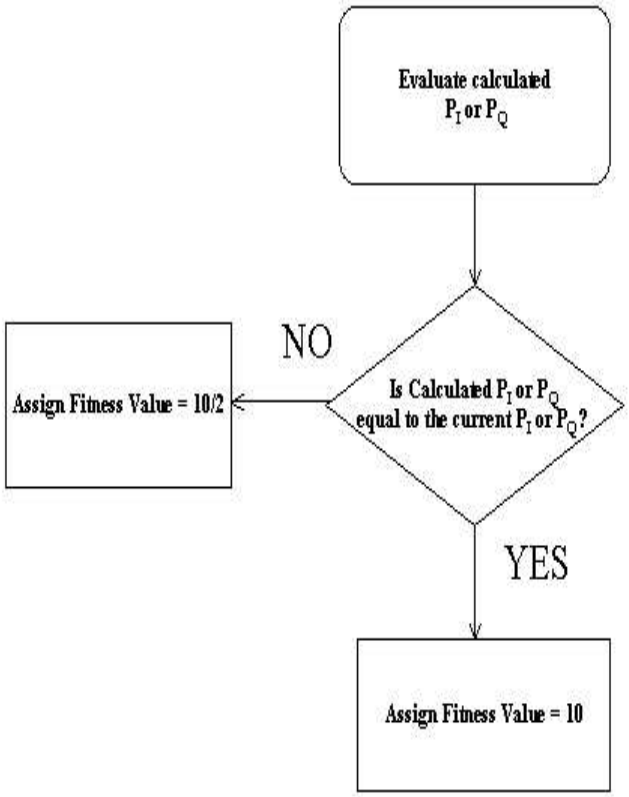

Figure 19. Initialization Fitness Assignment Procedure

4.4 Performance Test Results

To evaluate the performance of the IVAA for obtaining I and Q pilot PN sequences, simulations tests were performed in channel conditions with noise and without noise. Figure 20 shows a high level outline of the performance test approach. The performance of the IVAA is measured based on how fast the algorithm finds the I and Q PN sequences (in PN chips), and how accurate it performs. The PFCs of the LFSR defined in IS95A [78] and IS2000 [1] were tested with different initial values. In each test case, the performance result is shown by plotting the computation accuracy (%) and computation time. Average population fitness results are also shown to verify the performance of the GA.

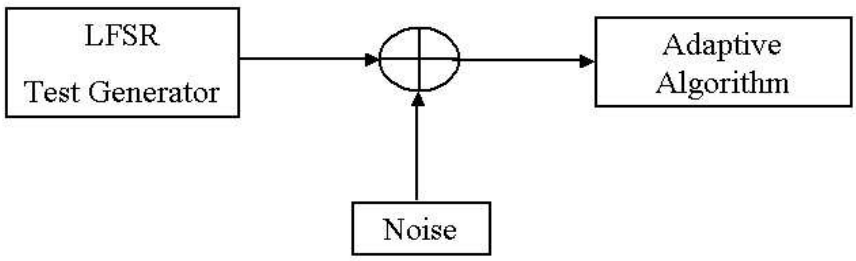

Figure 20. Performance Test Outline

In IS-2000 and IS-95, the forward CDMA channel consists of the following code channels: The pilot channel, up to one sync channel, up to seven paging channels, and a number of traffic channels. To provide orthogonal channelization among all code channels on a given forward CDMA Channel, each of these code channels is orthogonally spread by the appropriate Walsh function and is then spread by a quadrature pair of Pilot PN sequences at a fixed chip rate of 1.2288 Mcps. One of sixty-four time orthogonal Walsh functions available is assigned to the code channel number [1][27].

The forward Traffic Channel is used for the transmission of user and signaling information to a specific MS during a call. All the traffic and overhead (Pilot, Paging, and Sync) signals transmitted from a BS in a particular CDMA channel share a common PN code phase. These codes are distinguished at the MS receiver by using a binary orthogonal code based on Walsh functions [79][80] (Hadamard matrices). The Walsh function is 64 PN code chips long and represents 64 different orthogonal codes. The orthogonality provides nearly perfect isolation between the multiple signals transmitted by the base station. Ideally, the same BS users are orthogonalized, and they do not appear as interference to one another. Therefore, in our test simulations, we assumed that all of the forward channels are orthogonal.

In the conventional analog communication channels, one of the main system performance technique has been the signal to noise ration (SNR) [81]. The SNR easily indicate the over

all performance of an analog channel by describing how strong the information is, relative to the noise that also is present in the channel. In the typical environment of advanced mobile phone systems (AMPS) analog cellular wireless channel, an ideal SNR (also known as the carrier-to-interference (CIR)) level of 17dB[1] or greater is required to control co-channel interference [66]. Deterioration in audio quality occurs when the SNR level drops below 17dB level [41]. Figure 21 shows the appearance of signal and noise level.

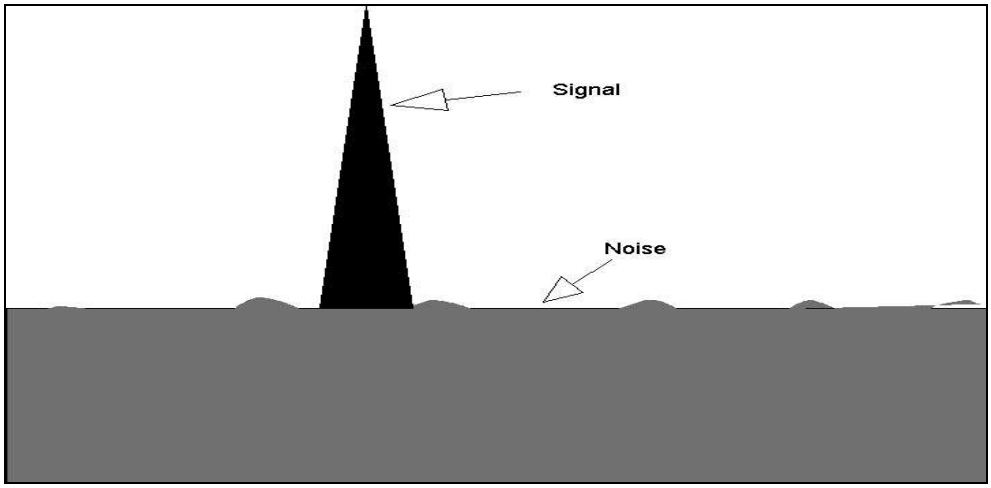

Figure 21. Signal and Noise Level

We can define the SNR as follows:

[1] 17dB level is an ideal level.

$$SNR = \frac{Signal(mW)}{Noise(mW)}$$

$$= [(10 * Log_{10}(S)) - (10 * Log_{10}(N))] \text{ (dBm)} \quad (16)$$

In the digital CDMA communication systems, the channel SNR is characterized as a bit energy over wideband noise spectrum or $\frac{E_b}{N_o}$. E_b is the bit energy corresponding to the amount of energy that each bit is transmitted with, and N_o is the amount of noise that contains in the channel. The $\frac{E_b}{N_o}$ in the CDMA is used as the SNR in the analog channel. Hence, we can define the $\frac{E_b}{N_o}$ measured at the mobile antenna as follows [27][80][81]:

$$\frac{E_b}{N_o} = SNR \quad (17)$$

$$\frac{E_b}{N_o} = \frac{S}{N_o} P_G \quad (18)$$

where:

S is the CDMA power received at the mobile antenna

N_o is the total noise spectrum in the CDMA channel

P_G is the Processing (spreading) Gain for the CDMA channel.

The P_G is defined as the $\frac{R}{W}$ where R represents the CDMA channel transmission rate and W is the CDMA channel spreading rate. In our simulation, we used spreading rate of

1228800 chips (spreading rate 1) with a data transmission rate equal to 9600 bps. Therefore, our P_G is:

$$P_G = \frac{1228800}{9600} = 128 \approx 21.072 \text{ dB}.$$

and

$$P_G = \frac{1228800}{14400} = 85 \approx 19.072 \text{ dB}.$$

The bit error rate (BER) or the bit error probability (P_b) is the probability that a bit sent over the link will be received incorrectly. BER is a very important system quality indicator in the digital communications. BER does not depend on the speed of a digital transmission [63].

To establish the BER in our test simulation, we used the widely accepted values [27] of the BER. The mean BER values are 8.9×10^{-5} for 9600bps and 5.7×10^{-5} for 14400bps vocoder. From the BER we determine the required $\frac{E_b}{N_o}$. To be meaningful, the BER in our simulation tests used an integration period of 1×10^7 bits.

We found that IVAA consistently achieves 100% prediction accuracy on the 15th PN chip. Figure 22 shows a plot of the adaptive algorithm estimation accuracy performance results in obtaining the I pilot PN sequences without any BER in the channel. Figure 23 shows the

results when the BER of 0.01 was used. Figure 24 shows a plot of the I PN sequence performance accuracy for both non-noisy and noisy conditions. Figure 25 shows comparison of the Q PN sequence performance with and without the presence of noise. The average fitness values for the Q PN sequences are shown in Figure 26 for both noisy and no noisy channel conditions.

