4

Pseudo-Random Code Sequences for Spread-Spectrum Systems

4.1 Introduction

In the previous chapter we learnt that essential to the spread-spectrum system is the spreading of the data energy at the transmitter, and the collapsing of the spreading (in a process known as de-spreading) at the receiver. The operations at the transmitter and the receiver generate the required processing gain without which the system could not combat jamming and interference. The spreading code sequences, used in the system, have special properties which will be introduced in this chapter. These code sequences have an important role to play in the spread-spectrum technique and have to be chosen carefully for efficient communication systems.

Each code sequence used in the spread-spectrum communications must easily be distinguishable from a time shifted version of itself, in order to enable the code acquisition and tracking and, therefore, the synchronization of the system. Furthermore, when multiple users are accessing the system for services, each code sequence assigned to a user must be distinguishable from every other user code sequence in the set and ideally should generate little or no interference to other users sharing the channel.

The code sequences are pseudo-random sequences but in practice they are generally periodic. Two types of these sequences can be used in spread-spectrum applications: the binary code sequence and the non-binary (also known as complex) code sequence. The elements of the binary sequence are made up of real numbers ± 1. These are most commonly applied in spread-spectrum communications, for ease of generation using widely available binary logic circuits. However, there has also been some interest in generating complex sequences through exhaustive computer searching, such as quadric phase and poly phase sequences which have low correlation properties.

The bit error probability performance of spread-spectrum communications depends on the correlation properties of the code sequence used since the latter define the amount of interference generated from multiple access. The technique for generating code sequences should be aimed at a large family of sequences in order to accommodate a number of users and, with an impulse-type autocorrelation which enhances the system synchronization and possibly with low cross-correlation functions, to reduce multiple access interference.

We start this chapter with basic binary algebra in Sections 4.2, 4.3 and 4.4. This is followed by block level structure of the commonly used binary pseudo-random sequence generators. The binary sequences are called *maximal-length sequences* or simply *m-sequences*. Their decimation and the preferred pairs of the m-sequence are discussed, together with other sequences widely used in spread spectrum such as the Gold, Kassami and Walsh sequences, in detail in Section 4.5. Finally, Section 4.6 introduces the complex sequences such as quadric phase, poly phase sequences, and the whole chapter is summarized in Section 4.7.

4.2 Basic Algebra concepts

We start this section with a basic definition of terms used in the numbers theory such as number set, group and field. The reader who may be unfamiliar with these terms may then feel at ease reading the section.

An algebraic *set* of M elements is defined by an array of M real or complex numbers acted upon by an operator for addition (and its inverse, subtraction) and division or a multiplication (and its inverse, division). The set is said to be a *closed set* if the algebraic operations (addition and division or multiplication) on the original set, yield a new element already existing in the same set. A *group* is a set of elements acted upon by an operator for addition (additive group) or multiplication (multiplicative group). A *ring* is a set of elements operated upon by addition and multiplication. A *field* is defined as a ring with every element in the ring (except zero) having an inverse.

A field is therefore a system that has two operations both with inverses and each element in the field (except zero) having an inverse. A field can be real or complex, depending on the type of its constituting elements being real or complex numbers. An infinite field has a theoretically infinite number of elements.

A field with a finite number of elements, M, is called a Galois (pronounced as 'Gal-Wah') field and is denoted GF(M). Generally, finite fields only exist when M is prime or M is the power of a prime, i.e. $M = P^m$ when m is integer. Galois field GF(M) has M elements with index $0, 1, 2, \ldots, M-1$. The simplest Galois field uses modulo 2 arithmetic and is denoted GF(2) with elements drawn from {0, 1} which is also called a binary field.

The elements of a given set must satisfy precise rules to become a constituent member of the field. Let e_i, e_j, e_k, e_n be any four elements of a set. The field requires the set to possess the following properties:

1 Commutative property

$$\begin{aligned} e_i + e_j &= e_j + e_i \\ e_i \cdot e_j &= e_j \cdot e_i \end{aligned} \tag{4.1}$$

2 Associative property

$$\begin{aligned} e_i + (e_j + e_k) &= (e_i + e_j) + e_k \\ e_i \cdot (e_j \cdot e_k) &= (e_i \cdot e_j) \cdot e_k \end{aligned} \tag{4.2}$$

3 Distributed property

$$\begin{aligned} e_i \cdot (e_j + e_k) &= e_i \cdot e_j + e_i \cdot e_k \\ (e_i + e_j) \cdot (e_k + e_n) &= e_i \cdot e_k + e_i \cdot e_n + e_j \cdot e_k + e_j \cdot e_n \end{aligned} \tag{4.3}$$

4 Inverse property

$$e_i + e_i^* = 0 \quad \text{additive inverse} \tag{4.4}$$

$$e_i \cdot e_i^{-1} = 1 \;\; e_i \neq 0 \quad \text{multiplication inverse} \tag{4.5}$$

where e_i^* and e_i^{-1} denote inverse elements.

5 Closure property

$$\begin{aligned} e_i + e_j &\in \mathrm{GF}(\cdot) \\ e_i \cdot e_j &\in \mathrm{GF}(\cdot) \end{aligned} \tag{4.6}$$

Subtraction of one element from any other element in the given field is performed by adding the additive inverse of the element while division by an element (other than zero) is accomplished by multiplying by the multiplicative inverse.

Example 4.1

Consider a set of binary elements drawn from {0, 1}. If the basic algebraic operations (addition/subtraction and multiplication/division) are acted upon each pair of the set, find the resulting elements.

Solution

1 Mod-2 addition

+	0	1
0	A1	A2
1	B1	B2

$A1 = 0$; $A2 = 1$; $B1 = 1$; $B2 = 0 \pmod 2$

2 Mod-2 subtraction

−	0	1
0	A1	A2
1	B1	B2

Here we apply the additive inverse such that:
$0 = -0$
$1 = -1$
Therefore, $A1 = 0$; $A2 = 1$; $B1 = 1$; $B2 = 0$ (mod 2)

3 Mod-2 multiplication

×	0	1
0	A1	A2
1	B1	B2

$A1 = 0$; $A2 = 0$; $B1 = 0$; $B2 = 1$

4 Mod-2 division

÷	0	1
0	A1	A2
1	B1	B2

Here we use the multiplicative inverse:
$A1 = ?$; $A2 = ?$; $B1 = 0$; $B2 = 1$

4.3 Arithmetic of binary polynomial

Galois's theory is based on algebra articulated by a young French mathematician (Evariste Galois [1811–1832]). Galois fields are extensively used in applications such as cryptography, error correcting codes and random number generators.

Galois field binary polynomials are seen as mathematical equivalents *of Linear Feed-Back Shift Registers* (LFSRs) in sequence generator design, using widely available logic gates such as X-OR, AND, and OR gates. This subject will be dealt with in more detail in Section 4.5.

In most applications of Galois Field GF(M), M is chosen as a prime number, i.e. *M is a positive integer not divisible, without a remainder, by any positive integer other than itself and one*. The elements in the field are integer numbers. Furthermore, M corresponds to an integral power of a prime number. A field GF(2^m) is considered to be an extension of the binary field GF(2), so to generate a field GF(2^m), we extend the constituent

elements {0, 1} of GF(2) using a primitive element 3. If $3 \in GF(2^m)$, then under multiplication $3.3 = 3^2$, $3.3^2 = 3^3, \ldots, 3^m - 1$, are also elements of $GF(2^m)$. Therefore, the elements of $GF(2^m)$ are:

$$GF(2^m) = \{0, 1, \beta, \beta^2, \beta^3, \ldots\ldots\ldots, \beta^m - 1. \tag{4.7}$$

Consider a polynomial P(x) of degree m and coefficients that are chosen from elements of the field GF(p) such that:

$$p(x) = \sum_{i=0}^{m} h_i \cdot x^i = h_0 + h_1 x + h_2 x^2 + \cdots\cdots \tag{4.8}$$

For example, the polynomial $[1 + x^2]$ is a second order with coefficients drawn from the binary field (i.e. $h_0 = 1$, $h_1 = 0$, $h_2 = 1$). On the other hand, the polynomial $[1 + x + 3x^3]$ is a third order polynomial with coefficients drawn from the field $GF(2^2)$ (i.e. $h_0 = 1$, $h_1 = 1$, $h_2 = 0$, $h_3 = 3$).

Example 4.2

Consider the polynomials $P_1(x)$ and $P_2(x)$ such that:

$$P_1(x) = 1 + x + x^3$$
$$P_2(x) = x + x^2 + x^3$$

Evaluate the following mathematical expressions:

i. $P_1(x) + P_2(x)$
ii. $P_1(x) - P_2(x)$
iii. $P_1(x) \times P_2(x)$
iv. $\dfrac{P_1(x)}{P_2(x)}$

Solution

Since both polynomials have binary coefficients, we can use binary arithmetic in the analysis.

i. $P_1(x) + P_2(x) = 1 + x + x^3 + x + x^2 + x^3 = 1 + x^2$ since $x + x = 0$ and $x^3 + x^3 = 0$
ii. $P_1(x) - P_2(x) = 1 + x^2$ Binary addition and subtraction are the same.
iii. $P_1(x)\, P_2(x) = (1 + x + x^3)\,(x + x^2 + x^3)$
$= x + x^2 + x^3 + x^2 + x^3 + x^4 + x^4 + x^5 + x^6$
$= x + x^5 + x^6$

iv. $\dfrac{P_1(x)}{P_2(x)}$ Using long division, we get:

$$\begin{array}{r|l} & 1 \\ \hline x^3+x^2+x & x^3+x+1 \\ & x^3+x^2+x \\ \hline & 1+x+x+x^2+x^3+x^3=x^2+1 \end{array}$$

Thus $\dfrac{P_1(x)}{P_2(x)} = 1 + \dfrac{x^2+1}{x^3+x^2+x}$

Example 4.3

Consider polynomials $P_1(x)$ and $P_2(x)$ with coefficients drawn from Galois field GF(3) such that:

$$P_1(x) = x + 2x^2 + x^3$$
$$P_2(x) = 1 + 2x + x^2$$

Evaluate the following mathematical expressions:

i. $P_1(x) + P_2(x)$
ii. $P_1(x) - P_2(x)$
iii. $P_1(x) \times P_2(x)$
iv. $\dfrac{P_1(x)}{P_2(x)}$

Solution

The arithmetic rules for Galois field GF(3) are:

Addition:

+	0	1	2
0	**0**	**1**	**2**
1	**1**	**2**	**0**
2	**2**	**0**	**1**

Subtraction:

Since $2+1=0$, then $-0=-0$, $-1=2$ and $-2=1$

–	0	1	2
0	**0**	**1**	**2**
1	**2**	**0**	**1**
2	**1**	**2**	**0**

Multiplication:
This is a straight forward arithmetic operation.

×	0	1	2
0	**0**	**0**	**0**
1	**0**	**1**	**2**
2	**0**	**2**	**1**

Division:
This is accomplished by multiplying by the multiplicative inverse.

$1^{-1} = 1$ and $2^{-1} = 2$

÷	0	1	2
0	**?**	**?**	**?**
1	**0**	**1**	**2**
2	**0**	**2**	**1**

i. $P_1(x) + P_2(x) = x + 2x^2 + x^3 + 1 + 2x + x^2 = 1 + x^3$
ii. $P_1(x) - P_2(x) = x + 2x^2 + x^3 - 1 - 2x - x^2 = 2 + 2x + x^2 + x^3$
iii. $P_1(x) \times P_2(x) = (x + 2x^2 + x^3) \times (1 + 2x + x^2)$
$= x + 2x^2 + x^3 + 2x^2 + 4x^3 + 2x^4 + x^3 + 2x^4 + x^5$
$= x + 4x^2 + 6x^3 + 4x^4 + x^5$
$= x + x^2 + x^4 + x^5$
iv. $\dfrac{P_1(x)}{P_2(x)} = x$

An irreducible polynomial is a polynomial that cannot be factored into non-trivial polynomials over the same field. For example, polynomial f(x) is said to be irreducible if it cannot be expressed as a product of at least two non-trivial polynomials in the f(x) field. An irreducible polynomial of degree m is primitive if it divides $[1 + x^n]$ for which the smallest positive integer $n = 2^m - 1$.

In general, a primitive polynomial exists for every irreducible polynomial with positive integer m. The addition of any two elements of GF(2^m) is defined as mod-2 addition of two binary polynomials. A multiplication of two elements of GF(2^m) is referred to as modulo-h(x) multiplication where h(x) is a primitive polynomial of order m.

Example 4.4

Find the elements that arise from the addition and multiplication of each pair of elements of the polynomials in GF(2^2).

Solution
The polynomials in GF(2^2) have 4 elements and each may be represented by a binary polynomial of order $m-1$ where $m=2$. These elements can be expressed as binary digits: 00, 01, 10, 11 and in binary polynomial as: 0, 1, x, $1+x$.