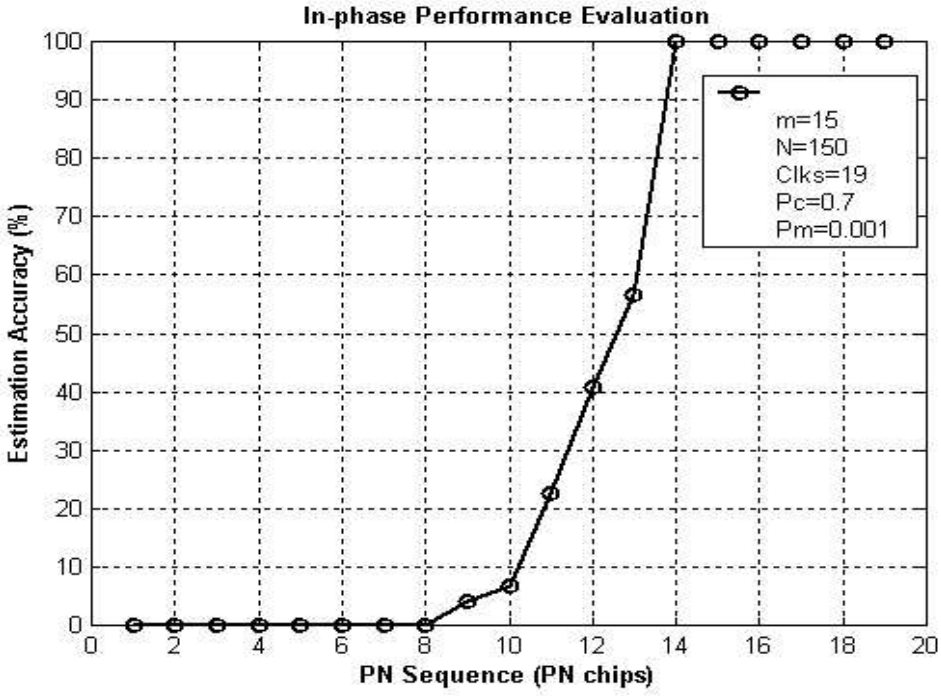

Figure 22. In-phase Performance Evaluation without noise

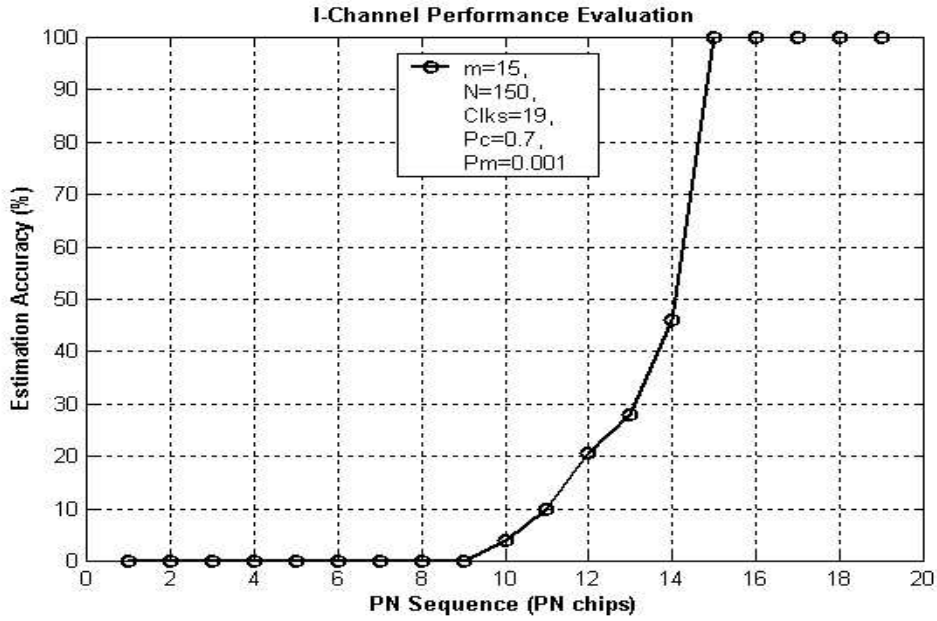

Figure 23. In-Phase performance evaluation with noise

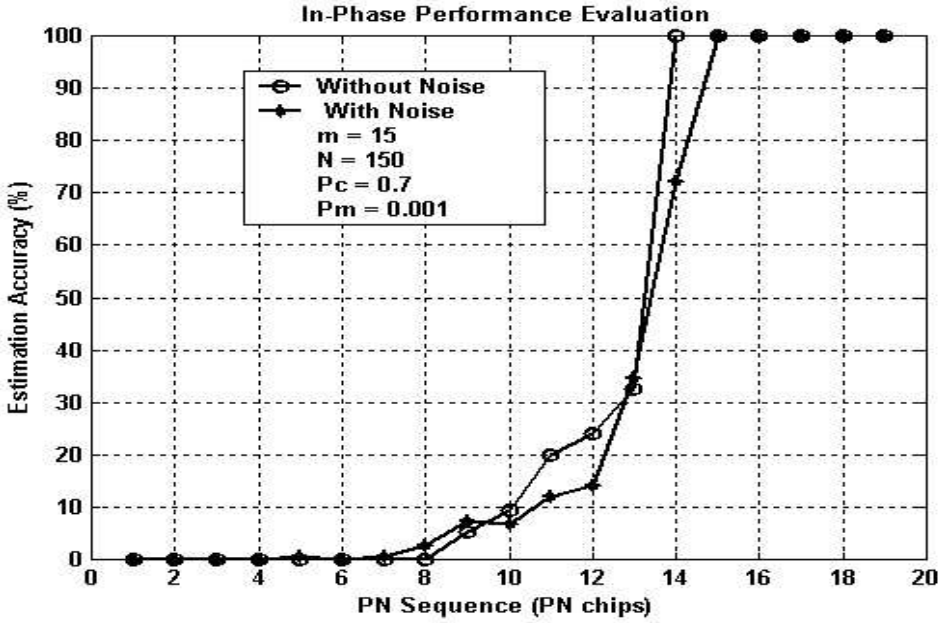

Figure 24. In-phase performance evaluation with and without noise.

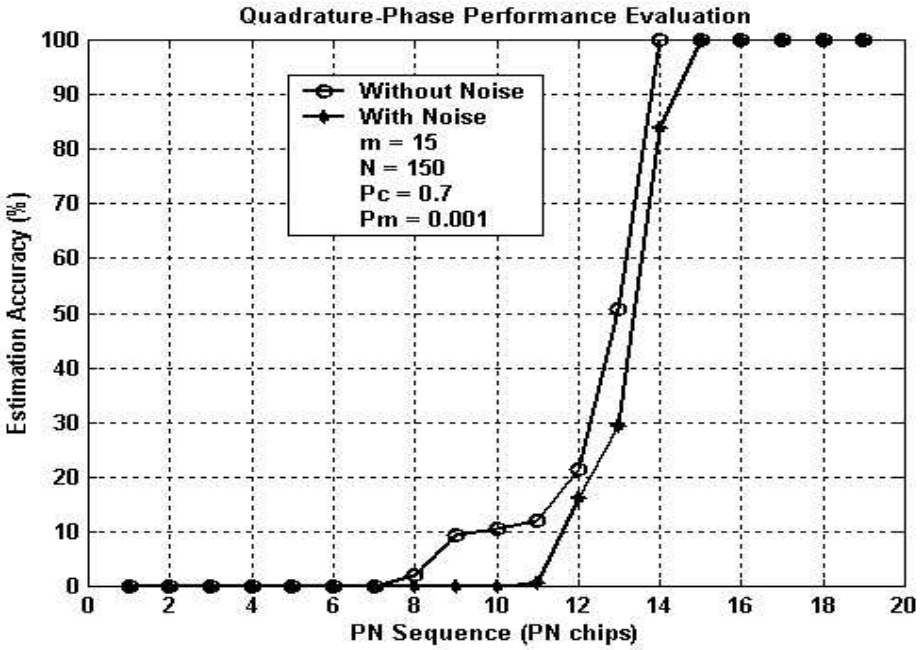

Figure 25. Q-phase performance evaluation with and without noise

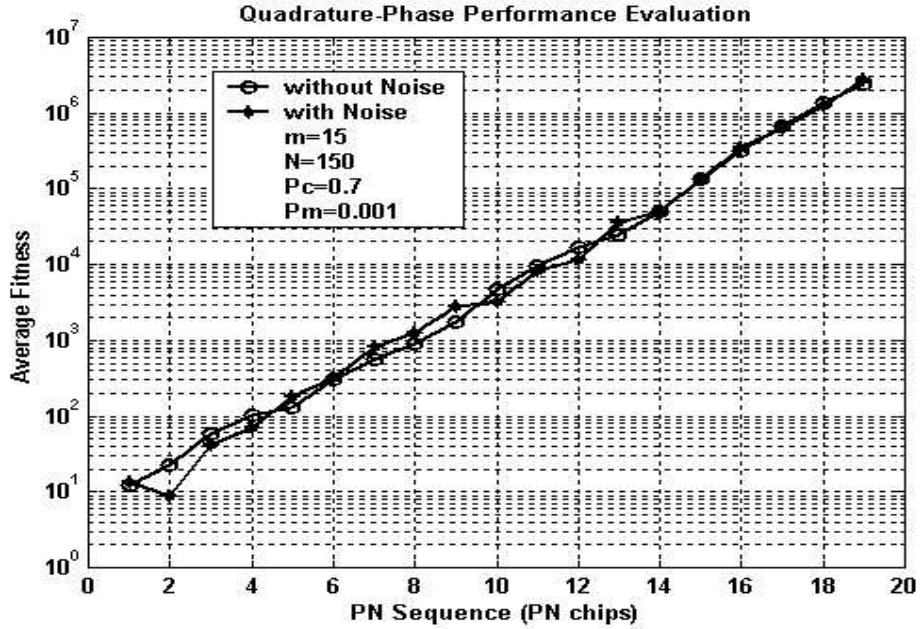

Figure 26. In-Phase Average Fitness Performance

4.5 Computational Complexity

When the IVAA is obtaining the pilot PN sequence, it is required to work in real time. Therefore, it is important that we consider the amount of computation required to obtain the PN sequences correctly. We can characterize the computational requirements of the IVAA as the computational complexity for its implementation. The computational complexities of numerical algorithms are expressed in units of "flops". Flops are roughly the numbers of multiply-and-accumulate operations required for execution. The divides and square roots are considered equal to multiply-and-adds. Also, the multiplies without adds are rounded-up to multiply-and-adds [35].

The computational complexity of the IVAA is tested using Intel Pentium III microprocessor, with Clock = 1GHz CPU speed Peak = 1000 Mfbps[1]. Knowing the size of ML LFSR, we are interested in computing the run time. To evaluate the computational complexity of the adaptive algorithm, we run test simulations for the algebraic and the IVAA using same test plot-form. The test results in Figure 27 shows that the IVAA does not increase the computational complexity when compared to the currently used algebraic methods. In fact, the computation time decreased in the IVAA comparing to the algebraic approach by about 5 seconds. This is because the IVAA finds the I and Q in a shorter time

---

[1] 1 flop = 1 floating point operation (additions, substructions, multiplication) per second.

while the algebraic computations require at least one cycle of the LFSR sequences to get the start of message (SOM) bit of the PN sequences.

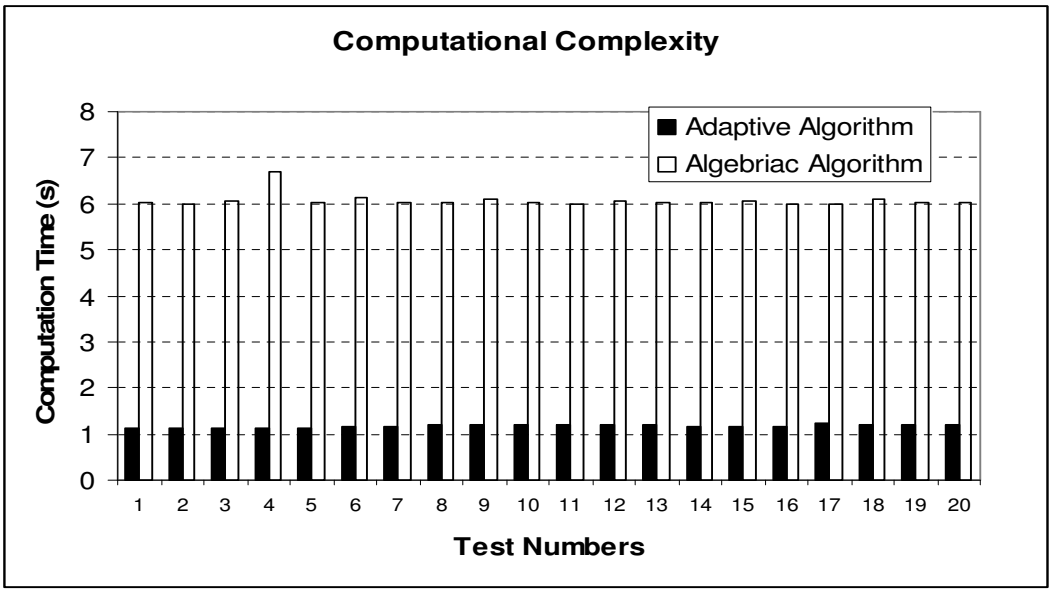

Figure 27. Computational Run-time Performances

Chapter 5

5 Computing PFC of LFSR

This chapter considers in the situations that the initial values of the ML-LFSR are known, and it presents a PFC adaptive algorithm (PFCAA) that allows computing the PFC. First, the shift registers are initialized randomly with at least one register having non zero value. Then GA is applied using selection, crossover, and mutation methods. Steps that the PFCAA follows are shown in the flow chart in Figure 28. The description of the process that the PFCAA follows is described below:

1. Restore received data a_R
2. Perform initialization
3. Generation PFC initial population
4. Calculate output data a_C
5. Assignment fitness value
6. Perform selection process
    a. Tournament selection
7. Perform recombination (Crossover) process
    a. Probability of crossover, Pc = 0.70
8. Perform mutation process

a. Probability of mutation, Pm = 0.001

9. Save the new population
10. Continue steps 4 to 9 until the correct PFC found

Note the steps that the PFCAA follows are similar to the steps shown for the IVAA in Table 2. The difference is in the PFCAA the IVs are known and the algorithm finds the PFCs.

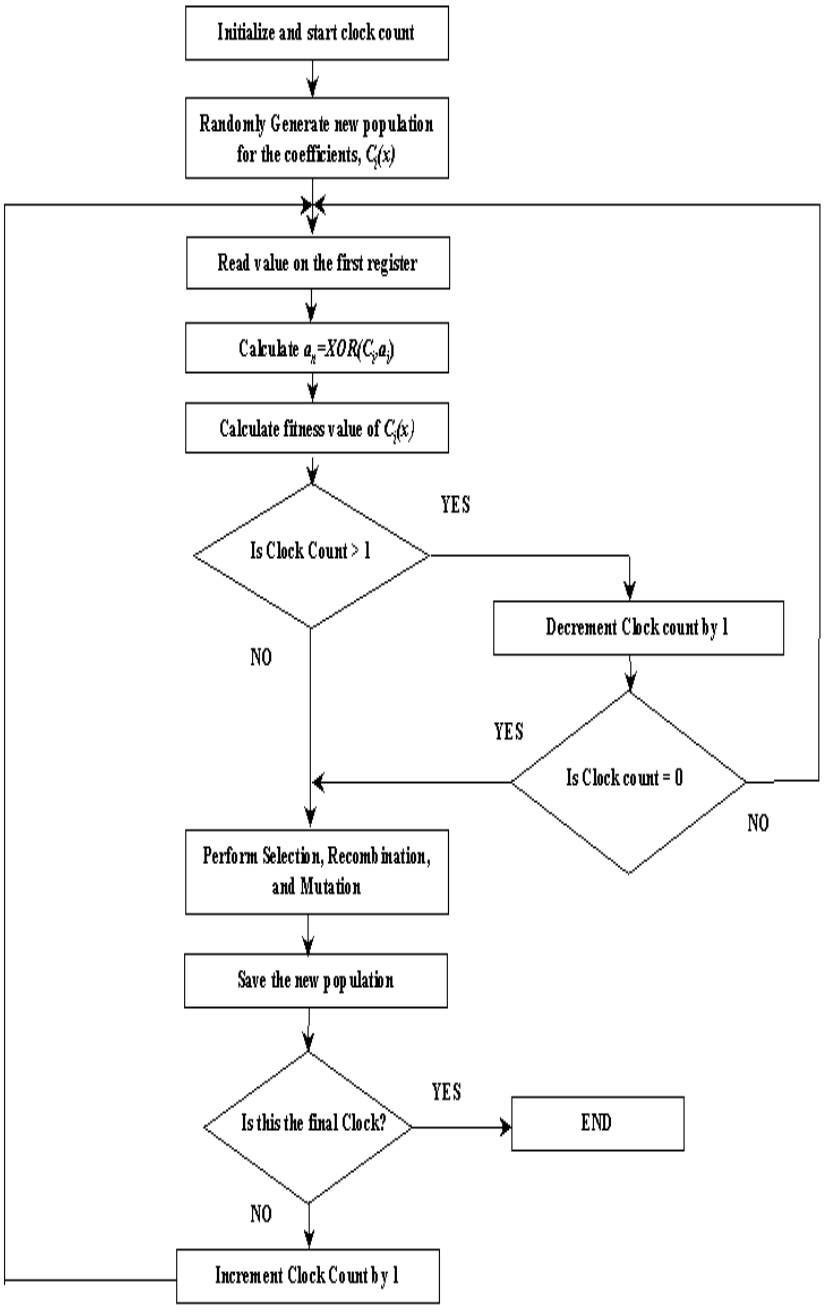

Figure 28. Flowchart of the PFCAA

5.1 Initialization

The initialization starts both the LFSR and the GA algorithm. The values of the ***m*** defined shift registers are initialized with non-null value on at least one shift register. In initialization process, the following parameters are initialized: **m, N, Clks, Pc, Pm,** where:

m = number of shift register (SR) cells.

N = Polynomial feedback coefficient population size

Clks = Number of clocks that the LFSR will be clocked

Pc = crossover probability

Pm = mutation probability

5.2 Generating Initial Population

After initialization, the value of the received data a_R will be restored. The GA algorithm generates initial population for the PFC ($\boldsymbol{C_i}$) using initial parameters defined in section 3.1. An output data a_C for $\boldsymbol{C_i}$ will be calculated using the initial values on the shift registers and the randomly generated population of the feedback coefficients. The value of the a_C is calculated by performing XOR logical function and shift operation on coefficient population In each clock sequence, the value of the received data a_R is restored and that value is used for computing the output data a_C .

5.3 Fitness Value Assignment

The fitness value is assigned to all individuals in the initial population based on how the computed a_C data and received a_R data match. The rank-based fitness assignment method is used, where the fitness value that is assigned to each individual depends on its matching to the value on a_R. At the first initialization, if the computing value on a_C is true, a fitness value equal to 10 is assigned to the individual. If it is not true, a fitness value equal to 10/2 is assigned. In the following clocks, the fitness values of the preceding are used as a base fitness. Whenever the computation becomes true, 2 multiply the previous fitness value. When the computation is not true 2 divides previous fitness value.

5.4 Selection, Recombination, and Mutation Process

Selection process is performed using Roulette Wheel, Universal sampling, Tournament selection and Truncation selection methods of the GA. The Tournament selection performed best in the prediction accuracy and therefore was chosen for this model. Probability of crossover *(Pc)* equal to 0.70 and a probability of mutation *(Pm)* equal to 0.001 are used for the recombination and mutation processes. These values are common values used in GA [61].

5.5 Performance Tests Results

This section presents the simulation test results for the PFCAA. To show the performance of the algorithm, we will show results from six different test cases of LFSR with different

IVs. The performance of the PFCAA is measured in terms of the computation accuracy and time (clock sequence shifts that it takes the model to achieve 100% accuracy). The list of the test cases are provided in Table 6:

Table 6. PFCAA Performance Test Configurations

Test Case	**f(x)**	**m**	**N**	**Pc**	**Pm**
1	$f(x) = 1 + x^2 + x^3$	3	30	0.70	0.001
2	$f(x) = 1 + x^3 + x^4 + x^5$	5	50	0.70	0.001
3	$f(x) = 1 + x^3 + x^6 + x^8 + x^9 + x^{10}$	10	100	0.70	0.001
4	$f(x) = 1 + x^5 + x^7 + x^8 + x^9 + x^{13} + x^{15}$	15	100	0.70	0.001
5	$f(x) = 1 + x^5 + x^7 + x^8 + x^9 + x^{13} + x^{15} + x^{18} + x^{20}$	20	150	0.70	0.001
6	$f(x) = 1 + x^5 + x^7 + x^6 + x^9 + x^{15} + x^{17} + x^{21} + x^{23}$	25	150	0.70	0.001

Test case 1:

$f(x) = 1 + x^2 + x^3$ with m=3, N=30

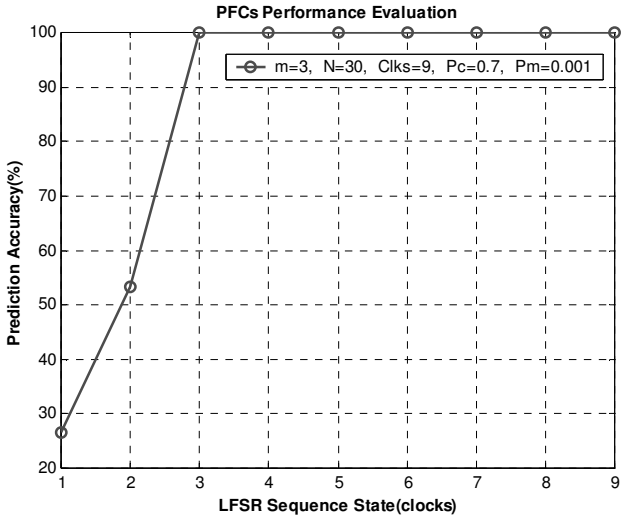

Figure 29. PFC Accuracy Performance with m=3 and N=30

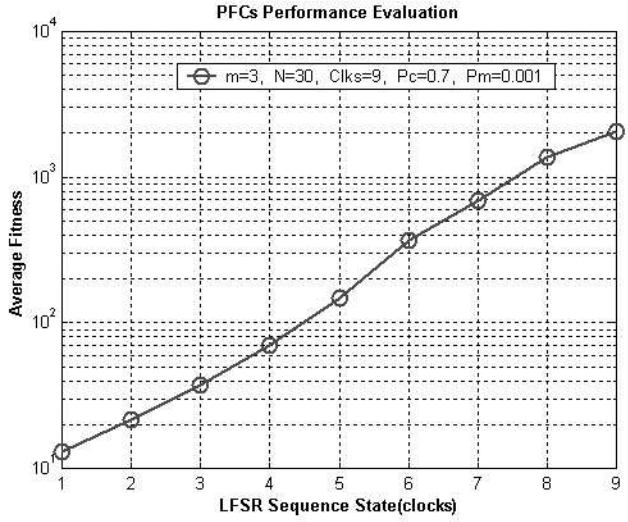

Figure 30. PFC Fitness with m=3 and N=30.

Test case 2

$f(x) = 1 + x^3 + x^4 + x^5$ with m=5, N=50

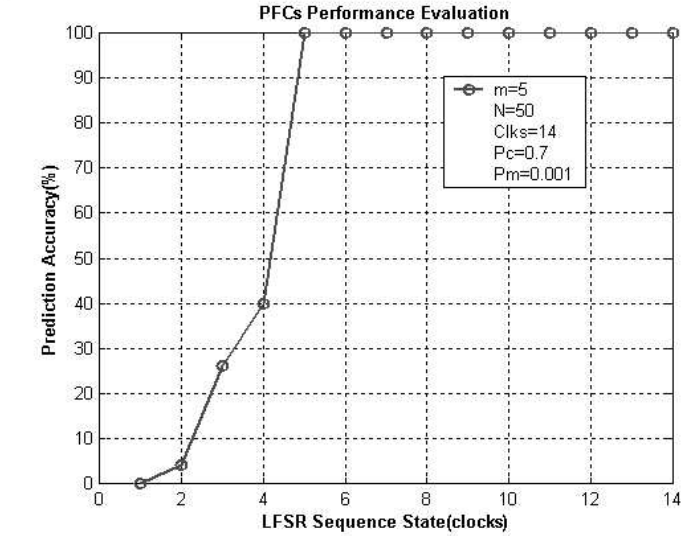

Figure 31. PFC Accuracy Performance with m=5 and N=50

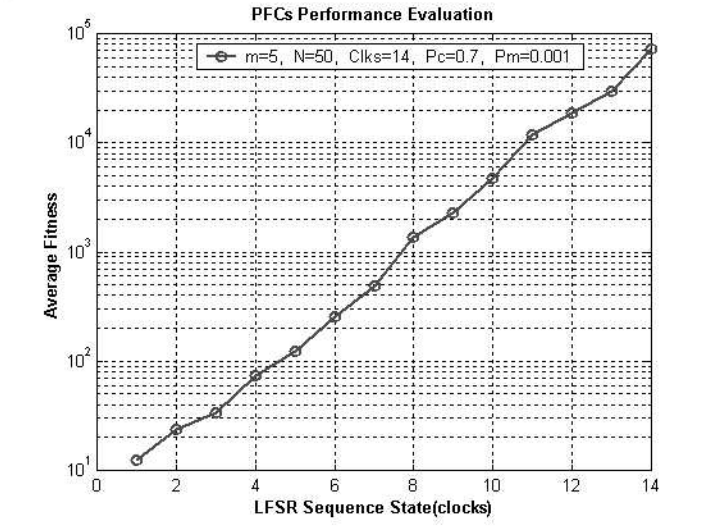

Figure 32. PFC Fitness with m=5 and N=50

Test case 3

$f(x) = 1 + x^3 + x^6 + x^8 + x^9 + x^{10}$ with m=10, N=100

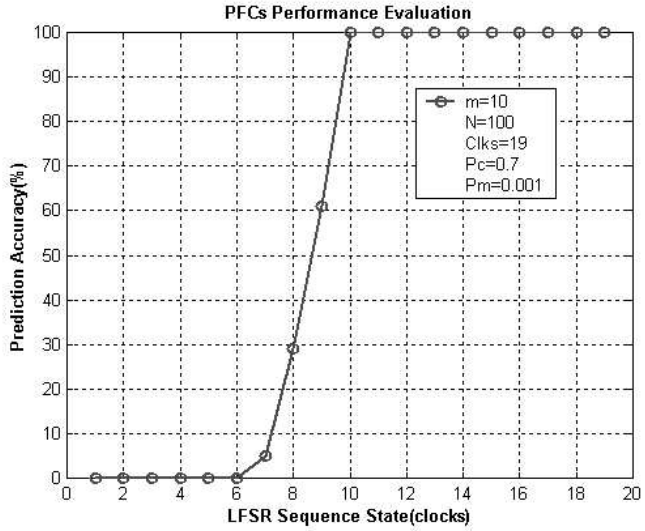

Figure 33. PFC Performance with m=10

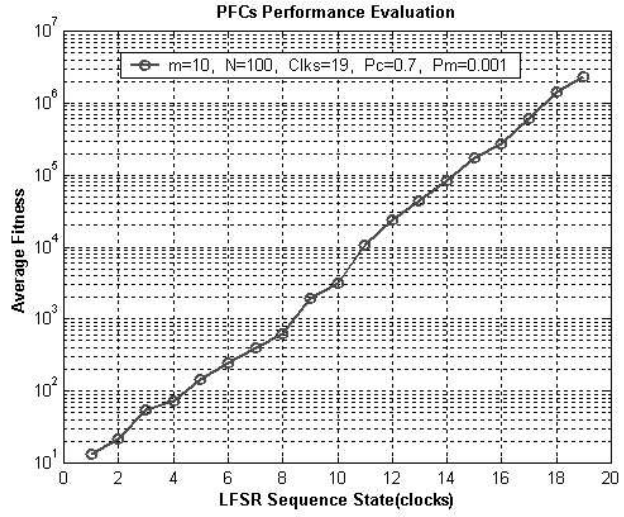

Figure 34. PFC Fitness with m=10 and N=100

Test case 4

$f(x) = 1 + x^5 + x^7 + x^8 + x^9 + x^{13} + x^{15}$ with m=15, N=100

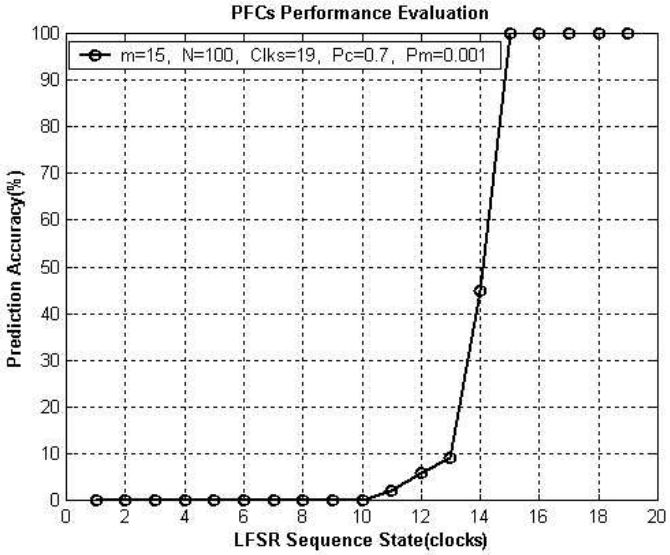

Figure 35. PFC Performance with m=15 and N=100

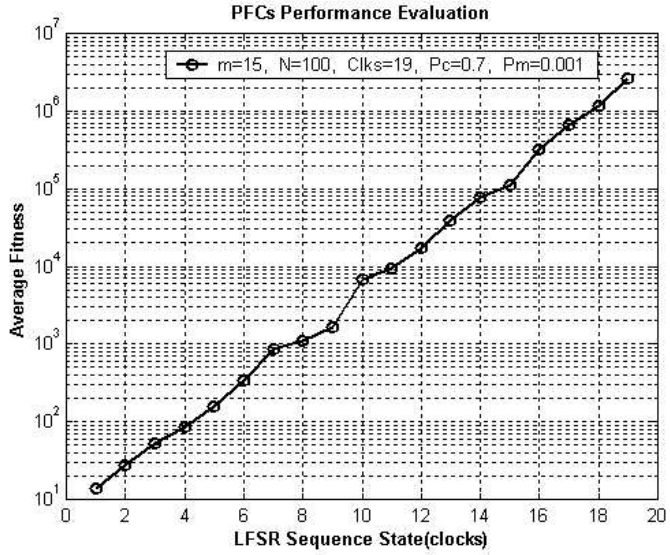

Figure 36. PFC Fitness with m=15 and N=100

Test case 5

$f(x) = 1 + x^5 + x^7 + x^8 + x^9 + x^{13} + x^{15} + x^{18} + x^{20}$ with m=20, N=150