Addition:

+	0	1	x	1 + x
0	**0**	**1**	**x**	**1 + x**
1	**1**	**0**	**1 + x**	**x**
x	**x**	**1 + x**	**0**	**1**
1 + x	**1 + x**	**x**	**1**	**0**

Multiplication:
To carry out the multiplication, we choose the following primitive polynomial, h(x) of degree 2:

$$h(x) = 1 + x + x^2$$

Now the mathematical operations that we need to evaluate are:

$$x \times x \bmod [h(x)] = (1 + x)$$
$$(1 + x) \times x \bmod [h(x)] = (x + x^2) \bmod [h(x)] = 1$$
$$(1 + x) \times (1 + x) \bmod [h(x)] = (1 + x + x + x^2) \bmod [h(x)] = x$$

×	0	1	x	1 + x
0	**0**	**0**	**0**	**0**
1	**0**	**1**	**x**	**1 + x**
x	**0**	**x**	**1 + x**	**1**
1 + x	**0**	**1 + x**	**1**	**x**

4.4 Computing elements of GF(2^m)

We now explain how to find the elements in GF(2^m) by a worked example. Consider a primitive polynomial $p(x_i)$ of degree m which is primitive over GF(2^m). Since $p(x_i)$ has a finite number of roots, if we associate these roots as elements of GF(2^m), this will ensure that GF(2^m) is a finite field. A polynomial of degree m has m roots, x_i, that satisfies the equation:

$$p(x_i) \approx 0 \tag{4.9}$$

Let us consider the elements in GF(2^3) using the primitive polynomial $P(x) = 1 + x + x^3$.

The polynomial in GF(2^3) has 8 elements. Since β is the root of P(x) then:

$$P(\beta) = 1 + \beta + \beta^3 = 0$$

Therefore, $1 + \beta = \beta^3$

The elements due to power of β are:

$$\beta^0, \beta, \beta^2, \beta^3, \beta^4, \beta^5, \beta^6, \beta^7$$

Therefore, the elements are:

$$0, 1, \beta, \beta^2, \ldots, \beta^7, \quad \text{where :}$$

$$\beta^3 = 1 + \beta,$$

$$\beta^4 = \beta\beta^3 = \beta(1 + \beta) = \beta + \beta^2,$$

$$\beta^5 = \beta\beta^4 = \beta^2(1 + \beta) = \beta^2 + \beta^3 = 1 + \beta + \beta^2$$

$$\beta^6 = \beta^3 \cdot (1 + \beta) = (1 + \beta) \cdot (1 + \beta) = \beta^3 + \beta^4 = 1 + \beta + \beta + \beta^2 = 1 + \beta^2$$

$$\beta^7 = \beta(1 + \beta^2) = \beta + \beta^3 = 1 + \beta + \beta = 1$$

4.5 Binary pseudo-random sequences

4.5.1 Generation of binary pseudo-random sequences

We have hinted in Section 4.3 that shift registers, with linear feedback, can be used in the implementation of binary code sequence generators. We now examine this proposal in more detail. Our knowledge of the Galois field algebra will help us to understand the involved algebraic analysis. To explain the basic concept of the sequence generators, consider the simple feedback shift registers shown in Figure 4.1 where the initial states of the r-stage shift registers are (a_{r-1}, $a_{r-2}, \ldots,$ a_1, a_0) and the feedback function $f(x_0, x_1, \ldots, x_{r-1})$ is a binary function.

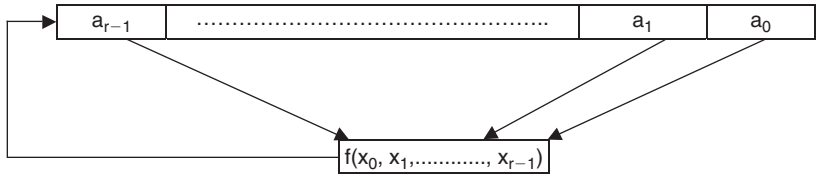


Figure 4.1 *Block diagram shift registers with linear feedback.*

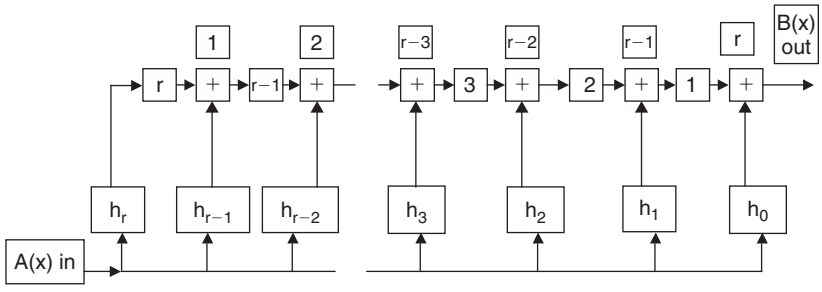


Figure 4.2 *Block diagram of a general sequence generator (Peterson et al., 1995).*

At each clock pulse, the content of each register is shifted to the next register on the left or right. For example, for shift right, the next state shown in Figure 4.1 is shown as:

a_r	……..	a_2	a_1

Where $a_r = f(a_0, a_1, \ldots\ldots\ldots\ldots, a_{r-1})$.

We now show that the feedback shift register circuit performs the multiplication of Galois polynomials. Consider the block diagram of a general sequence generator depicted in Figure 4.2 where we use the following symbols:

[j]	j^{th} stage shift register # j
[+]	Modulo-2 adder
[h_i]	Modular-2 multiplier

Let x denote the time delay of a unit clock duration and x^j denote the time delay of j such units. The input A(x) specifies the initial states of the registers and are denoted by the sequence $(a_{r-1}, a_{r-2}, \ldots, a_1, a_0)$.

The generator connections can be expressed by polynomial h(x) where:

$$h(x) = h_0 + h_1 \cdot x + h_2 \cdot x^2 \ldots\ldots\ldots\ldots\ldots h_r \cdot x^r \tag{4.10}$$

The coefficients h_i are such that a connection is present if $h_i = 1$ and no connection is present if $h_i = 0$. It is worth noting that there is no output from this generator for the first r clock time shifts and that once all registers are loaded with zeros (i.e. $A(x) = 0$), the generator could not change it's state. *Therefore an all-zero state is not allowed.*

Considering Figure 4.2, the top numbers (from left to right $1, 2, \ldots, r-2, r-1, r$) represent the adders. The numbers in the figure represent the numbers of the registers (from right to left $1, 2, 3, \ldots, r-2, r-1, r$). The output of the j^{th} adder is:

$$x: B_j(x) = B_{j-1}(x)x + A(x)h_{r-j}$$

Let us start the iteration from right to the left:

$$
\begin{aligned}
B(x) &= B_r(x)\cdot \\
&= B_{r-1}(x)\cdot x + A(x)\cdot h_0 \\
&= B_{r-2}(x)\cdot x^2 + A(x)\cdot xh_1 + A(x)\cdot h_0 \\
&= B_{r-3}(x)\cdot x^3 + A(x)x^2h_2 + A(x)\cdot xh_1 + A(x)\cdot h_0 \\
&\quad \bullet \\
&\quad \bullet \\
&\quad \bullet \\
&\quad \bullet \\
&\quad \bullet \\
&= B_1(x)\cdot x^{r-1} + A(x)x^{r-2}h_{r-2} + A(x)\cdot x^{r-1}h_{r-1} + \cdots\cdots\cdots\cdots + A(x)\cdot h_0 \\
&= A(x)x^r h_r + A(x)x^{r-1}h_{r-1} + A(x)x^{r-2}h_{r-2} + A(x)\cdot x^{r-1}h_{r-1} + \cdots\cdot + A(x)\cdot h_0 \\
&= \sum_{j=0}^{r} [A(x)\cdot x^j]\cdot h_j
\end{aligned}
$$

Thus
$$B(x) = A(x)\cdot h(x) \tag{4.11}$$

Equation (4.11) expresses the generator output B(x) as a product of two polynomials, i.e. the input polynomial A(x) and the generator connections polynomial h(x).

The maximum period of the binary sequence generated by the r-stage shift register is limited to $2^r - 1$. A binary sequence which achieves this maximum period is called *maximal-length sequence* or simply *m-sequence*. Primitive polynomials which can be used to connect the feedback are given in Table 4.1. It must be emphasized that the period of the generated sequence depends on the choice of h(x) and only connections based on these primitive polynomials are capable of generating sequences of length $2^r - 1$.

Having demonstrated that feedback shift register circuits are capable of performing multiplication of Galois polynomials, we now consider the block diagram depicted in Figure 4.3 to show that these circuits can also perform division of Galois polynomials. We use the same symbols in Figure 4.2.

Applying (4.11) to Figure 4.3a, we get:

$$B(x) = A_1(x)\cdot h(x) + A_2(x)\cdot k(x) \tag{4.12}$$

Suppose that $k_0 = 0$ so that the connection between multiplier k_0 and the corresponding adder is disconnected and that $A_2(x)$ is taken from the output, i.e. $A_2(x) = B(x)$. Let us define the polynomial g(x) such that $k(x) = g(x) + 1$.

Table 4.1 *Primitive polynomials for m-sequence generator connections*

Number of SR stages r	Sequence length $N = 2^r - 1$	Number of m-sequences	Polynomial generator h(x)
2	3	1	$1 + x + x^2$
3	7	2	$1 + x + x^3$ $1 + x^2 + x^3$
4	15	2	$1 + x + x^4$ $1 + x^3 + x^4$
5	31	6	$1 + x^2 + x^5$ $1 + x^3 + x^5$ $1 + x + x^2 + x^3 + x^5$ $1 + x + x^2 + x^4 + x^5$ $1 + x + x^3 + x^4 + x^5$ $1 + x^2 + x^3 + x^4 + x^5$
6	63	6	$1 + x + x^6$ $1 + x + x^3 + x^4 + x^6$ $1 + x^5 + x^6$ $1 + x + x^2 + x^5 + x^6$ $1 + x^2 + x^3 + x^5 + x^6$ $1 + x + x^4 + x^5 + x^6$
7	127	18	$1 + x + x^7$ $1 + x^3 + x^7$ $1 + x + x^2 + x^3 + x^7$ $1 + x^4 + x^7$ $1 + x^2 + x^3 + x^4 + x^7$ $1 + x + x^2 + x^5 + x^7$ $1 + x + x^3 + x^5 + x^7$ $1 + x^3 + x^4 + x^5 + x^7$ $1 + x + x^2 + x^3 + x^4 + x^5 + x^7$ $1 + x^6 + x^7$ $1 + x + x^3 + x^6 + x^7$ $1 + x + x^4 + x^6 + x^7$ $1 + x^2 + x^4 + x^6 + x^7$ $1 + x^2 + x^5 + x^6 + x^7$ $1 + x + x^2 + x^3 + x^5 + x^6 + x^7$ $1 + x^4 + x^5 + x^6 + x^7$ $1 + x + x^2 + x^4 + x^5 + x^6 + x^7$ $1 + x^2 + x^3 + x^4 + x^5 + x^6 + x^7$
8	255	16	$1 + x^2 + x^3 + x^4 + x^8$ $1 + x + x^3 + x^5 + x^8$ $1 + x^2 + x^3 + x^5 + x^8$ $1 + x^2 + x^3 + x^6 + x^8$ $1 + x + x^2 + x^3 + x^4 + x^6 + x^8$ $1 + x + x^5 + x^6 + x^8$

(*Continued*)

Table 4.1 *Continued*

Number of SR stages r	Sequence length $N = 2^r - 1$	Number of m-sequences	Polynomial generator h(x)
			$1 + x^2 + x^5 + x^6 + x^8$ $1 + x^3 + x^5 + x^6 + x^8$ $1 + x^4 + x^5 + x^6 + x^8$ $1 + x + x^2 + x^7 + x^8$ $1 + x^2 + x^3 + x^7 + x^8$ $1 + x^3 + x^5 + x^7 + x^8$ $1 + x + x^6 + x^7 + x^8$ $1 + x + x^2 + x^3 + x^6 + x^7 + x^8$ $1 + x + x^2 + x^5 + x^6 + x^7 + x^8$ $1 + x^2 + x^4 + x^5 + x^6 + x^7 + x^8$
9	511	48	$1 + x^4 + x^9$(example)
10	1023	60	$1 + x^3 + x^{10}$(example)