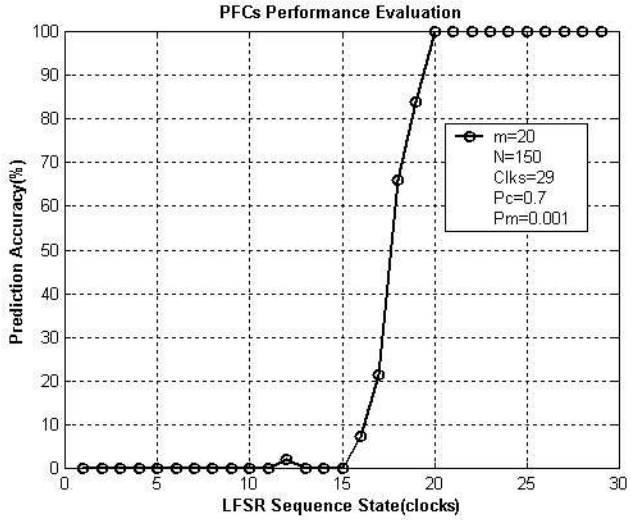

Figure 37. PFC Performance with m=20 and N=150.

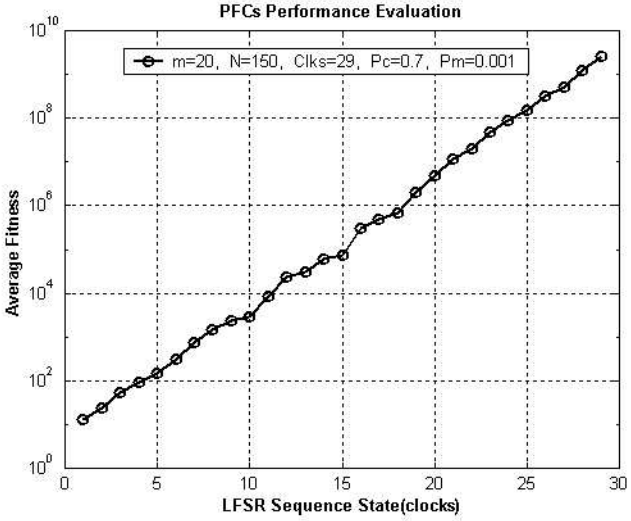

Figure 38. PFC Fitness with m=20 and N=150.

Test case 6

$f(x) = 1 + x^5 + x^7 + x^6 + x^9 + x^{15} + x^{17} + x^{21} + x^{23}$ with m=25 and N=150

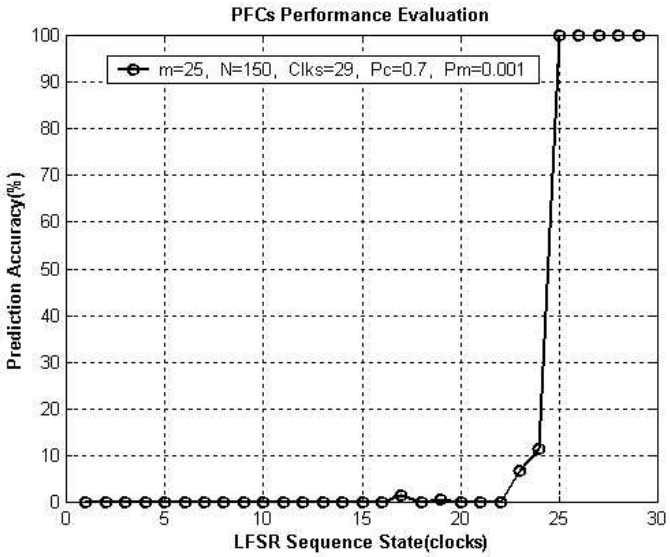

Figure 39. PFC Performance with m=25 and N=150.

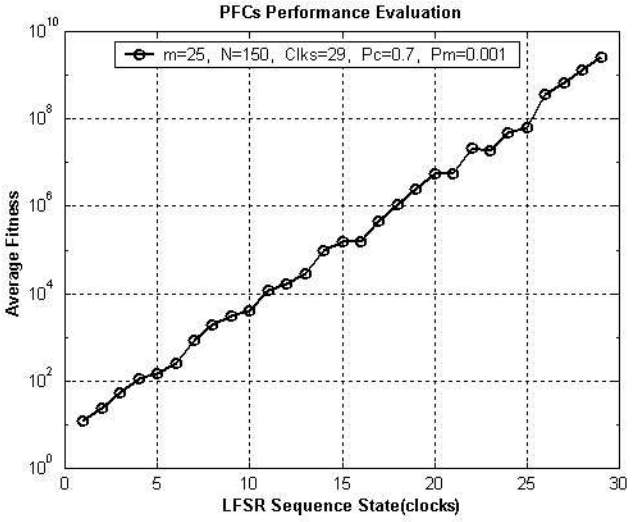

Figure 40. PFC Fitness with m=20 and N=150.

Chapter 6

6 Computing IV and PFC

Initial Value and Polynomial feedback coefficients adaptive algorithm (IVPFCAA) computes both the IV and the PFCs of ML LFSR at the same time. The algorithm reads the received output bits of the LFSR and computes the received sequence bits without knowing either IV or PFCs using Genetic algorithm [30] synthesis. The objective of the algorithm is to find both the initial value and the PFCs of the LFSR generated sequences without waiting for the complete cycle period. As described in section 2.1, the period of the ML LFSR repeats after 2^m-1 period (where m stands for the number of shift registers). First, the IVPFCAA reads and stores the output data generated by the LFSR. Then the IVPFCAA initializes its initialization parameters. The GA is applied using selection, crossover, and mutation methods. The description of the process that the IV-PFCAA is shown below:

1) Restore received data $[a_R]$
2) Perform Initialization
3) Generate initial population for IV and PFC
4) Calculate $[a_C]$
5) Assign fitness value
6) Perform selection process

   a. Tournament Selection

7) Perform recombination (crossover) process
   a. Probability of crossover, Pc = 0.70
8) Perform mutation process
   a. Probability of mutation, Pm = 0.001
9) Save the new population
10) Repeat steps 4 to 9 until 100% accuracy achieved

Note the steps that the PFC-IVAA follows are similar to the one for the PFCAA shown in the flow chart in Figure 28. The difference is in the PFC-IVAA, both IV and PFCs are unknown and the algorithm finds both.

6.1 Initialization

The initialization starts both the LFSR and the GA algorithm. The values of the ***m*** defined shift registers are initialized with non-null value on at least one shift register. In initialization process, the following parameters are initialized: **m, N, Clks, Pc, Pm,** where:

m = number of shift register (SR) cells.

N = Initial value population size

Clks = Number of clocks that the LFSR will be clocked

Pc = crossover probability

Pm = mutation probability

6.2 Initial Value Generation

After initializing the shift registers, the received data a_R is restored. The GA generates initial population for the initial value ($\boldsymbol{IV_i}$) and PFC ($\mathbf{C_i}$) using initial parameters defined in section 6.1. The a_C is calculated from the $\boldsymbol{IV_i}$ and $\mathbf{C_i}$ and this estimates the received a_R. In each clock sequence, the received bit from the LFSR is predicted using randomly generated population of $\boldsymbol{IV_i}$ ***and*** $\mathbf{C_i}$.

6.3 Fitness Value Assignment

After the value of a_C is calculated by performing XOR logical function on IV and PFC values, then a fitness value assignment process is performed. The rank-based fitness assignment method [30] was used, where the fitness value that is assigned to each individual depends on its matching to the value on a_R. At the first initialization, if the prediction value on a_R is true, the fitness equal to 10 is assigned to that individual. If the prediction is not true, then a fitness value equal to 10/2 is assigned. In the subsequent clocks, the fitness values of the preceding clocks are used as base fitness. Whenever the prediction becomes true, 2 multiply the previous fitness value. When the prediction is not true, 2 divides previous fitness value. This process ranks all of the individuals by assigning a fitness values based on how close that IV and PFC individual are to produce the a_C value on specific clock.

6.4 Selection, Recombination, and Mutation Process

Selection process is performed using Tournament selection of the GA [30]. Probability of crossover ***(Pc)*** equal to 0.70 was used during the recombination process. These values are values that are generally used in the literature [61]. Probability of mutation ***(Pm)*** equal to 0.001, which is in the commonly used range as described in [61] was also used for this model.

6.5 Identifying Security Threats in Ad Hoc Wireless Network

Ad hoc wireless networks are very vulnerable to security threats, therefore it is critical to monitor and identify any encountered security threats immediately. Once a threat is present on the network, an intrusion Detection System (IDS) tries to detect and alert on attempted intrusions into a network. The IV-PFCAA described in the last sections can be used to exploit IDS provided information for identifying the sources of threats containing probes. These probes could be from an alphabet. Since the LFSR is one of the widely used models of pseudo-noise number generator (PRNG), the characters in that alphabet, may be consider are being generated into the network by a LFSR.

Unlike typical mobile wireless networks, ad hoc networks do not depend on any fixed infrastructure such as base stations or mobile switching centers. Instead, hosts rely on each other to maintain network connectivity. Military tactical operations are one of the main

applications of the ad hoc networks. Security is an important issue for ad hoc network and it can be addressed in two main steps. The first step is the intrusion detection. The intrusion detection is one of the main techniques behind protecting network. The intrusion detection systems (IDS) aim to detect and alert on attempted intrusions into a network. This field has been extensively studied [84][13][51][58]. The second step focuses identifying the threats that has been reported by IDS. Proper identification of the sources of the threat is important and helps designing securer networks in the future. Threats such as the signal jamming require quicker attack identification. IV-PFCAA allows identifying the origin of a threat source that is present on a network. IV-PFCAA assumes that probe of characters from the threat is reported by an IDS, which is designed to monitor, detect, and report any threats present on the network. Figure 41 shows the overview of the IDS and the ad hoc threat identification network components.

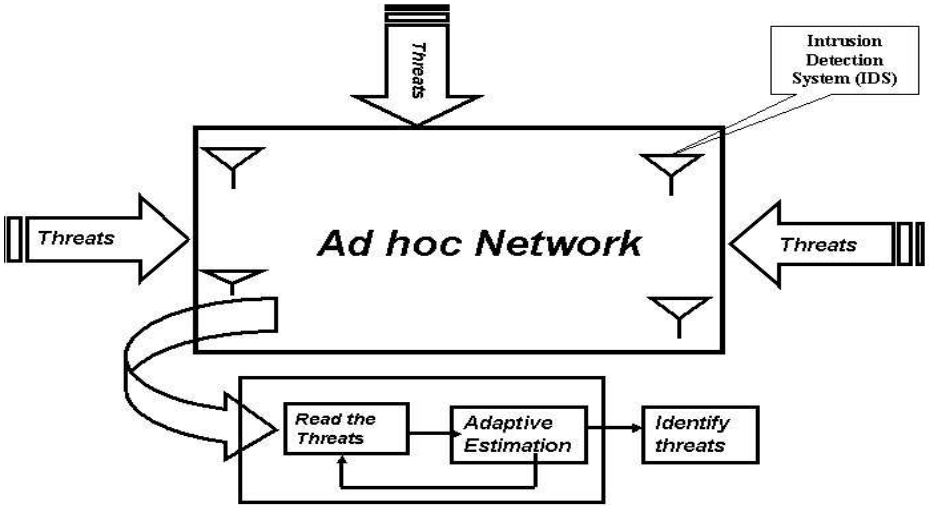

Figure 41. Ad-hoc Threat Identification Approach

As the probes of the threat arrive in the protected ad hoc wireless network, the IDS system reports them to the IV-PFCAA for identification. Then the algorithm reads and attempts to discover the identity of the source generating the threats. The threat probes are represented as a pseudo-noise or pseudo-noise (PN) binary sequences generated by a LFSR. IV-PFCAA is used to identify the threats by finding PFCs and IVs of the LFSR.