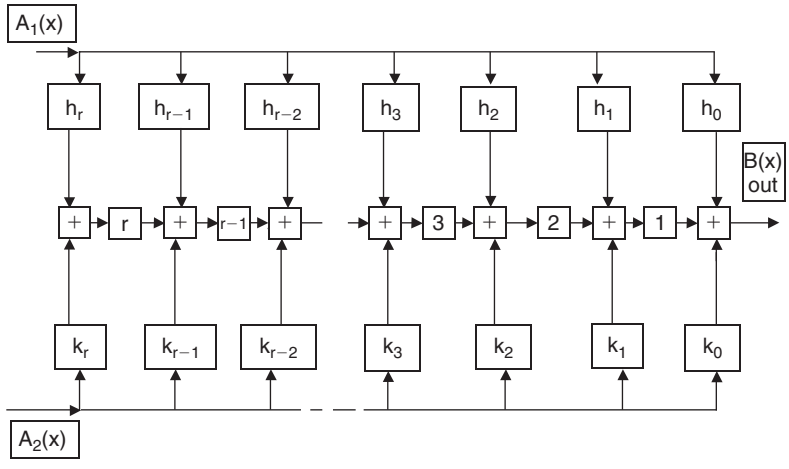


Figure 4.3a *Two-input polynomial multiplier (Peterson et al., 1995).*

Substituting in (4.12):

$$B(x) = A_1(x) \cdot h(x) + B(x) \cdot [g(x) + 1]$$

Add $B(x)[1 + g(x)]$ to both sides of the above expression:

$$B(x) + B(x) + B(x) \cdot g(x) = A_1(x) \cdot h(x) + B(x) \cdot g(x) + B(x) + B(x) + B(x)\, g(x)$$

Therefore, $B(x) \cdot g(x) = A_1(x) \cdot h(x)$

Thus

$$B(x) = A_1(x)\frac{h(x)}{g(x)} \tag{4.13}$$

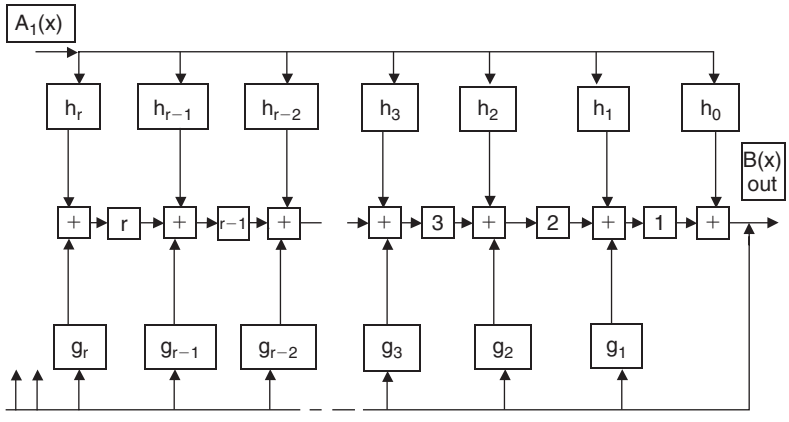


Figure 4.3b *Multiplication by h(x) and division by g(x) (Peterson et al., 1995).*

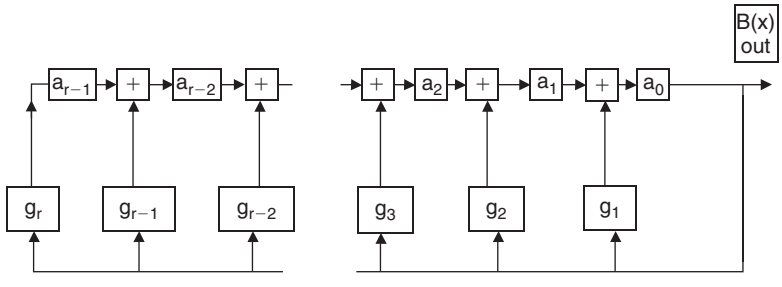


Figure 4.4 *Galois linear feedback shift register sequence generator (Peterson et al., 1995).*

Implementing these changes, Figure 4.3a will be transformed to Figure 4.3b.

Examining (4.13), it is clear that the circuit in Figure 4.3b divides h(x) by g(x) and multiply the result by $A_1(x)$. Generally $A_1(x)$ sets the initial state of the registers contents and can be represented by a finite polynomial given by:

$$A_1(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots\cdots\cdots\cdots\cdots\cdots a_{r-1} \cdot x^{r-1} \tag{4.14}$$

When the registers are loaded with sequence $A_1(x)$, the circuit in Figure 4.3b can be simplified to that shown in Figure 4.4. The loading process takes r time units and while the registers are loading, the output B(x) is zero for these r time units as has been mentioned previously. Therefore, B(x) starting at time r is:

$$B(x) = \frac{A_1(x)}{g(x)} \tag{4.15}$$

There are two methods of constructing LFSR sequence generators for a given generator polynomial (Hanzo et al., 2003). The generator shown in Figure 4.4 follows *Galois feedback implementation* where the output bits are feedback into the shift registers according to

the connection polynomial. The Galois implementation is commonly used when high speed sequences are required since feedback is not delayed. The other implementation is known as *Fibonacci feedback generator* and is shown in Figure 4.5 (see later). The Fibonacci generator can output several delayed versions of the same sequence at the output of each shift register with no additional hardware.

The maximum possible period for an LFSR with the generator connection defined by the polynomial g(x) is computed as follows:

Given the generator polynomial g(x), formulate $x^r \cdot g(x^{-1})$ then find the smallest integer N such that $x^N + 1$ is divisible by $x^r \cdot g(x^{-1})$ where N is the maximum period of the sequence. We now consider an example to explain the calculation.

Example 4.5

Consider the sequence generator shown in Figure 4.4 with the generator polynomial g(x) is given by:

$$g(x) = 1 + x^2 + x^3 + x^4$$

Assume the initial load of the register be 0001.

i. Find the output periodic sequence.
ii. What would the maximum possible period be?

Solution

Using the procedure described in this section, we have:

$$A_1(x) = 1$$

$$B(x) = \frac{A_1(x)}{g(x)} = \frac{1}{1 + x^2 + x^3 + x^4}$$

Using long division, we get:

$$B(x) = 1 + x^2 + x^3 + x^7 + x^9 + x^{10} + x^{14} + \cdots\cdots\cdots\cdots\cdots\cdots$$

The binary sequence at the output of generator shown in Figure 4.4 is:

$$B(x) = 1011000\ \mathbf{1011000}\ 101\ldots\ldots$$

The output B(x) is a periodic of 1011000 with a period N = 7.

We may check the length of the sequence using the minimum integer method described above for which $x^N + 1$ is divisible by $x^r \cdot k(x^{-1})$ as described above.

Now $x^r \cdot g(x^{-1}) = x^4(1 + x^{-2} + x^{-3} + x^{-4}) = x^4 + x^2 + x + 1$

$$\frac{x^N + 1}{x^r \cdot g(x^{-1})} = \frac{x^N + 1}{x^4 + x^2 + x + 1}$$

The minimum integer value for N is 7, that is:

$$\frac{x^N + 1}{x^r \cdot g(x^{-1})} = \frac{x^7 + 1}{x^4 + x^2 + x + 1} = x^3 + x + 1$$

Note that the maximum possible period for the output sequence given by this sequence generator is $2^4 - 1 = 15$.

Example 4.6

Find the sequence at the output of the generator shown in Figure 4.5 with polynomial given by:

$$g(x) = 1 + x^2 + x^3 + x^4$$

Assume the initial load is 0001.

Solution

Consider the initial load 0001 then:

$A_1(x) = 1$ which is the first out of the generator.

Let the equivalent initial load for generator in Figure 4.5 be expressed as

$$A_1'(x) = a_0' + a_1' \cdot x + \cdots\cdots\cdots\cdots + a_{r-1}' \cdot x^{r-1}$$

now $A_1(x) + b_r \cdot x^r + b_{r+1} \cdot x^{r+1} + \cdots\cdots\cdots\cdots = \frac{A_1'(x)}{k(x)}$ Equate the first r-coefficients, we get:

$$A_1(x) \cdot g(x) = a_0' + a_1' \cdots\cdots\cdots\cdots\cdots\cdots + a_{r-1}' \cdot x^{r-1}$$

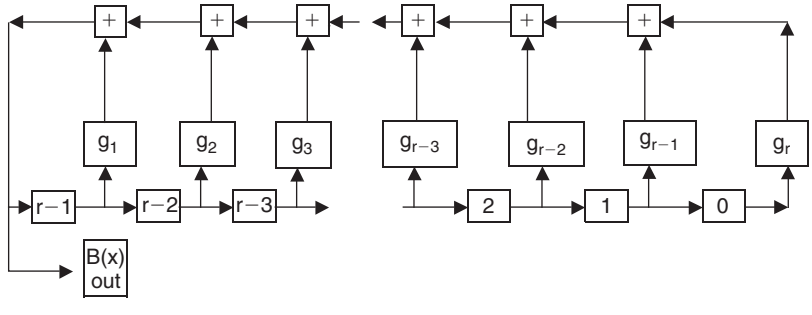


Figure 4.5 *Fibonacci linear feedback shift register sequence generator (Peterson et al., 1995).*

Therefore, $1 + x^2 + x^3 + x^4 = a_0^{'} + a_1' \cdot x + a_2' \cdot x^2 + a_3' \cdot x^3$

So that: $a_0' = 1\ a_1' = 0\ a_2' = 1\ a_3' = 1$

Therefore, $A_1^{'}(x) = 1 + x^2 + x^3$

$$\begin{aligned} B(x) &= \frac{x^{-r} \cdot A_1'(x)}{g(x)} = x^{-4} \cdot \frac{1 + x^2 + x^3}{1 + x^2 + x^3 + x^4} \\ &= x^{-4} \cdot [1 + x^4 + x^6 + x^7 + x^{11} + x^{13} + x^{14} + x^{18} + \cdots\cdots\cdots\ ] \\ &= x^{-4} + 1 + x^2 + x^3 + x^7 + x^9 + x^{10} + x^{14} + \cdots\cdots\cdots \end{aligned}$$

The output polynomial B(x) is the same as that generated from generator shown in Figure 4.4 in the previous example.

4.5.2 Maximal-length sequences (m-sequences)

The m-sequences have found numerous applications in digital communication systems, including spread-spectrum systems. These sequences have a maximum period $N = 2^r - 1$ for an r-stage LFSR generator connected according to a primitive binary polynomial of degree r selected from Table 4.1.

The salient features of the m-sequences are their two-valued autocorrelation functions which are optimal, with the absence of any side-lobe peaks. This is the key parameter which determines the probability of detection and false alarm, during code acquisition and tracking, as we will discover in the material presented in Chapter 5.

The periodic cross-correlation function between any pair of m-sequences of the same period can be relatively large. However, the peak values depend on the sequences chosen and their respective phases. To reduce interference, it is desirable to constrain these peak values to a minimum. All m-sequences of the same length can be derived from each other by a process of proper decimation discussed in Section 4.5.3.

A list of the peak magnitude for the periodic cross-correlation between pairs of m-sequences for $3 \leq r \leq 12$ is shown in Table 4.2.

The m-sequences have the following well-known properties discussed in Sarwate and Pursley's (1980) paper:

i. There are exactly N non-zero sequences representing the N different phases of the m-sequence. If the m-sequence is $\mathbf{x} = (x_0, x_1, x_2, \ldots, x_{N-1})$, then the non-zero sequences are $(x_1, x_2, x_3, \ldots, x_{N-1}, x_0)$, $(x_2, x_3, x_4, \ldots, x_{N-1}, x_0, x_1)$, $(x_3, x_4, x_5, \ldots, x_{N-1}, x_0, x_1, x_2)$, etc.

Table 4.2 *Peak periodic cross-correlation between a pair of m-sequences*

r	$N = 2^r - 1$	Number of m-sequences	Peak cross-correlation
3	7	2	5
4	15	2	9
5	31	6	11
6	63	6	23
7	127	18	41
8	255	16	95
9	511	48	113
10	1023	60	383
11	2047	176	287
12	4095	144	1407

Source: Proakis, 1995

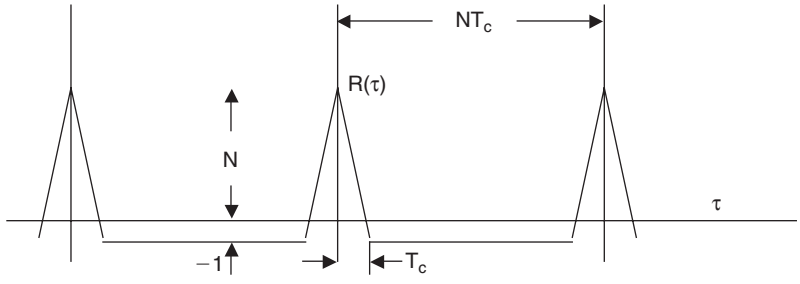


Figure 4.6 *Auto-correlation function for an m-sequence.*

ii. Shift-and-add property of the m-sequences suggests that the modulo-2 sum of an m-sequence and any phase shifted version of itself is another phase of the same m-sequence.

iii. The Hamming weight of an m-sequence is $\left(\frac{N+1}{2}\right)$. This is because the number of ones in an m-sequence is $\left(\frac{N+1}{2}\right)$. The number of zeros is of course $\left(\frac{N-1}{2}\right)$.

iv. The periodic autocorrelation function of an m-sequence is a two-valued function given by

$$\begin{aligned} R(\tau) &= N \quad \text{for } \tau = jN \\ &= -1 \quad \text{for } \tau \neq jN \end{aligned} \tag{4.16}$$

where j is any integer. A plot of the autocorrelation for an m-sequence with chip duration T_c and time period NT_c is shown in Figure 4.6.

v. A run is defined as a set of identical symbols within the m-sequence. The length of the run is equal to the number of these symbols in the run. For any m-sequence generated by r-stage shift registers, it has the following statistics:

1 run of ones of length r

1 run of zeros of length $r - 1$

1 run of ones and one run of zeros of length $r-2$
2 runs of ones and 2 runs of zeros of length $r-3$
4 runs of ones and 4 runs of zeros of length $r-4$
8 runs of ones and 8 runs of zeros of length $r-5$
•
•
•
2^{r-3} runs of ones and 2^{r-3} of zeros of length 1.