GA is successfully utilized in the identification process. The PN sequences generated by the ML LFSR representing the probes of the threats are reported to the adaptive algorithm by IDS. The algorithm first reads the probes and estimates the threats by applying GA methods of fitness assignment, selection, crossover and mutation. The algorithm tries to find the threat sources by estimating the IVs and PFCs of the LFSR. In each clock sequence

of the LFSR, the GA compares the pattern of the received LFSR output sequences to its estimated values. Therefore, the fitness values of each clock sequences generation improves as the clock continuous. After initializing the GA parameters, random population of chromosomes of size N is generated with at least one register having non zero value. Then the GA is applied using selection, crossover, and mutation methods. Figure 42 shows flow diagram of the IVPFCAA. Figure 43 and 45 show performance test results of the IVPFCAA.

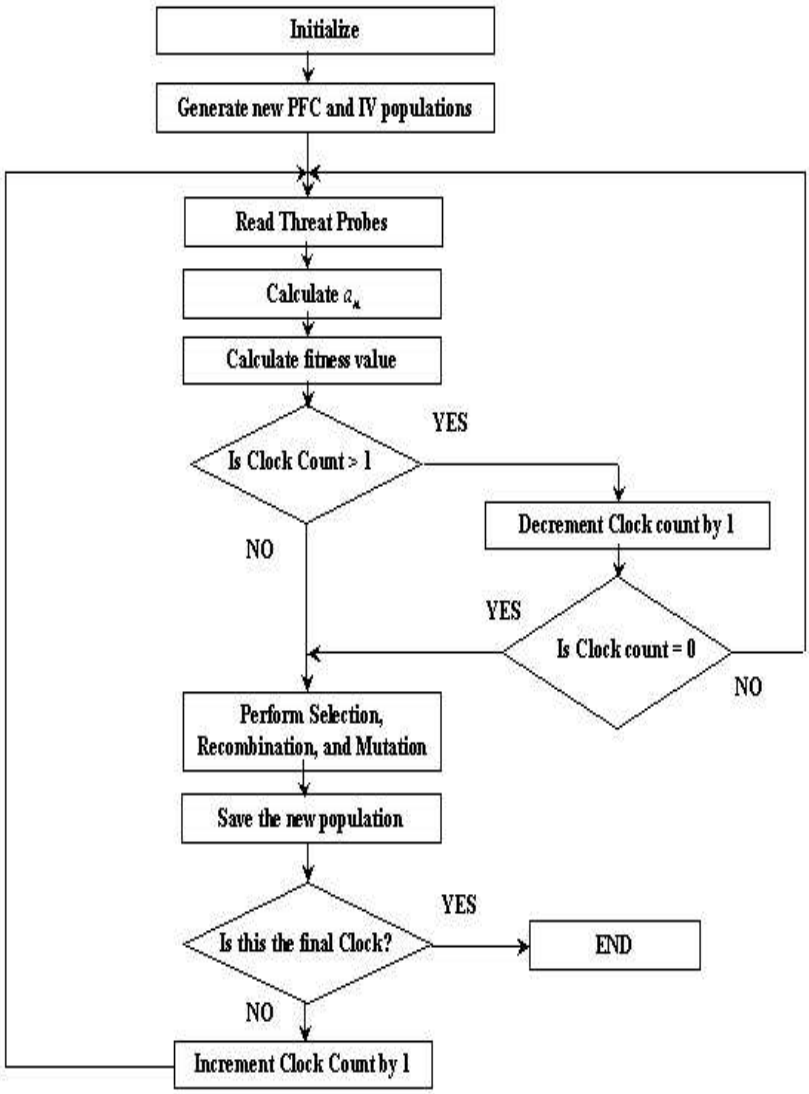

Figure 42. Detail algorithm flow diagram

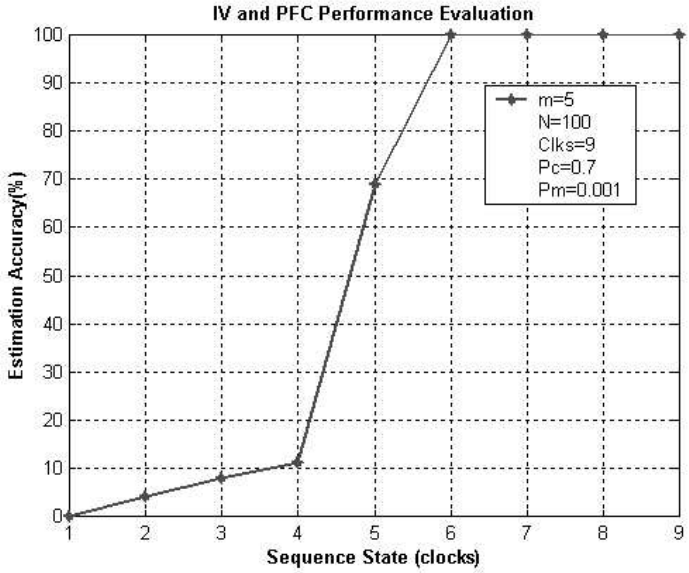

Figure 43. IV and PFC Performance Evaluation

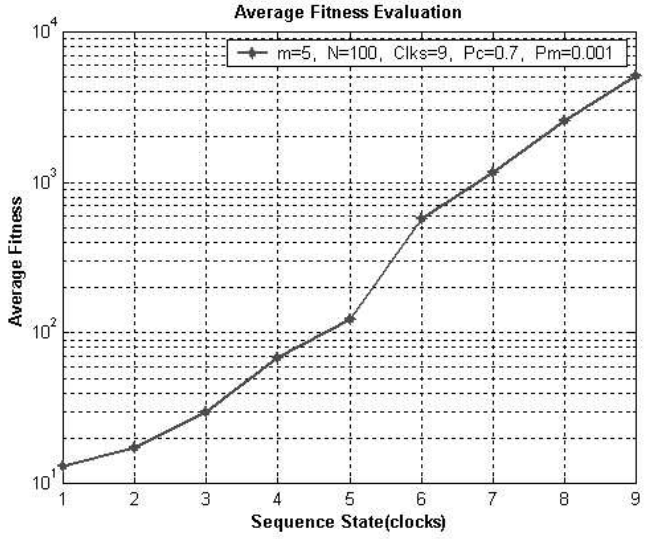

Figure 44. IV and PFC Fitness Evaluation

Chapter 7

7 Conclusion

The linear feedback shift register (LFSR) has been extensively used in many communication systems as a PN generator. Primarily, LFSR is used as a pseudo-noise (PN) generator in the wireless Code Division Multiple Access (CDMA) communication systems, and as crypto-analysis in security systems. The LFSR is used more often comparing to the other pseudo-random generators, this is because the LFSR is easier to implement in hardware. The maximal length LFSR with ***m*** cascaded shift registers has a sequence period N equal to $2^m - 1$. Therefore, the sequence of the LFSR repeats every N period.

Most of the current methods used to find the PN sequences generated by ML LFSR are algebraic based methods. Theses methods require waiting for at least one complete LFSR period to find either the initial value or polynomial feedback coefficients. This causes longer computation time. Furthermore, these methods perform poorly in noisy channel environments.

This dissertation presented an adaptive algorithm that computes efficiently the sequences of the ML LFSR using GA synthesis. First, a ML LFSR with known polynomial feedback coefficients and unknown initial values was considered. The adaptive algorithm computed

LFSR initial values using an optimized GAs and was implemented for the wireless code division multiple access (CDMA) systems applications.

CDMA systems based on the interim standards IS-95 and IS-2000 use a 15-stage LFSR shift with standard defined polynomial feedback coefficients. The forward Pilot channel is spread in quadrature spreading. The quadrature spreading is a spreading sequence of length 2^{15} pseudo-noise (PN) chips. For spreading rate (SR) 1, each PN chip is about 0.813μs. This sequence is called the pilot PN sequence and consist of a pair of PN sequences called the In-phase (**I**) and quadrature-phase (**Q**) sequences generated by a Linear Feedback Shift Register (LFSR). This pair of PN sequences is used to spread the forward and the reverse CDMA Channel. During the forward Pilot and Sync channel processing, the mobile station acquires and synchronizes to the CDMA system while it is in the initialization state. In the current CDMA systems, to obtain **I** and **Q** Pilot PN sequences for time alignment, a zero is inserted in the LFSR sequences after 14 consecutive "0" outputs. In other words, an additional "0" is added at the end of the LFSR period sequences. When the mobile station (MS) power is turned on, the MS synchronizes with the base station (BS) timing using Global Positioning System (GPS) timing reference on the Sync channel and the pilot channel. .

In this dissertation, we proposed an algorithm that reduces the computation time to find the start of message (SOM) bit of the **I** and **Q** Pilot PN sequences. We started by developing

evolutionary computation models to find the initial values (IV) and polynomial feedback coefficients (PFC) for the maximal-length LFSR that is used to generate the CDMA PN sequences. Next, we developed an adaptive algorithm for computing the **I** and **Q** pilot PN sequences in the CDMA systems that can result in a shorter synchronization time between the base station and mobile station. Simulation results presented showed that the adaptive algorithm performed well in both clean and noisy channel conditions. Simulation test results show that the proposed algorithms cuts the synchronization time by a minimum time of 32753 PN chips without degrading the channel performance, and with less computational complexity than the currently used algebraic methods.

This dissertation also considered a ML LFSR with unknown polynomial feedback coefficients, and the cases where both IV and polynomial feedback coefficients were unknown. In these cases, adaptive algorithms for computing the polynomial feedback coefficients and initial values were presented.

Simulation test results presented show that the evolutionary computations such as the GA have great potential in the area of LFSR sequence computation. Test results show that the GA adaptive algorithms accurately computed the LFSR initial values and polynomial feedback coefficients in a relatively shorter LFSR sequence period.

Future work from this thesis may include studying the implementation of the adaptive algorithms in reverse link CDMA applications.

Appendix A

In this appendix, an example showing the process of the IVAA for computing the initial values of a ML LFSR is presented. This example considers the deterministic ML LFSR in Figure 45 to be a test sequence generator. In each clock time, the register shifts all contents to the right. The sequence a_n propagates through with each term generated linearly from the proceeding terms according to the following formula:

$$a_n = \sum_{i=1}^{3} C_i a_{i-1} = c_1 a_{n-1} + c_2 a_{n-2} + c_3 a_{n-3} \tag{19}$$

Here, all terms are binary (0 or 1), C_1 to C_3, are connection variables (1 for connection and 0 for no connection). The addition is modulo-2 (or "exclusive-OR"). Table 7 shows the linear binary sequence that is generated by this ML LFSR with initial register contents of 1,0,0 and connection variables of 1,0,1.

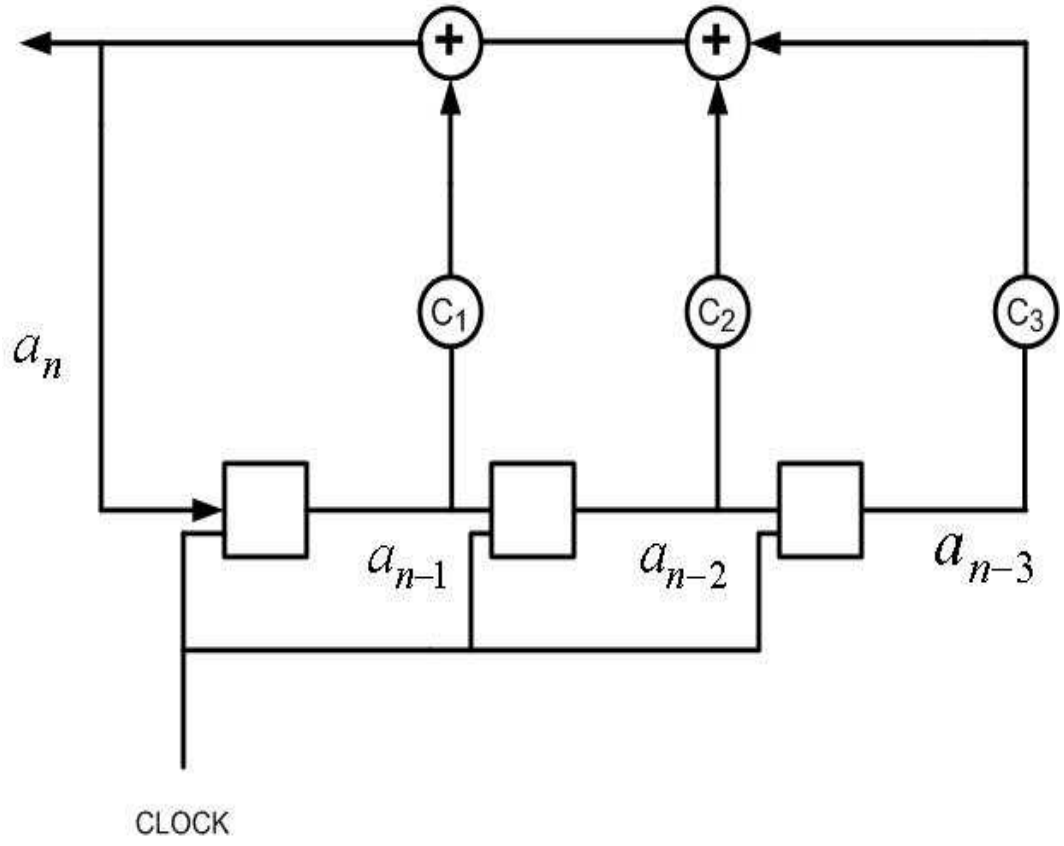

Figure 45. A Three Stage ML LFSR.