For example the m-sequence 000100110101111 contains a total of eight runs as follows: one run of four 1s, one run of three 0s, one run of two 0s, two runs of s single 1, two runs of a single 0.

4.5.3 Decimation of m-sequences

The application of code sequences in spread-spectrum communications necessitates the generation of large sets of codes with highly peaked autocorrelation and minimum cross-correlation. In Section 4.5.1 we considered electronic circuits to generate these sequences. We now look at the generation of number sequences by decimating a single sequence. As usual, we start by defining the basic element involved in the decimation process.

Consider sequence $u = u_0, u_1, u_2, u_3, \ldots$, then sequence v, constructed by taking every q^{th} bit of the nonzero elements of sequence u denoted by u(q) and said to be the *decimation by q of u* where q is a positive integer, that is:

$$v = u(q) \tag{4.17}$$

where $v = u_0,\ u_q,\ u_{2q},\ u_{3q},\ \ldots\ldots\ldots\ldots\ldots$.

If u has a period N and is generated by LFSR with generator connection polynomial h(x) then u(q) has period N_v

$$N_v = \frac{N}{\gcd(N, q)} \tag{4.18}$$

The sequence u(q) can be generated using LFSR with generator polynomial $\hat{h}(x)$ whose roots are the qth powers of the roots of h(x).

The term gcd(N, q) denotes the *greatest common divisor* of N and q. For example, the $\gcd(23, 7) = 1$ because the two numbers can only be divisible by 1, the $\gcd(81, 45) = 9$ since 45 is divisible by 3, 5 and 9 and 81 is divisible by 3 and 9.

The decimation of an m-sequence *may or may not yield another m-sequence*. When the decimation yields an m-sequence, it is called *proper decimation* and if $\gcd(N, q) = 1$, sequence $v = u(q)$ has a period N. Proper decimation guarantees that sequence $v = u(q)$ is an m-sequence and the polynomial $\hat{h}(x)$ is primitive. Clearly, there are N possible sequences that correspond to the N phases of m-sequence u.

The decimation of any phase of sequence u will give a certain phase of v. In general, regardless of which of the m-sequences generated by h(x) we choose to decimate, the result will be an m-sequence generated by $\hat{h}(x)$.

The characteristic sequence $\tilde{u}$ of m-sequence u is such that $\tilde{u} = \tilde{u}(2)$. Since the m-sequence u is periodic, we only need to consider values of less than or equal to $[N-1]$, that is $u[q] = u\,[q \cdot \bmod N]$. When proper decimation is achieved by odd integer q, then $u[2^j q]$ represents different phases of the same m-sequence u(q). Let the m-sequence u be generated by polynomial h(x) such that:

$$h(x) = h_0 x^r + h_1 x^{r-1} + \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots + h_{r-1}x + h_r \tag{4.19}$$

Decimating u by $q = \frac{1}{2}[N-1]$ will generate the reciprocal polynomial of h(x) that is $\hat{h}(x)$ where:

$$\hat{h}(x) = h_r x^r + h_{r-1} x^{r-1} + \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots + h_1 x + h_0 \tag{4.20}$$

Example 4.7

A primitive polynomial h(x) of degree 5, given by the octal number 45, is used to generate an m-sequence u. Decimation of u by 3 generates the m-sequence 75 and decimation by 5 produces the m-sequence 67. Consider every possible decimation in the range $1 \le q \le N-1$, find the m-sequences that can be formulated by each decimation.

Solution

Sequence u, given by polynomial 45, has period $N = 2^5 - 1 = 31$. The decimation of u will then take place for values of q in the range:

$$1 \le q \le 30$$

Sequence u is generated by primitive polynomial $h_0(x)$ given by the octal number 45 and is represented by [100101] in binary so that $h_0(x)$ is given by:

$$h_0(x) = 1 \cdot x^0 + 0 \cdot x^1 + 1 \cdot x^2 + 0 \cdot x^3 + 0 \cdot x^4 + 1 \cdot x^5$$

Therefore

$$h_0(x) = 1 + x^2 + x^5$$

Decimation of u by $2^j q$ where $j \ge 0$ with $q = 1$, that is 1, 2, 4, 8, 16 produces different phases of u.

Decimating the sequence u by 3 ($q = 3$) generates an m-sequence with primitive polynomial $h_3(x)$ given by the octal number 75 which is equivalent to [111101] in binary. Using the same procedure used for representing $h_0(x)$, we can express $h_3(x)$ as:

$$h_3(x) = 1 + x^2 + x^3 + x^4 + x^5$$

Decimating the sequence u by $2^j q$ where $j \geq 0$ and $q = 3$ results in phases of m-sequence given by $h_3(x)$, that is 3, 6, 12, 24, 17. Now decimating the sequence u by 17 is the same as decimating the same sequence by 48 since the sequence is periodic with period 31. Therefore, $48 - N = 17$ where $N = 31$.

Decimating the sequence u by 5 ($q = 5$) generates an m-sequence with primitive polynomial $h_5(x)$ given by the octal number 67 is equivalent to [110111] in binary where:

$$h_5(x) = 1 + x + x^2 + x^4 + x^5$$

Similarly, decimating 5, 10, 20, 9, 18 produces the m-sequence given by polynomial 67. The last two decimations are given by:

$$40 - N = 9 \quad \& \quad 80 - 2N = 18.$$

Consider decimating u by 7. This decimation will generate the same primitive polynomial as decimating by 14, 28, 25, 19. Note that decimation by $25 = 56 - N$ and decimation by $19 = 112 - 3N$ generates the same m-sequences.

Now decimation by 14 is equivalent to decimation by $14 + N = 45$ which is the same as decimating u(3) by 15. Decimation by $15 = \frac{N-1}{2}$ results in an m-sequence generated by the reciprocal of polynomial 75. The octal number 75 is [111101] in binary and the reciprocal polynomial $h_7(x)$ is given by [101111] that is the octal number 57.

$$h_7(x) = 1 + x + x^2 + x^3 + x^5$$

Consider decimating u by $q = 11$. The same primitive polynomial is used when decimation by 22, 13, 26, 21. Now $13 = 44 - N$, $26 = 88 - 2N$, $21 = 176 - 5N$.

Now the decimation by 13 is equivalent to decimating by $13 + 2N = 75$ which is the same as decimating u(5) by 15. Thus the m-sequence is produced by the reciprocal polynomial 67. The octal number $67 = [110111]$ in binary and the reciprocal polynomial is given by [111011] which is 73 in octal number format. Thus, the primitive polynomial $h_{11}(x)$ corresponding to decimation by 11 is:

$$h_{11}(x) = 1 + x + x^3 + x^4 + x^5$$

Lastly, consider decimating u by 15. The same polynomial corresponds to decimation by 30, 29, 27, 23. Note that 29 is equivalent to $60 - N$, 27 is $120 - 3N$ and 23 is $240 - 7N$. The primitive polynomial is the reciprocal polynomial 45. That is, octal number 45 is [100101] in binary and the reciprocal polynomial $h_{15}(x)$ is [101001], which is 51 in octal format.

$$h_{15}(x) = 1 + x^3 + x^5$$

4.5.3.1 Summary of the sequences

The decimation of u generates a total of six m-sequences for primitive polynomials of degree 5. These m-sequences have the following primitive polynomials:

$h_0(x) = 1 + x^2 + x^5$ generates m-sequence u.
$h_3(x) = 1 + x^2 + x^3 + x^4 + x^5$ generates u(3)
$h_5(x) = 1 + x + x^2 + x^4 + x^5$ generates u(5)
$h_7(x) = 1 + x + x^2 + x^3 + x^5$ generates u(7)
$h_{11}(x) = 1 + x + x^3 + x^4 + x^5$ generates u(11)
$h_{15}(x) = 1 + x^3 + x^5$ generates u(15)

4.5.4 Preferred pairs of m-sequences

According to property (iv) in Section 4.5.2, the periodic autocorrelation of m-sequence is a two-valued function. However, the cross-correlation between two m-sequences generated by two different primitive polynomials can be three-valued, four-valued, or possibly many-valued. It is possible to choose a pair of m-sequences which has a three-valued cross-correlation function. These two chosen m-sequences are called the *preferred pair*. The designated pair could be selected as the m-sequence u and its decimated version $v = u(q)$ using the decimation process discussed in Example 4.7. The preferred pairs that have period $N (= 2^r - 1)$ must satisfy the following conditions:

i. $r \neq 0 \bmod 4$, that is n is odd or $r = 2 \bmod 4$ (4.21)
Where n is the degree of the primitive polynomial and r could not take on such values as: 4, 8, 12, 16, 20, …. That is, $r = 2, 6, 10, 14, 18, \ldots$, etc. These values of r give odd values for N (=3, 63, 1023, 16383, 262143, . . . , etc.).

ii. $v = u(q)$ q is odd given by:

$$q = \begin{matrix} 2^k + 1 \\ \text{or} \\ 2^{2k} - 2^k + 1 \end{matrix} \tag{4.22}$$

where k is given by property (iii).

iii. $$\gcd(r, k) = \begin{matrix} 1 & \text{for n odd} \\ 2 & \text{for } r = 2 \bmod 4 \end{matrix} \tag{4.23}$$

We have shown in Section 4.5.3 how to find the gcd of two numbers. It is clear that because $r \neq 0 \bmod 4$, N is not a power of 2. Typical values for k are (1, 2). These values of k make q = 3, 5, 13. The preferred pairs of m-sequences have three-valued cross-correlation function defined as $[-1, -t(r), t(r)-2]$ where

$$t(r) = 1 + 2^{\frac{r+1}{2}} \text{ for r odd} \tag{4.24}$$

$$t(r) = 1 + 2^{\frac{r+2}{2}} \text{ for } r = 2 \bmod 4 \tag{4.25}$$

Table 4.3 *Maximum cross-correlation associated with preferred pair of m-sequences*

r	1	2	3	5	6	7	10
t(r)	3(4.24)	5 (4.25)	5 (4.24)	9 (4.24)	15(4.25)	15(4.24)	64(4.25)
Cross-correlation	−1, −3, 1	−1, −5, 3	−1, −5, 3	−1, −9, 7	−1, −15, 13	−1, −15, 13	−1, −64, 63

Let us compute typical values of the cross-correlation for an assumed m-sequences with $r = 1, 2, 3, 5, 6, 7$, and 10. Using (4.24) and (4.25), t(r) and maximum cross-correlations are given in Table 4.3.

A collection of m-sequences where the property of each pair in the set is a preferred pair is called a *connected set*. The largest possible connected set is called *Maximal connected set*. The size of this set, M_n, is important in applications such as multiple users' spread-spectrum systems.

Example 4.8

Consider the m-sequence u generated by a primitive polynomial of degree $n = 5$ as given in Example 4.7. Construct the maximal connected set of preferred pairs of m-sequences produced by the decimation of u. What is the size of this set?

Solution

A preferred pair of m-sequences must satisfy conditions i, ii, iii as stated in Section 4.5.4. Condition (i) is being satisfied since n in this example is odd.

Consider the pair [u, u(3)] , where $q = 3$ (odd) and $k = 1$ so that $\gcd(r,k) = 1$.

Therefore [u, u(3)] are a preferred pair.

The pair [u, u(5)] gives $q = 5$ (odd) and $k = 2$ so that $\gcd(r,k) = 1$, therefore, [u, u(5)] is another preferred pair.

The m-sequence u(5) is another phase for m-sequence u(9) as we proved in Example 4.7. The pair [u(3), u(5)] produces the same cross-correlation as the pair [u(3), u(9)] so that [u(3), u(5)] is a preferred pair of m-sequences.

Following a similar procedure, we can show that the following pairs of m-sequences are preferred pairs [u(3), u(5)], [u(5), u(15)] and [u(15), u(3)].

Considering all of the possible pairs, we can present a graphical plot with each preferred pair connected by a line as shown in Figure 4.7.