Table 7. Binary Data Sequence for a three Stage ML LFSR

Clock	SR Values
	1 0 0
1	1 1 0
2	1 1 1
3	0 1 1
4	1 0 1
5	0 1 0
6	0 0 1
7	1 0 0

Definitions of the terms used in the computation are described below:

m is the number of flip-flops. In this example, we have 3 flip-flops m_1, m_2, and m_3.

C is the connection variables. In this example the connection, variables are c1, c2, and c3.

N is the LFSR sequence length. For this example, $N = 2^3 - 1 = 7$.

nind is the population size of that the GA uses. In this example, *nind = 30*.

Pc is probability of the crossover process.

Pm is probability of the mutation process.

Initial_fitness is the initial fitness value.

Clock is the clock sequence triggered to the ML LFSR for XORing and shifting the contents of the registers to the right

Shift represents the sub-clocks within a clock sequence. The GA uses the sub-clocks to compute output bit.

$[IV]_i(x)$ is the initial value (IV) contents of the SR at i th clock at x shift.

P_{IV} = initial value (IV) population

$[a_n]_{\mathrm{Re}ceived} = [a_1, a_2, a_3, \cdots, a_k]$ is the received binary bits generated from ML LFSR tester.

$[a_n]_{Calculated}$ is the a_n computed from P_{IV} and known LFSR PFC.

$Popfitness_i(x)$ is the population fitness value for i th clock at x shift

$selected\_pop_i(x)$ is the final selected population for i th clock at shift x.

$selected\_pop_i(x, y)$ is the final selected population for i th clock at shift x and y.

$selected\_ind_i(x)$ is the selected individual number for i th clock at shift x .

$selected\_ind_i(x, y)$ is selected individual number for i th clock at shift x and shift y .

$\left[(a_n)_{calculated}\right]_i(x) = [a_n]_{Calculated}$ for i th clock at shift x .

$\left[(a_n)_{calculated}\right]_i(x, y) = [a_n]_{Calculated}$ for i th clock at shift x and shift y

Initialization Parameters

m =3

C = 1 0 1

n =7

nind =30

G = 4

Pc= 0.7

Pm = 0.001

Initial_fitness = 10

Clock 1

Read in coming bits from the LFSR test generator and restore it.

$[a_n]_{Received} = [a_1, a_2, a_3, \cdots, a_k]$

$[a_n]_{Received}$ bit = 1

First, initial values population, P_{IV} , of 30 random chromosomes are generated. Table 8 shows that results of the P_{IV} . a_n is calculated using P_{IV} and known polynomial feedback coefficients (C). In this example the PFC are 1, 0, 1:

$$[a_n]_{Calculated} = \sum_{i=1}^{m} C_i a_{i-1}$$

The results of the a_n calculated with other IV contents are shown Table 9.

Fitness value is to each chromosome of P_{IV} based on how each chromosome's $[a_n]_{Calculated}$ matches to the $[a_n]_{Received}$:

If $[a_n]_{Calculated} = [a_n]_{Received}$

assign fitness value = 10

else

assign fitness value = 10/2

end

The fitness assignment for this generation of P_{IV} is shown Table10.

The first generation's average fitness value is 17.

Table 8. P_{IV}

Number	Chromosome
1	1 0 0
2	1 1 0
3	0 1 1
4	1 1 0
5	1 1 1
6	1 0 0
7	0 1 0
8	1 0 0
9	0 1 1
10	1 0 0
11	1 0 0
12	1 0 0
13	1 1 0
14	1 0 0
15	1 1 0
16	1 0 0
17	1 0 0
18	0 1 0
19	0 0 1
20	1 1 0
21	1 0 0
22	1 0 1
23	1 0 0
24	1 1 0
25	1 0 1
26	0 1 0
27	1 1 0
28	0 0 1
29	1 0 0
30	1 0 0

Table 9. $[IV]_1(1)$
1 1 0
1 1 1
1 0 1
1 1 1
0 1 1
1 1 0
0 0 1
1 1 0
1 0 1
1 1 0
1 1 0
1 1 0
1 1 1
1 1 0
1 1 1
1 1 0
1 1 0
0 0 1
1 0 0
1 1 1
1 1 0
0 1 0
1 1 0
1 1 1
0 1 0
0 0 1
1 1 1
1 0 0
1 1 0
1 1 0

Table10 $Popfitness_1(1)$
20
20
20
20
5
20
5
20
20
20
20
20
20
20
20
20
20
5
20
20
20
5
20
20
5
5
20
20
20
20

Tournament selection is performed (Tournament size = 30) using the population fitness. The wining individuals after the Tournament selection for Clock 1 are shown Table 11. In this simple example, 19 out of 30 individuals are selected and the rest are discarded. The selected population is shown in Table 12. The result after the multipoint crossover is shown in Table 13. The result after the mutation is shown in Table 14.

Clock 2, shift 1

Restore next $[a_n]_{Received} = [a_1, a_2, a_3, \cdots, a_k]$

$[a_n]_{Received}$ bit = 1= 1, 1

Calculate a_n and show IV for clock 2, shift 1. Result is shown in Table 15.

Assign Fitness value to each chromosome of P_{IV} based on how each chromosome's $[a_n]_{Calculated}$ matches to the $[a_n]_{Received}$.

If $[a_n]_{Calculated} = [a_n]_{Received}$

assign fitness value = 10

else

assign fitness value = 10/2

The population fitness assignment value for clock 2, shift 1 is shown in Table.

Compute a_n (showing SR contents) and popfitness for the clock 2 shift 2 using contents from the previous clock 2 shift 1. The a_n value for clock 2 after shift 1 and 2 is shown Table. The popfitness is calculate for the clock 2, shift 2 using the following outline:

If $[a_n]_{Calculated} = [a_n]_{\text{Re}ceived}$

Assign fitness value = previous fitness value x 2

Else

Assign fitness value = previous fitness value / 2

end

Table 18 shows the fitness assignment values for both shift 1 and 2 in clock 2:

Average fitness is 19.75. Table 19 shows the individual numbers selected after the tournament selection is performed for clock 2, shift 1 and 2. Table 20 shows the population selected after the tournament selection is performed for clock 2, shift 1 and 2. Table 21 shows the results after the crossover is performed for clock 2, shift 1 and 2. Table 25 shows the results after mutation is performed. Average fitness for clock 1 and 2

Average fitness is 17 and 19.75, respectively.

Table 11. $selected\_ind_1(1)$

24
20
9
17
30
14
1
9
1
30
8
27
16
20
2
15
11
27
17
19
3
11
13
23
29
17
12
11
9
23

Table 12. $selected\_pop_1(1)$

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

Table 13. After crossover for clock 1

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

Table 14. After Mutation for clock 1		
0	0	1
0	1	0
1	1	0
0	0	0
1	0	0
1	0	0
1	1	1
0	1	1
1	0	0
1	0	0
0	1	1
0	1	0
1	1	0
1	0	0
1	0	0
1	1	0
1	1	1
1	1	0
0	0	0
0	0	1
0	0	0
1	1	0
1	0	1
1	0	0
0	0	0
0	1	0
1	0	0
1	1	0
0	0	1
0	1	0

Table 15. $[IV]_2(1)$		
1	0	0
0	0	1
1	1	1
0	0	0
1	1	0
1	1	0
0	1	1
1	0	1
1	1	0
1	1	0
1	0	1
0	0	1
1	1	1
1	1	0
1	1	0
1	1	1
0	1	1
1	1	1
0	0	0
1	0	0
0	0	0
1	1	1
0	1	0
1	1	0
0	0	0
0	0	1
1	1	0
1	1	1
1	0	0
0	0	1

Table 16. $Popfitness_2(1)$
20
5
20
5
20
20
5
20
20
20
20
5
20
20
20
20
5
20
5
20
5
20
5
20
5
5
20
20
20
5

Table 17.
$\lfloor (a_n)_{calculated} \rfloor_2 (1,2)$

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

Table 18.
$Popfitness_2(1,2)$

20	40
5	10
20	10
5	2.5
20	40
20	40
5	10
20	10
20	40
20	40
20	10
5	10
20	10
20	40
20	40
20	10
5	10
20	10
5	2.5
20	40
5	2.5
20	10
5	2.5
20	40
5	2.5
5	10
20	40
20	10
20	40
5	10

Table 19.
$selected\_ind_2(1,2)$

20
29
14
6
6
5
27
14
27
27
20
5
9
20
10
5
15
15
1
10
14
24
27
6
20
10
20
29
5
1

Table 20. $selected\_pop_2(1,2)$		
0	0	1
0	0	1
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
0	0	1
1	0	0
1	0	0
0	0	1
1	0	0
1	0	0
1	0	0
1	0	0
0	0	1
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
0	0	1
1	0	0
0	0	1
0	0	1
1	0	0
0	0	1

Table 21. After Crossover for clock 2		
0	0	1
0	0	1
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
0	0	0
1	0	1
1	0	1
0	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
0	0	1
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
0	0	0
1	0	1
0	0	1
0	0	1
1	0	1
0	0	0

Table 22. After mutation for clock 2		
0	0	1
0	0	1
1	1	0
1	0	1
0	0	0
1	0	0
1	1	0
0	0	0
1	0	0
0	0	0
0	0	0
1	0	1
0	0	0
0	0	1
1	1	0
1	0	0
1	0	0
0	1	0
0	0	1
1	0	0
0	1	1
1	0	0
1	1	0
1	0	1
1	0	0
1	0	1
1	0	1
0	0	1
1	0	1
0	0	0

Clock 3

All a_n received so far are: 1, 1, 1.

current a_n bit is 0.

Table 23 shows the calculate a_n for clock 3 shift 1.

Table 24 shows calculate popfitness for clock 3 shift 1.

Table 25 shows calculated a_n for clock 3 shift 2.

Table 26 shows popfitness for clock 3 shift 1 and 2.

Table 27 shows calculated a_n popfitness for clock 3 shift 3.

Table 28 shows popfitness for clock 3 shift 3.

Average fitness for clock 3 shift 3 = 24.75.

Table 29 shows results after tournament selection.

Table 30 shows selected population for clock 3.

Number of correct IV population found = 30/30.

Table 31 shows results after crossover.

Table32 shows results after mutation.

Average fitness values for clock 1, 2, and 3 are: 17, 19.75, 24.75 respectively.

Table 23.
$[IV]_3(1)$

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

Table 24.
$Popfitness_3(1)$

20
20
20
5
5
20
20
5
20
5
5
5
5
20
20
20
20
5
20
20
20
20
20
5
20
5
5
20
5
5

Table 25.
$[IV]_3(2)$

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

Table 26.
$Popfitness_3(1,2)$

20	40
20	40
20	10
5	2.5
5	2.5
20	40
20	10
5	2.5
20	40
5	2.5
5	2.5
5	2.5
5	2.5
20	40
20	10
20	40
20	40
5	10
20	40
20	40
20	10
20	40
20	10
5	2.5
20	40
5	2.5
5	2.5
20	40
5	2.5
5	2.5

Table 27.
$\lfloor (a_n)_{calculated} \rfloor_3(1,2,3)$

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

Table 28. $Popfitness_3(1,2,3)$		
20	40	20
20	40	20
20	10	5
5	2.5	1.25
5	2.5	5
20	40	80
20	10	5
5	2.5	5
20	40	80
5	2.5	5
5	2.5	5
5	2.5	1.25
5	2.5	5
20	40	20
20	10	5
20	40	80
20	40	80
5	10	5
20	40	20
20	40	80
20	10	20
20	40	80
20	10	5
5	2.5	1.25
20	40	80
5	2.5	1.25
5	2.5	1.25
20	40	20
5	2.5	1.25
5	2.5	5

Table 29. $selected\_ind_3(1,2,3)$
25
17
6
6
17
25
17
6
22
16
25
6
9
9
6
6
6
16
22
17
25
6
25
20
16
16
6
6
17
16

Table 30. $selected\_pop_3(1,2,3)$
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0

Table 31. After Crossover for clock 3
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0

Table 32. After Mutation for clock 3
1 0 0
1 1 0
1 0 0
0 0 0
0 0 0
1 0 0
0 0 0
1 0 0
1 0 0
1 1 0
1 0 0
1 1 1
1 0 0
1 0 1
0 0 0
0 0 0
1 1 0
1 0 1
0 0 0
0 0 1
0 0 1
1 0 0
1 1 0
0 0 0
1 0 0
1 0 1
1 0 0
1 0 0
1 1 0
0 1 1

Clock 4

All a_n received so far are: 1, 1, 0, 1

current a_n bit = 1

Table 33 shows Calculated a_n for clock 4 shift 1.