It can be seen from the plot that $M_5 = 3$. Considering all of the m-sequences generated by primitive polynomials of degree n, where $3 \leq n \leq 16$, the size of the maximal connected

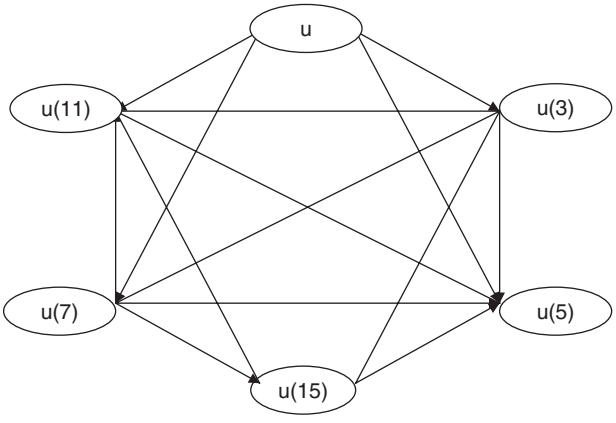


Figure 4.7 *Set of preferred pairs of m-sequence.*

Table 4.4 *Maximum connected size of m-sequences*

Number of SR stages r	Number of m-sequences	Maximal connected set size M_n
3	2	2
4	2	0
5	**6**	**3**
6	6	2
7	18	6
8	16	0
9	48	2
10	60	3
11	176	4
12	144	0
13	630	4
14	756	3
15	1800	2
16	2048	0

set is a small fraction of the number of m-sequences as given by Fan and Darnell (1995) (see Table 4.4).

4.5.5 Gold sequences

If [u, v] is any preferred pair of m-sequences generated by primitive polynomials h(x) and $\hat{h}(x)$ and each of degree n and period $N = 2^n - 1$, then a set of Gold sequences G[u, v] can be generated by $u \oplus v$ where $\oplus$ represents module-2 addition. Taking into consideration

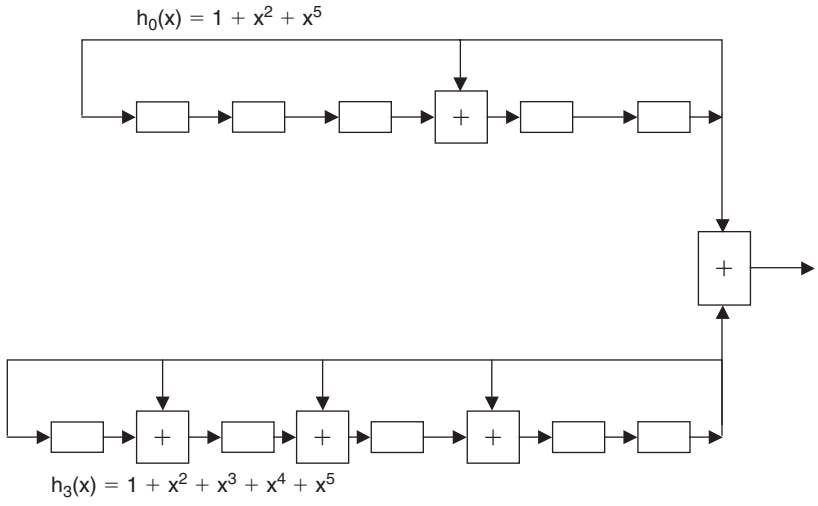


Figure 4.8 *Block diagram of Gold generator.*

the N possible phases of the sequences, we can define the set G[u, v] as:

$$u, v, u \oplus v, u \oplus Tv, u \oplus T^2 v, u \oplus T^3 v, \ldots\ldots\ldots\ldots\ldots\ldots\ldots, u \oplus T^{N-1} v \tag{4.26}$$

Where $T^i v$ represents m-sequence v phase shifted by i symbols with $i = 0, 1, 2, \ldots, N-1$.

The Gold set of sequences contains $N + 2$ sequences and is generated by polynomial given by $h(x)\hat{h}(x)$. A typical Gold generator, shown in Figure 4.8, can be constructed using the preferred pair of m-sequences u[u, u(3)] from Example 4.7 where:

$$h_0(x) = 1 + x^2 + x^5 \qquad \text{gives m-sequence u.}$$
$$h_3(x) = 1 + x^2 + x^3 + x^4 + x^5 \qquad \text{gives u(3)}$$

Considering the set of Gold sequences, the out-of-phase autocorrelation of any sequence in the set and the cross-correlation between any pair in the set have three-valued correlation functions given by $[-1, -t(r), t(r) - 2]$ as mentioned previously. However, Gold sequences have cross-correlation (-1) for many offsets of the preferred pair of m-sequence. It turns out that attaching '0' to the original Gold sequences will eliminate the cross-correlation. In fact, simple zero-padding to the Gold sequences can originate 2^r code sequences which have *zero cross-correlation* between them. These code sequences are called '*orthogonal Gold codes*'.

It should be noted that the literature presents an earlier definition for the set of Gold sequences as G[u, v] where $v = u[t(r)]$. At present, it has been accepted that u and v should be any preferred pair of m-sequences.

The lower bound on the peak cross-correlation (Φ_{max}) between any pair of binary sequences of period N in a set of M sequences is given by Welch bounds (Welsh, 1974) as:

$$\Phi_{max} \geq N\sqrt{\frac{M-1}{NM-1}} \tag{4.27}$$

For large values of N and M, Φ_{max} can be approximated as:

$$\Phi_{max} \approx \sqrt{N} \tag{4.28}$$

This lower bound is commonly taken as a bench mark for the cross-correlation between a set of binary sequences when computing the multiple access interference.

For a Gold sequence with a reasonably large value for m,

$$N = 2^m - 1 \approx 2^m \tag{4.29}$$

Thus the lower bound on Gold sequences Φ_{max} is:

$$\Phi_{max} \approx 2^{\frac{m}{2}} \tag{4.30}$$

The maximum cross-correlation between the preferred sequences of a Gold sequence is:

$$2^{\frac{m+1}{2}} + 1 \text{ m is odd, i.e. } \sqrt{2} \text{ times lower bound} \tag{4.31}$$

$$2^{\frac{m+2}{2}} + 1 \text{ m is even, i.e. 2 times lower bound} \tag{4.32}$$

Example 4.9

Four sequences, 7-bits each, are all chosen from a set of Gold sequences. Compute the cross-correlation between each pair and suggest a format that eliminates this cross-correlation. Assume the zero time shift between the sequences.

$G_1 = [01\ 01\ 00\ 0]$
$G_2 = [00\ 10\ 01\ 0]$
$G_3 = [10\ 00\ 00\ 1]$
$G_4 = [11\ 11\ 01\ 1]$

Solution

Using the expressions for cross-correlation function derived in Section 4.5.5, we now replace logic '0' with '−1' and the cross-correlations are given below:

$$R_{1,2}(0) = -1, \quad R_{1,3}(0) = -1, \quad R_{1,4}(0) = -1$$
$$R_{2,3}(0) = -1, \quad R_{2,4}(0) = -1$$
$$R_{3,4}(0) = -1$$

Now let us zero-pad the sequences, we get the following sequences:

$G_1 = [01\ 01\ 00\ 00]$
$G_2 = [00\ 10\ 01\ 00]$
$G_3 = [10\ 00\ 00\ 10]$
$G_4 = [11\ 11\ 01\ 10]$

It can be shown that the cross-correlation between any pair of these sequences is zero.

4.5.6 Kasami sequences

A set of Kasami sequences can be generated using two different procedures described below:

i. Generating a small set of Kasami sequences.
Starting with the m-sequence u generated by a primitive polynomial $h_u(x)$ with period $N = 2^n - 1$ where n is an even number, we can generate a sequence v using primitive polynomial $h_v(x)$ by decimating u by $2^{\frac{n}{2}} + 1$; that is:

$$v = u(2^{\frac{n}{2}} + 1) \tag{4.33}$$

It has been proven (Fan and Darnell, 1995) that v is an m-sequence with period derived as follows:

$$\text{period} = \frac{N}{\gcd[N, (2^{\frac{n}{2}} + 1)]} \tag{4.34}$$

$$= \frac{2^n - 1}{\gcd[(2^n - 1), (2^{\frac{n}{2}} + 1)]} = \frac{(2^{\frac{n}{2}} - 1)(2^{\frac{n}{2}} + 1)}{\gcd[(2^{\frac{n}{2}} - 1)(2^{\frac{n}{2}} + 1), (2^{\frac{n}{2}} + 1)]}$$

$$\text{Period} = N_v = \frac{(2^{\frac{n}{2}} - 1)(2^{\frac{n}{2}} + 1)}{(2^{\frac{n}{2}} + 1)} = 2^{\frac{n}{2}} - 1 \tag{4.35}$$

The small set of Kasami sequences is generated by the primitive polynomial $h(x) = h_u(x)\ h_v(x)$ using a module addition of u with all possible phases of v; that is:

$$\{u, u \oplus v, u \oplus Tv, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, u \oplus T^{N_v}v\} \tag{4.36}$$

The set contains $2^{\frac{n}{2}}$ sequences, each of period N and with three-valued correlation function $[-1, -s(n), s(n)-2]$ where

$$s(n) = 2^{\frac{n}{2}} + 1 \tag{4.37}$$

The maximum magnitude of correlation acquired is s(n) and it is approximately one half of the maximum magnitude value achieved by Gold set.

Table 4.5 *Comparison between Kasami and Gold sequences*

	Small set of Kasami	Large set of Kasami	Gold
Period of individual sequence	$2^n - 1$	$2^n - 1$	$2^n - 1$
Size of set	$2^{\frac{n}{2}}$	$2^{\frac{n}{2}}(2^n + 1)$	$2^n + 1$
Values of n	even	2 mod 4 or 0 mod 4	odd or 2 mod 4
Max correlation between any pair	$2^{\frac{n}{2}} + 1$	$2^{\frac{n+2}{2}} + 1$	$2^{\frac{n+2}{2}} + 1$

ii. Generating a large set of Kasami sequences.
Consider the following m-sequences: sequence u is generated by primitive polynomial $h_u(x)$ of degree n and has a period N; sequence v is the decimation of u by s(n), i.e. $v = u[s(n)]$ generated by the primitive polynomial $h_v(x)$of degree $\frac{n}{2}$ and has period $2^{\frac{n}{2}} - 1$ and $w = u[t(r)]$ generated by a polynomial $h_w(x)$ of degree n with period N where t(r) is given by (4.24) and (4.25) where n is even. Then the large set of Kasami sequences $K_L(u)$ is generated by primitive polynomial $h(x) = h_u(x)\ h_v(x)\ h_w(x)$ and is given by:

$$K_L(u) = u \oplus v \oplus w \tag{4.38}$$

and has a period $N = 2^n - 1$. The size of $K_L(u)$ is $2^{\frac{n}{2}}(2^n + 1)$ for $n \equiv 2 \bmod 4$, and $2^{\frac{n}{2}}(2^n + 1) - 1$ for $n \equiv 0 \bmod 4$.

The correlation function for $K_L(u)$ is many-valued with values chosen from the set $\{-1, -t(r), t(r) - 2, -s(n), s(n) - 2\}$. The maximum magnitude of correlation is t(r).

It is interesting to compare the Kasami sequences with the Gold sequences and such a comparison is given in Table 4.5.
For example, for $n = 6$ (i.e. 6-stage LFSR generator), the length of Kasami sequences is 63 bits, the size of the small set is 8 sequences; the size of the large set is 520 sequences and the size of Gold set for the 6-stage LFSR generator is 65. For the same 6-stage LFSR generator, the maximum magnitudes of the cross-correlation between these sequences are as follows: 9 for the Kasami small set, 17 for the Kasami large set, and 17 for the Gold set.

4.5.7 Walsh sequences

Walsh code sequences are obtained from the Hadamard matrix which is a square matrix where each row in the matrix is orthogonal to all other rows, and each column in the matrix is orthogonal to all other columns. The Hadamard matrix H_n is generated by starting with zero matrix and applying the Hadamard transform successively. Each column or row in the Hadamard matrix corresponds to a Walsh code sequence of length n. Orthogonality between codes in the Hadamard matrix is defined such that the cross-correlation values, associated with zero offset between the pair of sequences, is zero.

The Hadamard transform is defined as

$$\mathbf{H}_n = [0] \tag{4.39}$$

$$\mathbf{H}_{2n} = \begin{array}{|c|c|}\hline \mathbf{H}_n & \mathbf{H}_n \\ \hline \mathbf{H}_n & \overline{\mathbf{H}}_n \\ \hline \end{array} \tag{4.40}$$

Thus, n = 1 and we get:

$$\mathbf{H}_2 = \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \tag{4.41}$$

Repeating the Hadamard transform again for n = 2, we get $\mathbf{H}_4$ as:

$$\mathbf{H}_4 = \begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 \\ \hline 0 & 1 & 1 & 0 \\ \hline \end{array} \tag{4.42}$$

Repeating the Hadamard transform again for n = 4, we get:

$$\mathbf{H}_8 = \begin{array}{|c|c|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ \hline \end{array} \tag{4.43}$$

Matrix (4.43) consists of 8 Walsh codes, each of length 8 bits. It is worth noting that in matrices (n = 2), (n = 4), (n = 8), *and it is true for all Hadamard matrices*, the first column (or row) has all zero sequences and the second column (or row) has alternating sequences of '0' and '1'. The interesting property of the matrix is that any column (or row) differs from any other column (or row) in exactly $\frac{N}{2}$ positions. Hardware implementation is shown in Figure 4.9 comprising of a clock, three AND gates, and two toggle flip flops and OR gate.

The edge-triggered Toggle Flip Flop (T-FF) is a standard JK flip flip, with both J and K connected to high, and each stage divides the input clock by 2. The input to the generator is eight bits from the clock 01010101, so the output from the first T-FF is 00110011 and from the second T-FF is 00001111. The binary variables u_2 u_1 u_0 represent a Walsh code

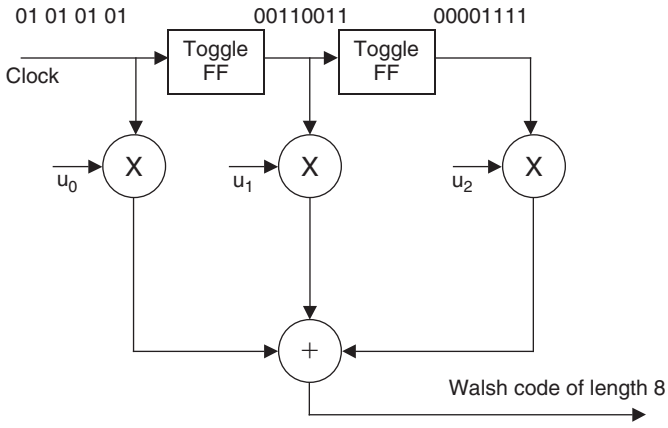


Figure 4.9 *Walsh code generator for Walsh code of length 8 (Hanzo et al., 2003).*

index as shown in this table below:

$u_2\, u_1\, u_0$	Walsh code
000	00000000
001	01010101
010	00110011
011	01100110
100	00001111
101	01011010
110	00111100
111	01101001

(4.44)

It can be seen that to generate a Walsh code of length 16 bits, we need to use three T-FFs, four ANDs and one OR gate. In general, for a Walsh code of length n, we use n–1 toggled flip flips, n AND gates and one multi-input OR gate.

4.5.8 Multi-rate orthogonal codes

In applications such as data transmission using CDMA spread-spectrum systems, there is a need to use orthogonal codes that enable variable data rate transmission. Consequently, for constant chip rate, the spreading factor (i.e. spreading gain) has to be varied. These codes are described in the literature as multi-rate orthogonal codes. An orthogonal code generator is shown in Figure 4.10 and comprises of a Walsh code generator, similar to the one shown in Figure 4.9, and the orthogonal Gold code generator.

In Figure 4.10, R_c is the chip rate for the orthogonal Gold code generator and L_g is the length of the orthogonal Gold code, $\frac{R_c}{L_g}$ is a chip rate for the Walsh code generator.

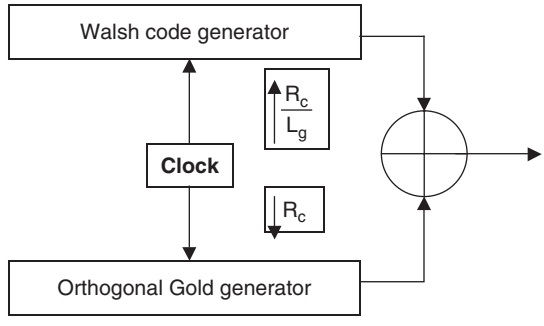


Figure 4.10 *Multi-rate orthogonal code generator (Hanzo et al., 2003).*

The maximum length of the Walsh code L_w is given by:

$$L_w = \frac{G_{p\,max}}{L_g}$$

where $G_{p\,max} = \frac{R_c}{R_b'}$ and R_b' is the lowest bit rate used in the transmission. The Walsh generator is clocked once every L_g Gold chips. Walsh codes with indices selected between 0 to $L_w - 1$ can be used for data transmission at the rate R_b'. However, for data rate $2^k R_b'$, Walsh codes have to be selected carefully to guarantee the orthogonality between the multi-rate codes, and this is better explained by considering the following example.

Example 4.10

Consider the multi-rate orthogonal Gold code generator shown in Figure 4.10 and let $R_c = 128$ kc/s and $R_b' = 8$ kb/s and $R_b = 16$ kb/s. Let the length of the orthogonal Gold code $L_g = 8$.

1. Find the code matrix at the generator output.
2. Find the codes that are not orthogonal.

Solution

The maximum spreading factor (spreading gain) $= \dfrac{R_c}{R_b'} = \dfrac{128}{8} = 16$

Walsh code length $L_w = \dfrac{\text{Max. Spreading factor}}{L_g} = \dfrac{16}{8} = 2$

Thus $$\mathbf{H}_2 = \begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 1 \\ \hline \end{array}$$

Therefore, the length of each of the codes generated from the multi-rate code is $L_w L_g = 2 \times 8 = 16$ and the multi-rate code matrix M_{16} is a square-shaped (16×16) matrix given by:

$$\mathbf{M}_{16} = \begin{array}{|c|c|}\hline \mathbf{G}_8 & \mathbf{G}_8 \\ \hline \mathbf{G}_8 & \bar{\mathbf{G}}_8 \\ \hline \end{array}$$

Substituting for $\mathbf{G}_8$, we get:

$$\mathbf{M}_{16} = \begin{array}{|cccccccccccccccc|}
\hline
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
\hline
\end{array}$$

Consider sequences $\mathbf{M}_{16}(1)$ and $\mathbf{M}_{16}(9)$ to work out the cross-correlation associated with them given as:

0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0
0	1	0	1	0	0	0	0	1	0	1	0	1	1	1	1
1	+1	+1	+1	+1	+1	+1	+1	−1	−1	−1	−1	−1	−1	−1	−1

Clearly, this cross-correlation function has an amplitude zero for zero offset between the sequences indicating these two codes are orthogonal. Similarly, we can show each row in $\mathbf{M}_{16}$ is orthogonal to other rows.

When data rate is doubled (i.e. $R_b = 16$ kb/s), the number of chips per data bit is halved which makes codes $\mathbf{M}_{16}(i)$ and $\mathbf{M}_{16}(i+8)$ not orthogonal where $i = 0, 1, \ldots, 7$ since both are expanded from $\mathbf{G}_8(i)$. Thus for doubling the rate, the number of orthogonal sequences available is halved. Consequently, for transmission at double bit rate, two independent orthogonal codes have to be used.

4.6 Complex sequences

Although only binary code sequences are used in CDMA spread-spectrum systems (IS-95) standards, research carried out on systems using complex codes shows improved bit error rates compared with those using binary Gold codes. Consequently, future wireless systems may benefit from using complex code sequences, although there are challenges in terms

of finding large size code sequences with correlation properties suitable for multi-user applications.

4.6.1 Quadriphase sequences

The elements of the quadriphase sequences correspond to the complex qth root of unity and a large family of these sequences can be generated as we will show in this section. Each quadriphase sequence has q complex elements. Such sequences match the M-level phase modulation schemes used in spread-spectrum systems so that the carrier magnitude and phase are modulated by complex valued code sequence.

In its basic form, the quadriphase sequence can be acquired using any two binary sequences $\{a_n\}$ and $\{b_n\}$, each of period N and which are combined into a quadriphase sequence $\{c_n\}$ of period N given by:

$$c_n = \frac{1}{2}(1 + j) \cdot a_n + \frac{1}{2}(1 - j) \cdot b_n \tag{4.45}$$

Conversely, any quadriphase sequence of length N can be decomposed into two binary sequences, each of length N. The maximum cross-correlation of the quadriphase set is usually lower than the maximum cross-correlation between constituent binary sequences by $\sqrt{2}$ which corresponds to a 3 dB improvement in signal to interference ratio (Fan and Darnell, 1995).

Quadriphase sequences, which can be used as signature sequences in multi-user spread-spectrum systems, have been studied extensively in the literature. Given the size of the literature on this topic, it is not necessary to cover all of the work on quadriphase sequences in this section. Therefore, with apology to researchers whose work is not discussed here, we proceed with selected samples of papers published on this topic and that cover properties such as sequence length, family size and correlation functions suitable for application in spread-spectrum systems. We start with Krone and Sarwate (1984) who have presented methods of construction that obtain sets of quadriphase sequences from sets of binary sequences. A set of $2(q + 1)$ sequences can be constructed, each with period $N = (q - 1)$ and maximum periodic cross-correlation and periodic out-of-phase autocorrelation bounded by $3\sqrt{q} + 5$. Implementation of these methods is not discussed in Krone and Sarwate (1984) and the performance of spread-spectrum systems using the quadriphase code sequences is an open problem.

Kumar et al. (1996) presented design methods for large families of quadriphase sequences with low correlation, and three sets of families of quadriphase sequences are proposed. Each sequence in each family has length N given by:

$$N = 2^r - 1 \tag{4.46}$$

The sequences can be generated using two multistage LFSR generators – one is binary (mod 2 arithmetic) and the other is quaternary (mod 4 arithmetic). The family size M, the maximum periodic correlation C_{max}, for family sets S(0), S(1), and S(2) are given in Table 4.6.

Consider family set S(1) with $r = 5$ (number of shift registers), the length of each sequence $N = 31$ and the family size $M = 31^2 = 3 \times 31 + 2 = 1056$. The maximum magnitude of the periodic correlation $C_{max} = 12.31$.

The following primitive polynomial of degree 5 is used for the connection of the binary LFSR

$$h(x) = 1 + x^2 + x^3 + x^4 + x^5 \tag{4.47}$$

Sequences in the family set S(1) can be generated using the LFSR shown in Figure 4.11.

The desired quadriphase sequence $\{\hat{a}_n\}$ is obtained by the mapping relation:

$$\hat{a}_n = j^{a_n} \quad j = \sqrt{-1} \tag{4.48}$$

Table 4.6 *Family size of polyphase sequences (Kumar et al., 1996)*

Family set	Family size M	C_{max}
S(0)	$N + 2$	$\sqrt{N+1}+1$
S(1)	$\geq N^2 + 3N + 2$	$2\sqrt{N+1}+1$
S(2)	$\geq N^3 + 4N^2 + 5N + 2$	$4\sqrt{N+1}+1$

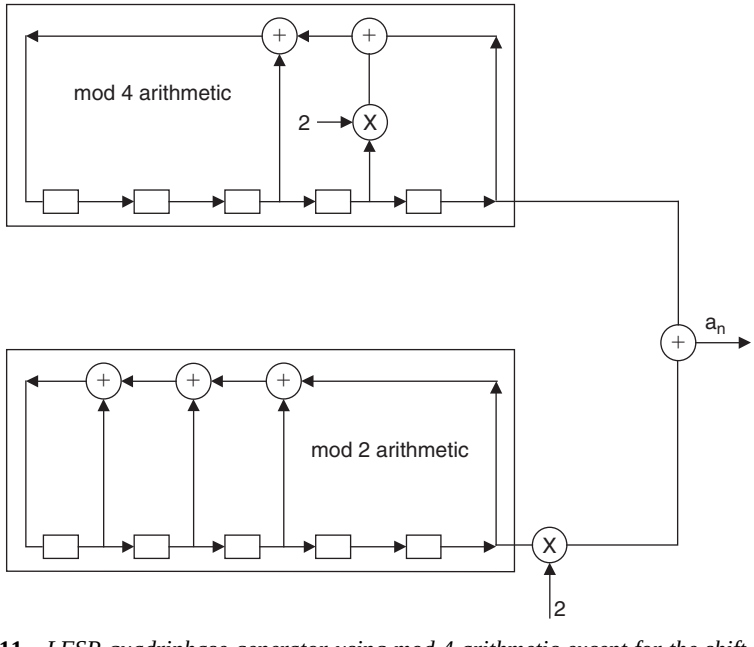


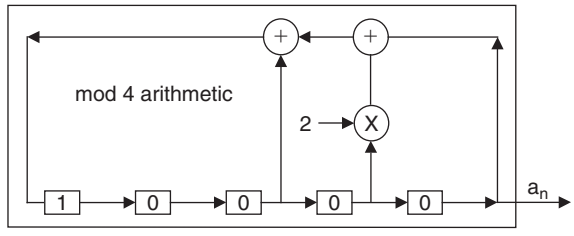
Figure 4.11 *LFSR quadriphase generator using mod 4 arithmetic except for the shift register at the bottom which is implemented using mod 2 arithmetic (Kumar et al., 1996).*

Example 4.11

Consider the quadriphase generator in Figure 4.11 with M = 5. The shift registers are loaded with the following initial states: $(a_{n-1}, a_{n-2}, a_{n-3}, a_{n-4}, a_{n-5}) = (1,0,0,0,0)$. Generate the quadriphase sequences.