Table 34 shows popfitness for clock 4 shift 1

Table 35 shows calculated a_n clock 4 shift 2.

Table 36 shows a_n for clock 4 shift 1 and 2.

Table 37 shows the popfitness for clock 4 shift 1 and 2.

Table 38 shows calculated a_n for clock 4 shift 3.

Table 39 shows calculated a_n for clock 4 shift 1, 2, and 3.

Table 40 shows popfitness for clock 4 shifts 1,2, and 3.

Clock 4 shift 1, 2, and 3 are calculated as shown for clock 3.

Table 41 shows calculated a_n for clock 4 shifts 4.

Table 42 shows popfitness for clock 4 shifts 1,2,3, and 4.

Table 43 shows calculated a_n for clock 4 1,2,3, and 4.

Table 44 shows popfitness for all clocks.

Average fitness for clock 4 is 62.25.

Table 45 shows results after the tournament selection.

Table 46 shows selected population for clock 4.

Number of correct IV population found = 30.

Table 47 shows results after crossover.

Table 48 shows results after mutation.

Table 33. $[IV]_4(1)$		
1	1	0
1	1	1
1	1	0
0	0	0
0	0	0
1	1	0
0	0	0
1	1	0
1	1	0
1	1	1
1	1	0
0	1	1
1	1	0
0	1	0
0	0	0
0	0	0
1	1	1
0	1	0
0	0	0
1	0	0
1	0	0
1	1	0
1	1	1
0	0	0
1	1	0
0	1	0
1	1	0
1	1	0
1	1	1
1	0	1

Table 34. $Popfitness_4(1)$
20
20
20
5
5
20
5
20
20
20
20
5
20
5
5
5
20
5
5
20
20
20
20
5
20
5
20
20
20
20

Table 35.
$[IV]_4(2)$

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

Table 36.
$\left[(a_n)_{calculated}\right]_4(1,2)$

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

Table 37.
$Popfitness_4(1,2)$

20	40
20	10
20	40
5	2.5
5	2.5
20	40
5	2.5
20	40
20	40
20	10
20	40
5	10
20	40
5	2.5
5	2.5
5	2.5
20	10
5	2.5
5	2.5
20	40
20	40
20	40
20	10
5	2.5
20	40
5	2.5
20	40
20	40
20	10
20	10

Table 38. $[IV]_4(3)$
0 1 1
1 0 1
0 1 1
0 0 0
0 0 0
0 1 1
0 0 0
0 1 1
0 1 1
1 0 1
0 1 1
0 1 0
0 1 1
1 0 0
0 0 0
0 0 0
1 0 1
1 0 0
0 0 0
1 1 1
1 1 1
0 1 1
1 0 1
0 0 0
0 1 1
1 0 0
0 1 1
0 1 1
1 0 1
0 0 1

Table 39. $\left[(a_n)_{calculated}\right]_4(1,2,3)$
1 1 0
1 0 1
1 1 0
0 0 0
0 0 0
1 1 0
0 0 0
1 1 0
1 1 0
1 0 1
1 1 0
0 1 0
1 1 0
0 0 1
0 0 0
0 0 0
1 0 1
0 0 1
0 0 0
1 1 1
1 1 1
1 1 0
1 0 1
0 0 0
1 1 0
0 0 1
1 1 0
1 1 0
1 0 1
1 0 0

Table 40. $Popfitness_4(1,2,3)$		
20	40	80
20	10	5
20	40	80
5	2.5	5
5	2.5	5
20	40	80
5	2.5	5
20	40	80
20	40	80
20	10	5
20	40	80
5	10	20
20	40	80
5	2.5	1.25
5	2.5	5
5	2.5	5
20	10	5
5	2.5	1.25
5	2.5	5
20	40	20
20	40	20
20	40	80
20	10	5
5	2.5	5
20	40	80
5	2.5	1.25
20	40	80
20	40	80
20	10	5
20	10	20

Table 41. $[IV]_4(4)$		
1	0	1
0	1	0
1	0	1
0	0	0
0	0	0
1	0	1
0	0	0
1	0	1
1	0	1
0	1	0
1	0	1
0	0	1
1	0	1
1	1	0
0	0	0
0	0	0
0	1	0
1	1	0
0	0	0
0	1	1
0	1	1
1	0	1
0	1	0
0	0	0
1	0	1
1	1	0
1	0	1
1	0	1
0	1	0
1	0	0

Table 42. $Popfitness_4(1,2,3,4)$			
20	40	80	160
20	10	5	2.5
20	40	80	160
5	2.5	5	2.5
5	2.5	5	2.5
20	40	80	160
5	2.5	5	2.5
20	40	80	160
20	40	80	160
20	10	5	2.5
20	40	80	160
5	10	20	10
20	40	80	160
5	2.5	1.25	2.5
5	2.5	5	2.5
5	2.5	5	2.5
20	10	5	2.5
5	2.5	1.25	2.5
5	2.5	5	2.5
20	40	20	10
20	40	20	10
20	40	80	160
20	10	5	2.5
5	2.5	5	2.5
20	40	80	160
5	2.5	1.25	2.5
20	40	80	160
20	40	80	160
20	10	5	2.5
20	10	20	40

Table 43. $\lfloor (a_n)_{calculated} \rfloor_4 (1,2,3,4)$			
1	1	0	1
1	0	1	0
1	1	0	1
0	0	0	0
0	0	0	0
1	1	0	1
0	0	0	0
1	1	0	1
1	1	0	1
1	0	1	0
1	1	0	1
0	1	0	0
1	1	0	1
0	0	1	1
0	0	0	0
0	0	0	0
1	0	1	0
0	0	1	1
0	0	0	0
1	1	1	0
1	1	1	0
1	1	0	1
1	0	1	0
0	0	0	0
1	1	0	1
0	0	1	1
1	1	0	1
1	1	0	1
1	0	1	0
1	0	0	1

Table 44.

$Popfitness_{1,2,3,4}(1,2,3,4)$

20	40	20	160
20	10	20	2.5
20	10	5	160
20	2.5	1.25	2.5
5	40	5	2.5
20	40	80	160
5	10	5	2.5
20	10	5	160
20	40	80	160
20	40	5	2.5
20	10	5	160
20	10	1.25	10
20	10	5	160
20	40	20	2.5
20	40	5	2.5
20	10	80	2.5
20	10	80	2.5
5	10	5	2.5
20	2.5	20	2.5
20	40	80	10
20	2.5	20	10
5	10	80	160
20	2.5	5	2.5
20	40	1.25	2.5
5	2.5	80	160
5	10	1.25	2.5
20	40	1.25	160
20	10	20	160
20	40	1.25	2.5
20	10	5	40

Table 45.

$selected\_ind_4(1,2,3,4)$

11
1
25
1
22
8
9
27
28
3
11
3
9
22
11
9
6
6
8
28
3
3
3
11
13
1
3
1
28
28

Table 46. $selected\_pop_4(1,2,3,4)$
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0

Table 47. After crossover for clock 4
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0
1 0 0

Table 48. After Mutation for clock 4		
0	0	1
1	0	0
1	0	0
0	0	1
1	1	0
1	0	0
1	0	1
1	1	1
0	0	0
1	0	1
1	0	0
1	0	0
1	0	1
1	0	0
1	1	0
1	1	0
1	0	0
1	0	0
1	0	0
1	0	0
0	1	0
0	1	0
0	0	0
1	0	1
1	0	0
1	0	0
1	0	0
0	0	0
0	0	0
1	0	0

The average fitness results are plotted in Figure 46 and the performance accuracy of the adaptive algorithm for this example is plotted in Figure 47.

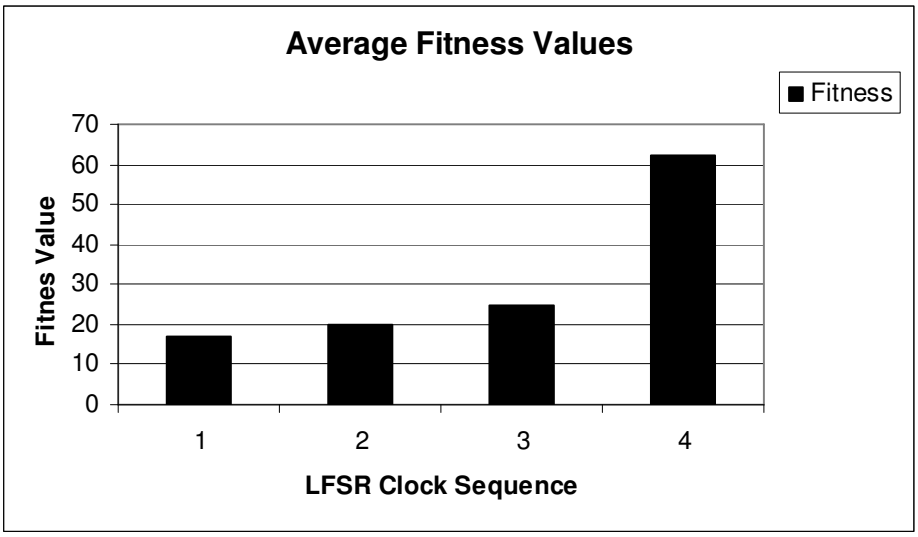

Figure 46. Average Fitness Values

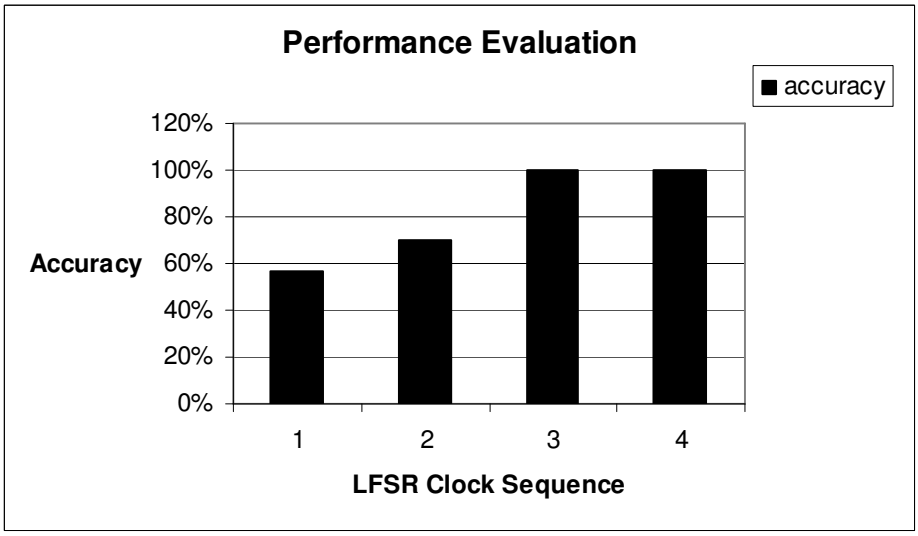

Figure 47. Performance Evaluation for 3 Stage LFSR

Bibliography

[1] 3rd Generation Partnership, "3rd Generation Partnership Project 2 (3GPP2) C.S0002-C, Physical Layer Standard for CDMA2000 Spread Spectrum Systems," Release C, Version1.0, May 28, 2002.

[2] 3rd Generation Partnership, "3rd Generation Partnership Project 2 (3GPP2) C.S0010-A, Recommended Minimum Performance Standards for Base Stations Supporting Dual-Mode Spread Spectrum Mobile Stations," April 2001.

[3] 3rd Generation Partnership, "3rd Generation Partnership Project 2 (3GPP2) C.S0011-A, Recommended Minimum Performance Standards for Dual-Mode Spread Spectrum Mobile Stations," April 2001.

[4] P. Agrawal and V. D. Agrawal, "Probabilistic analysis of random test generation method for irredundant combinational logic networks*," IEEE Trans. Comput*., pp. 691–695, July 1975.

[5] H. Amirazizi and M. Hellman, "Time memory-processor trade-offs", *IEEE Transactions on Information Theory*, 34 (1988), pp.505–512.

[6] J. D. Bagley, "The behavior of adaptive systems which employ genetic and correlation algorithms," doctoral dissertation, University of Michigan, 1967.

[7] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," *Proc. ICGA 2*, pp. 14-21, 1987.

[8] J. E. Baker, "Adaptive Selection Methods for Genetic Algorithms," *Proc. ICGA* 1, pp. 101-111, 1985.

[9] P. H. Bardell, W. H. McAnney, and J. Savir, "Built-In Test for VLSI Pseudorandom Techniques," Wiley, New York, 1987.

[10] M. Barrett, et al, "Intelligent Agents for Vulnerability Assessment of Computer Networks," Proceedings of the 2nd Annual FedLab Symposium – Advanced Telecommunications & Information Distribution, U.S. Army Research Labs, 1998.

[11] M.J. Beller, L.-F. Chang, And Y. Yacobi, "Privacy and authentication on a portable communications system," *IEEE Global Telecommunications Conference*, pp. 1922–1927, 1991.

[12] E. R. Berlekamp, Algebraic Coding Theory, Aegean Park Press, Walnut Creek, CA, 1982.

[13] M.C. Bernardes, E. Moreira, "Implementation of an Intrusion Detection System based on Mobile Agents," Proceedings of International Symposium on Software Engineering for Parallel and Distributed Systems, 2000, pp. 158-164.

[14] R. Biesbroek, GA tutorial homepage: http://www.estec.esa.nl/outreach/gatutor/, modified 02/17/2000.

[15] W. Booth, et al, "ATIRP Task 5.5 Intelligent Agents System Requirements and Architecture," U.S. Army Research Labs, October 1998.

[16] M. F. Bramlette and R. Cusin, "A Comparative Evaluation of Search Methods Applied to Parametric Design of Aircraft," Proc. ICGA 3, pp213-218, 1989.

[17] G. Castellano, A. Fanelli, E. Gentile and T. Rosell, "A GA-based approach to optimization of fuzzy models learned from data," Genetic Evolution and Computation Conference 2002 (GECCO-2002) Workshop Proceedings, pp , July 9, 2002, New York, NY.

[18] A. J. Chipperfield, P. J Fleming, P. J., Pohlheim, and C. M. Fonseca, "A Genetic Algorithm Toolbox for MATLAB," Proc. International Conference on Systems Engineering, Coventry, UK, 6-8 September 1994.

[19] A. J. Chipperfield and P. J Fleming, "The MATLAB Genetic Algorithm Toolbox," IEE Colloquium on Applied Control Techniques Using MATLAB, Digest No. 1995/014, 26/01/95.

[20] M. Conner, et all, "Genetic Algorithm/Artificial Life Evolution of Security Vulnerability Agents," Proceedings of the 3rd Annual FedLab Symposium – Advanced Telecommunications & Information Distribution, U.S. Army Research Labs, 1999.

[21] Y, Davidor, Genetic Algorithms and Robotics: a Heuristic Strategy for Optimization. Singapore: World Scientific Computing Co., 1991.

[22] L. Davis, "Handbook of Genetic Algorithms," Van Nostrand Reinhold, New York, 1991.

[23] K. De Jong, An analysis of the behavior of a class of genetic adaptive systems, Doctoral dissertation, University of Michigan, 1975.

[24] E. B. Eichelberger, E. Lindbloom, J. A. Waicukauski, and T. W. Williams, Structured Logic Testing, Englewood Cliffs, NJ: Prentice-Hall, 1991.

[25] D. B. Fogel, "An Introduction to Simulated Evolutionary Optimization". *IEEE Trans. on Neural Networks:* Special Issue on Evolutionary Computation, Vol. 5, No. 1, pp. 3-14, 1994.

[26] D. B. Fogel, Evolving Artificial Intelligence, Dissertation, University of California, San Diego, 1992.

[27] V. Garg, Wireless Network Evolution: 2G to 3G, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 2002.

[28] D. E. Goldberg, Computer-aided gas pipeline operation using genetic algorithms and rule learning, Doctoral dissertation, University of Michigan, 1983.

[29] D. E. Goldberg and K. Deb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms, Foundations of Genetic Algorithms," San Mateo, California, USA: Morgan Kaufmann Publishers, pp. 69-93,1991.

[30] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning," Addison Wesley, Reading, MA, 1989.

[31] S. W. Golomb, Shift Register Sequences, Holden-Day, San Francisco, 1967.

[32] S. W. Golomb, "Shift-Register Sequences and Spread-Spectrum Communications," Keynote Address, *IEEE Third International Symposium on Spread Spectrum Techniques and Applications*, Oulu, Finland, July 4 –6, 1994.

[33] S. W. Golomb, Shift Register Sequences, Aegean Park Press, Walnut Creek, CA, 1982.

[34] V. Grant, The Evolutionary Process, Columbia University Press, New York, 1985.

[35] M. Grewal and A. Andrews, Kalman Filtering: Theory and Practice, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[36] J. Haines, et al, “Extending the DARPA Off-Line Intrusion Detection Evaluations,” Proceedings of DARPA Information Survivability Conference & Exposition II, Volume: 1, 2001, pp. 35-45.

[37] K. Hassan and M. Conner, "Identifying Security Threats In Ad Hoc Wireless Network," Proc. of the international conference on security and management (SAM) 2003, Las Vegas, Nevada, pp. 34-38, June 23-26, 2003.

[38] K. Hassan and M. Conner, "Adaptive Scheme For Identifying Network Threats From Pseudo-Random Sources," Proc. of Collaborative Technology Alliances (CTA) and Communications & Networks (C&N) Alliance 2003 Annual Symposium, University of Maryland Conference Center, April 29-May 1, 2003.

[39] K. Hassan and M. Conner, "Predicting Polynomial Feedback of Linear Feedback shift Register (LFSR) using Genetic Algorithm,” Genetic Evolution and Computation Conference 2002 (GECCO-2002) Proceedings, pp 14-18, July 9, 2002, New York, NY.

[40] K. Hassan and M. Conner, “Smart Synchronization for CDMA Mobile Stations,” submitted to *IEEE Pervasive Computing: Mobile & Ubiquitous Systems*.

[41] K. Hassan "Cellular Optimization,” Cellular Business: the Journal of Cellular Telecommunications, pp. 122-126, September 1995.

[42] K. Hassan, Traffic Reporting System and Method over Wireless Communication Systems, US patent No. 6813247, Issued Nov. 02, 2004.

[43] K. Hassan, “Steering Filter Implementation for Digital Beamformers,” Master of Science in Electrical Engineering thesis, University of North Carolina at Charlotte, 1994.

[44] R. Haupt, and S. Haupt, Practical Genetic Algorithms, John Wiley and Sons, Inc., New York, NY, 1998.

[45] J. C. Hernandez et al., “Using Classifiers to Predict Linear Feedback Shift Registers,” Security Technology, 2001 IEEE 35th International Carnahan Conference, pp. 240 -249.

[46] J. C. Hernandez, et al., “Using the general next bit predictor as an evaluation criteria”, First New European Schemes for Signatures, Integrity, and Encryption (NESSIE) Workshop, KU Leuven, Belgium. October 2000.

[47] J. H. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, MI: University of Michigan Press, 1975.

[48] R. B. Hollstien, Artificial genetic adaptation in computer control systems. Doctoral dissertation, University of Michigan, 1971.

[49] M. Huesken, Y. Jin and B. Sendhoff, "Structure optimization of neural networks for evolutionary design optimization," Genetic Evolution and Computation Conference 2002 (GECCO-2002) Workshop Proceedings, pp , July 9, 2002, New York, NY.

[50] W.C. Jakes, Jr., Microwave Mobile Communications. New York: Wiley, 1974.

[51] O. Kachirski, R. Guha, "Intrusion Detection Using Mobile Agents in Wireless Ad Hoc Network," Proceedings of the IEEE Workshop on Knowledge Media Network (KMN'02), 2002.

[52] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge: MIT Press, 1992.

[53] J. R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs. Cambridge: MIT Press, 1994.

[54] K. Krishnakumar and D. E. Goldberg, "Control System Optimization Using Genetic Algorithms," *Journal of Guidance, Control and Dynamics*, Vol. 15, No. 3, pp. 735-740, 1992.

[55] A.C.Y. Lee, Mobile Communications Engineering, New York: McGraw-Hill, 1982.

[56] W.C.Y. Lee, "Overview of cellular CDMA," *IEEE Trans. Vehic. Technol.,* Vol. VT-40, No. 2, pp. 291-302, may 1991.

[57] Q. Li, et al., "Scattering center analysis of radar targets using fitting scheme and genetic algorithm," *IEEE trans. Antennas Propagation Systems,* APS-42(2), pp. 198-207.

[58] R. Lippmann et. al., "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation," Proceedings of DARPA Information Survivability Conference & Exposition II, Volume: 2, 1999, pp. 12-26.

[59] J.L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions of Information Theory,* vol. 15, pp. 122-127. 1969.

[60] Z. Michalewicz, Genetic Algorithms + Data Stractures = Evolution Programs, 2nd ed., Springer-Verlag, New York, 1994.

[61] M. Negnevitsky, Artificial Intelligence: a guide to Intelligent System, Addison Wesley, Essex, England, 2002.

[62] C.H. Papadimitriou, Computational Complexity, Addison-Wesley publishing company, Inc., Reading, MA, 1994.

[63] H. Pohlheim, Evolutionary Algorithms: overview, methods, and operators. http://www.geatbx.com/, December 1999.

[64] T. Pratt, et al., Satellite communications: second edition, John Wiley & Sons, Inc., Hoboken, NJ, 2003.

[65] J.G. Proakis, Digital Communications. New York: McGraw-Hill, 1983.

[66] T. Rappaport, Wireless Communications, Principle & Practice, Prentice Hall, Upper Saddle River, NJ, 1996.

[67] Kh. Rasheed, X. Ni and S. Vattam, "Comparison of methods for using reduced models to speed up design optimization," Genetic Evolution and Computation Conference 2002 (GECCO-2002) Workshop Proceedings, pp, July 9, 2002, New York, NY.

[68] M. Y. Rhee, CDMA Cellular Mobile Communications and Network Security," Prentice Hall, Upper Saddle River, NJ, 1998.

[69] S.O. Rice, "Mathematical analysis of random noise," *Bell System tech.* J., Vol. 23, No. 7, pp. 282-333, July 1944, and Vol. 24, No. 1, pp. 46-156, Jan. 1945.

[70] *R. S Rosenberg.* Simulation of genetic populations with biochemical properties. Doctoral dissertation, University of Michigan, Doctoral Dissertation, 1967.

[71] C. Ryan, Reducing Premature Convergence in Evolutionary Algorithms. Doctoral Dissertation, University College Cork, Ireland, 1996.

[72] J. Savir, G. S. Ditlow, and P. H. Bardell, "Random pattern testability," *IEEE Trans. Comput*., pp. 79–90, Jan. 1984.

[73] C.E. Shannon, "Communications in Presence of Noise," Proceedings of IRE, 1949, No. 37, pp.10-21.

[74] D. Stinson, Cryptology: Theory and Practice," CRC Press, Boca Raton, Florida, 1995.

[75] G. Stüber, Principles of Mobile Communication, Kluwer Academic Publishers, Norwell, MA, 1996.

[76] TIA/EIA, "Recommended Minimum Performance Standards for Base Stations Supporting Dual-Mode Spread Spectrum Mobile Stations," TIA/EIA-97-D, October 23,2000.

[77] TIA/EIA, "Recommended Minimum Performance Standards for Dual-Mode Spread Spectrum Mobile Stations," TIA/EIA-98-D, November 27,2000.

[78] TIA/EIA, "Interim Standard TIA/EIA/IS-95A: Mobile Station - Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System," TIA/EIA, January 11,1996.

[79] A. J. Viterbi, CDMA: the principles of Spread Spectrum Communications, Addison-Wesley, 1995.

[80] A.J. Viterbi and J.K. Omura, Principles of Digital Communications and Coding. New York: McGraw-Hill, 1979.

[81] J. Wachura, "Bit Error Analysis and Beyond," Communication Design Engineering Conference, Jan. 2005.

[82] D. Whitley, T. Starkweather, and D. Shaner, The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination, in L. Davis (Ed.), handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[83] S. B. Wicker, Error Control Systems, Prentice-Hall Inc., Upper Saddle River, NJ, 1995.

[84] Y. Zhang and W. Lee, "Intrusion Detection in Wireless Ad-Hoc Networks," Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, MobiCom'2000, pp. 275-283.

[85] L. Zhou and Haas L, "Securing Ad Hoc Networks," *IEEE Network*, Volume 13, Issue 6, pp. 24-30, November 1999.