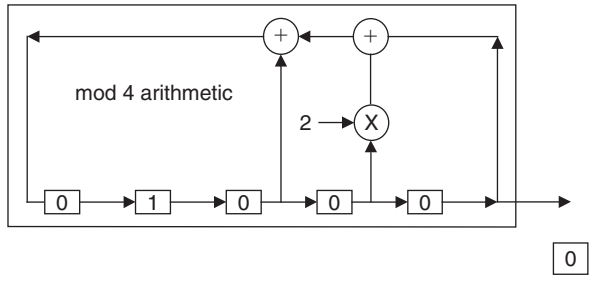
Solution

Consider the mod 4 arithmetic generator with initial states 10000 shown below:

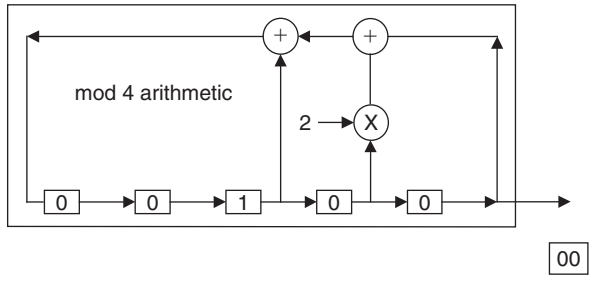


Clearly each register is a 2-bit register with a clock twice as fast as the binary register and all computations are mod 4 arithmetic. We will consider the binary LFSR generator clock as a reference clock. Let us compute the mod 4 output sequence.

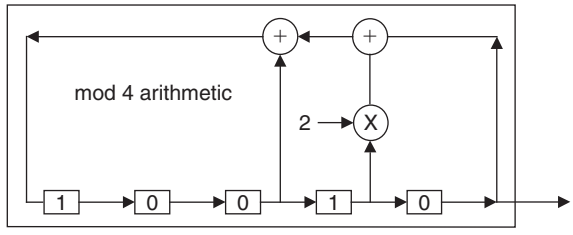
At the first clock pulsed, the generator output '0' and the generator states change to:



At the 2nd clock pulse, the generator output '0' and its states change to:

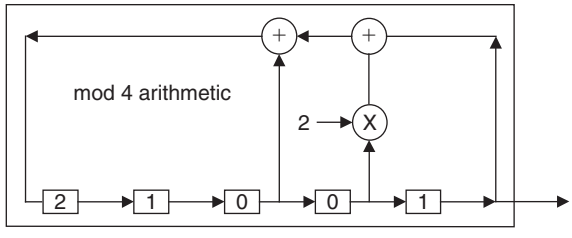


At the 3rd clock pulse, the generator output ‘0’ and its states change to:



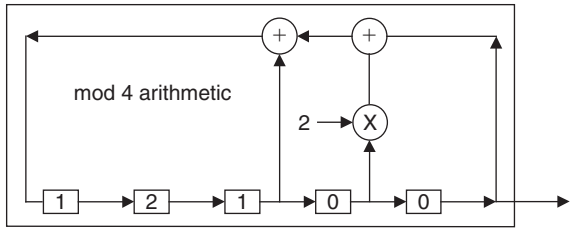
000

At the 4th clock pulse, the generator output ‘0’ and its states change to:



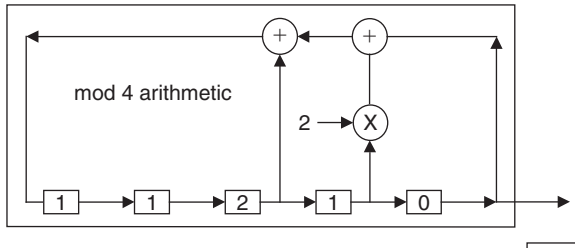
0000

At the 5th clock pulse, the generator output ‘1’ and its states change to:



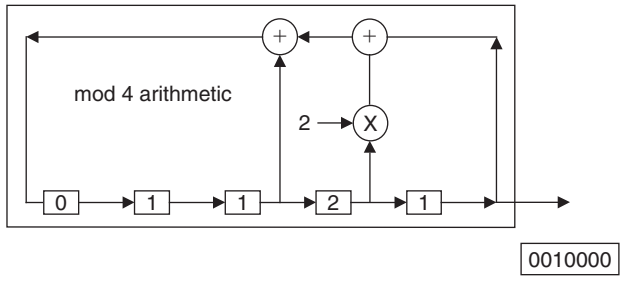
10000

At the 6th clock pulse, the generator output ‘0’ and its states change to:

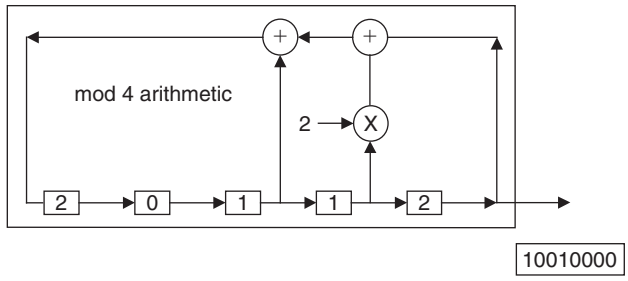


010000

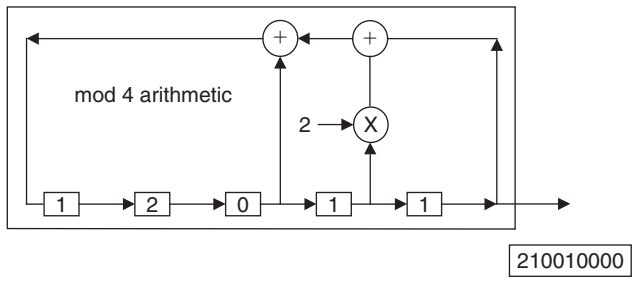
At the 7th clock pulse, the generator output ‘0’ and its states change to:



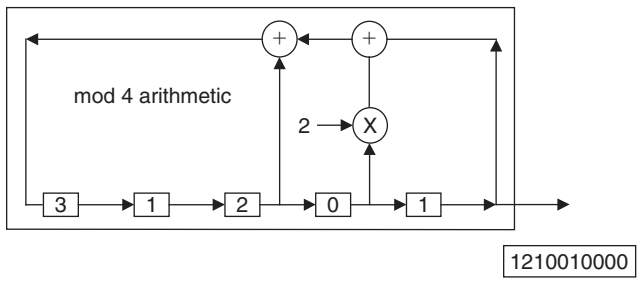
At the 8th clock pulse, the generator output ‘1’ and its states change to:



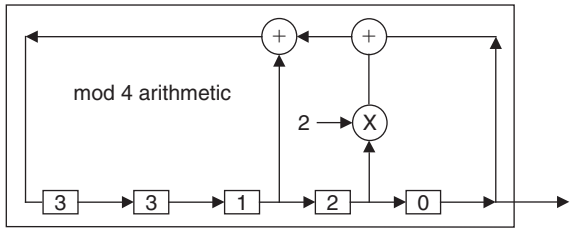
At the 9th clock pulse, the generator output ‘2’ and its states change to:



At the 10th clock pulse, the generator output ‘1’ and its states change to:

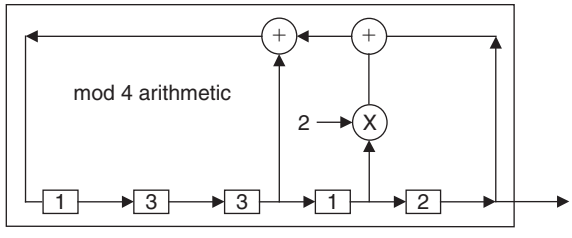


At the 11th clock pulse, the generator output '1' and its states change to:



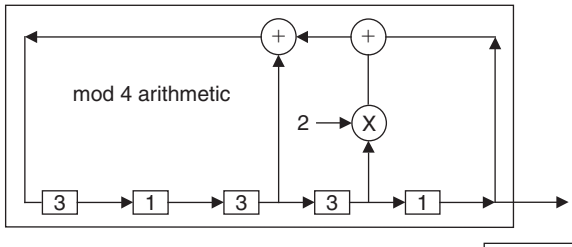
11210010000

At the 12th clock pulse, the generator output '0' and its states change to:



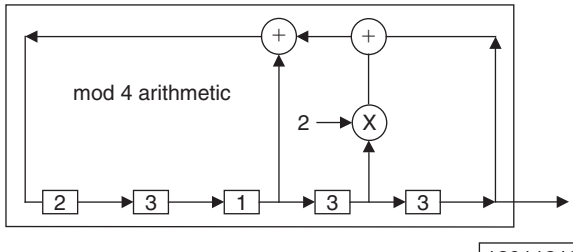
011210010000

At the 13th clock pulse, the generator output '2' and its states change to:



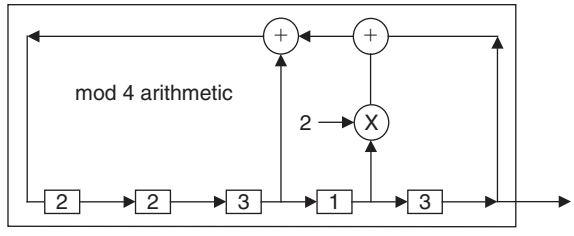
2011210010000

At the 14th clock pulse, the generator output '1' and its states change to:



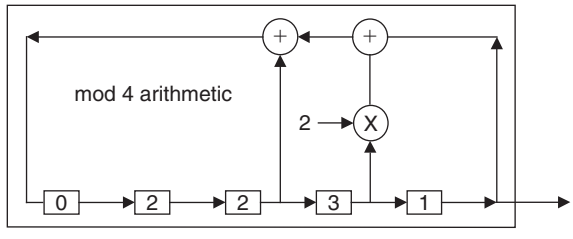
12011210010000

At the 15th clock pulse, the generator output '3' and its states change to:



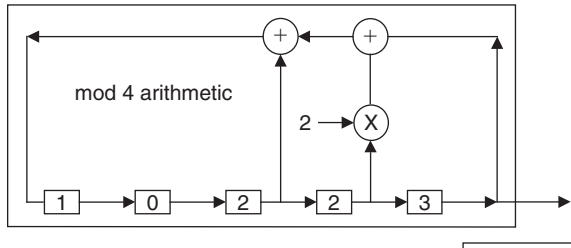
3120112100100000

At the 16th clock pulse, the generator output '3' and its states change to:



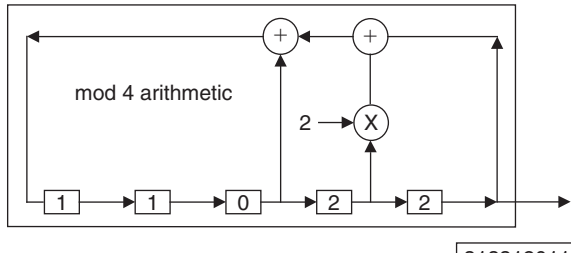
33120112100100000

At the 17th clock pulse, the generator output '1' and its states change to:



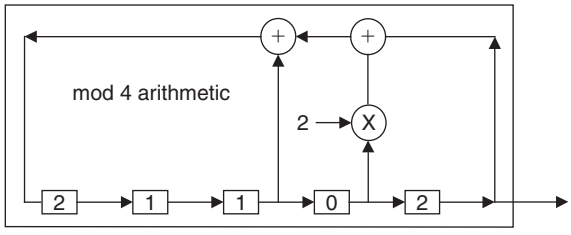
133120112100100000

At the 18th clock pulse, the generator output '3' and its states change to:



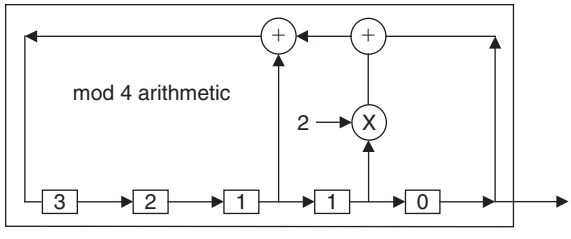
3133120112100100000

At the 19th clock pulse, the generator output '2' and its states change to:



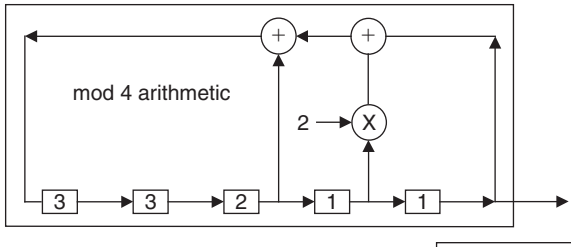
2313312011210010000

At the 20th clock pulse, the generator output '2' and its states change to:



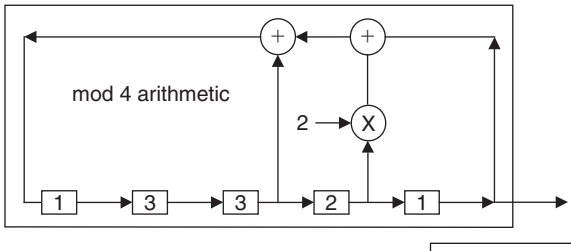
22313312011210010000

At the 21st clock pulse, the generator output '0' and its states change to:



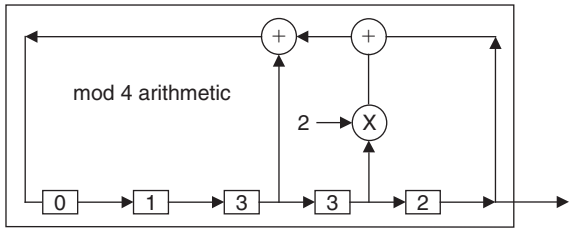
022313312011210010000

At the 22nd clock pulse, the generator output '1' and its states change to:



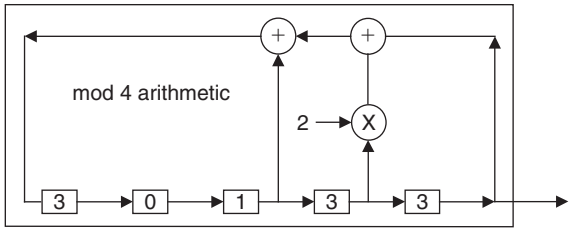
1022313312011210010000

At the 23rd clock pulse, the generator output ‘1’ and its states change to:



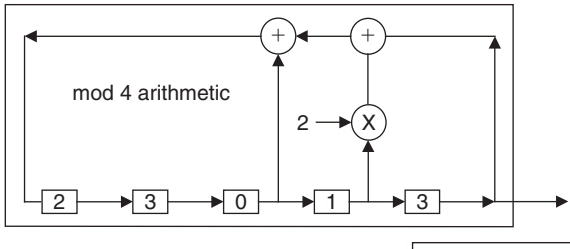
1102231331201121001000O

At the 24th clock pulse, the generator output ‘2’ and its states change to:



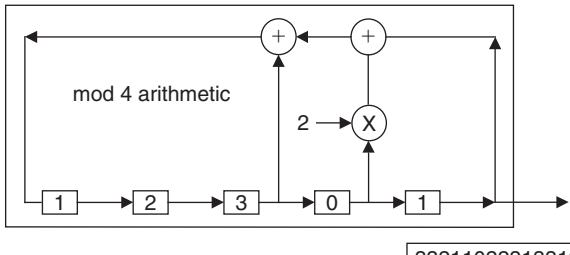
2110223133120112100100O0

At the 25th clock pulse, the generator output ‘3’ and its states change to:



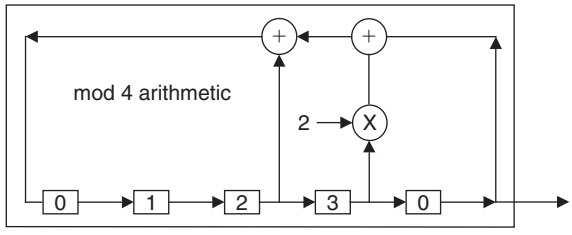
32110223133120112100100O0

At the 26th clock pulse, the generator output ‘3’ and its states change to:



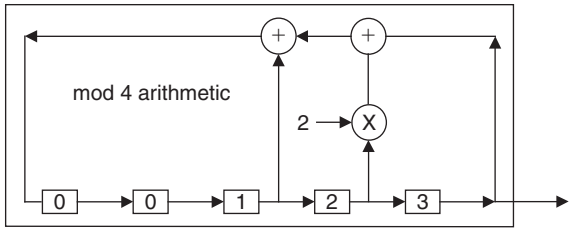
332110223133120112100100O0

At the 27th clock pulse, the generator output ‘1’ and its states change to:



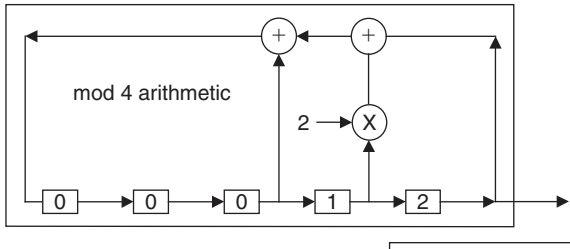
133211022313312011210010000

At the 28th clock pulse, the generator output ‘0’ and its states change to:



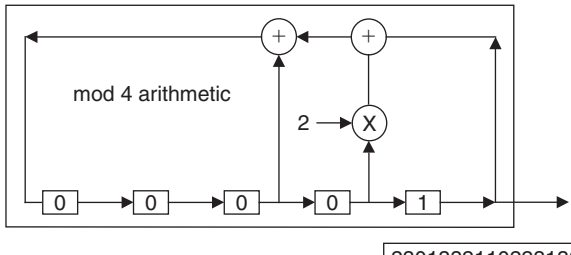
013321102231331201121001000 0

At the 29th clock pulse, the generator output ‘3’ and its states change to:



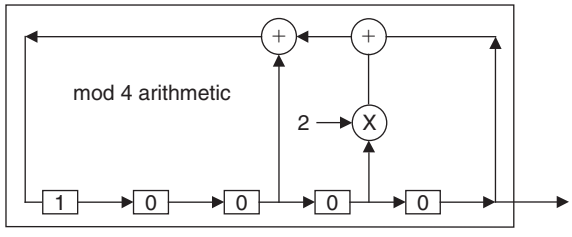
301332110223133120112100100 00

At the 30th clock pulse, the generator output ‘2’ and its states change to:



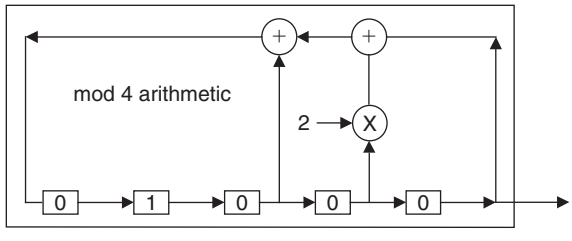
230133211022313312011210010000

At the 31st clock pulse, the generator output '1' and its states change to:



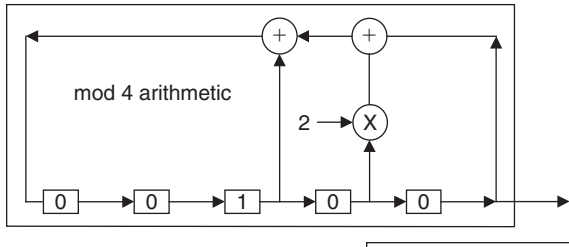
1230133211022313312011210010000

At the 32nd clock pulse, the generator output '0' and its states change to:



01230133211022313312011210010000

At the 33rd clock pulse, the generator output '0' and its states change to:



001**230133211022313312011210010000**

Similarly, it can be shown that the output is:

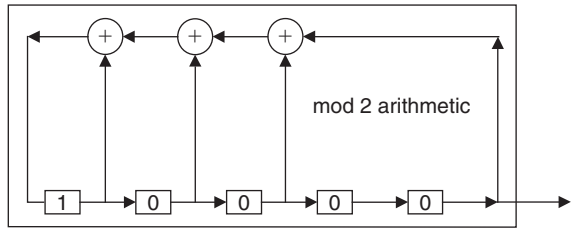
– – – – – – – – – – – – – – – –0010000**123013321102231331201121001000 0**

Consequently, the output periodic mod 4 sequence is of length (period) 31:

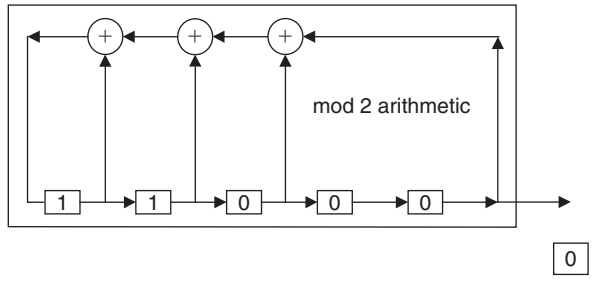
1230133211022313312011210010000

Now consider the binary LFSR generator in Figure 4.11. The shift registers are initialized with 10000 as shown below. The multiplication of the output binary sequence by 2

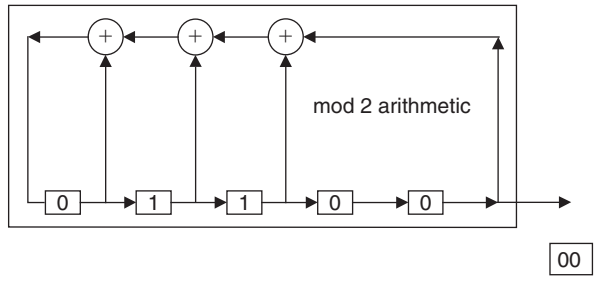
corresponds to a shift of the output sequence one place to the left that increases its value by a power of 2. Let us compute the output binary sequence.



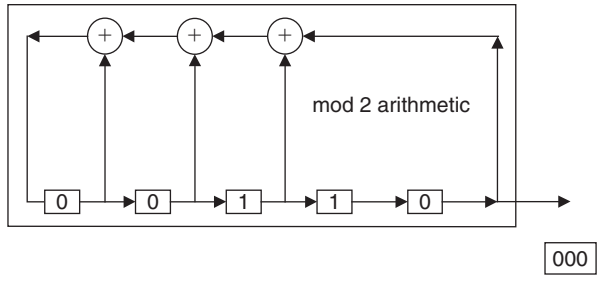
At the first clock pulsed, the generator output '0' and the generator states change to:



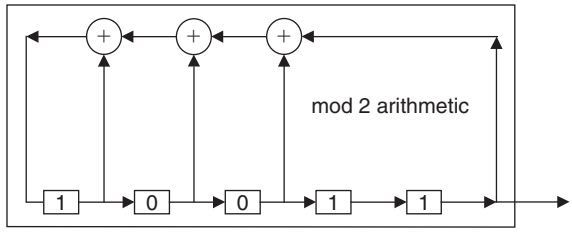
At the 2nd clock pulse, the generator output '0' and its states change to:



At the 3rd clock pulse, the generator output '0' and its states change to:

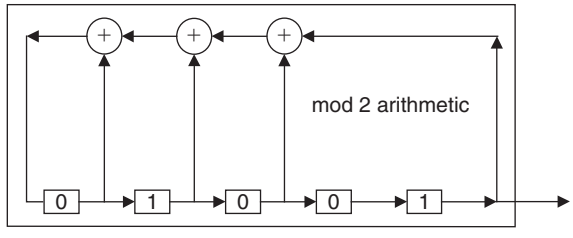


At the 4th clock pulse, the generator output ‘0’ and its states change to:



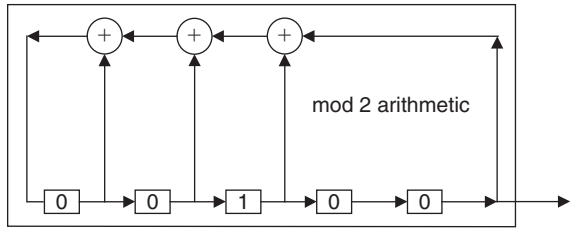
0000

At the 5th clock pulse, the generator output ‘1’ and its states change to:



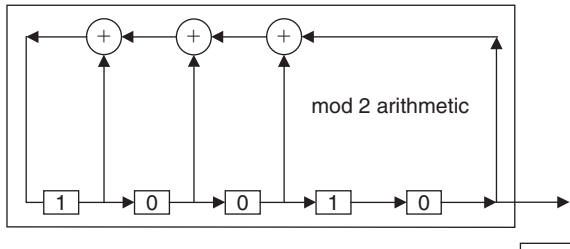
10000

At the 6th clock pulse, the generator output ‘1’ and its states change to:



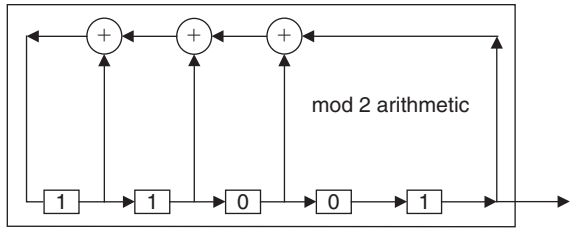
110000

At the 7th clock pulse, the generator output ‘0’ and its states change to:



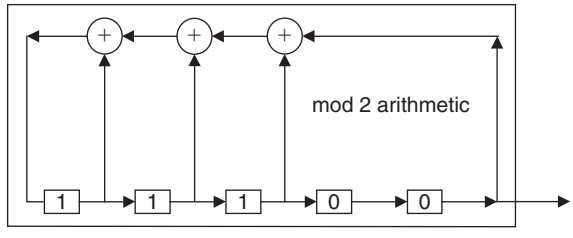
0110000

At the 8th clock pulse, the generator output ‘0’ and its states change to:



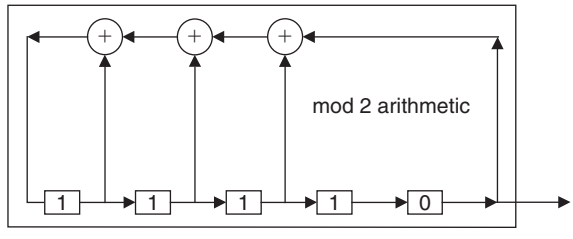
00110000

At the 9th clock pulse, the generator output ‘1’ and its states change to:



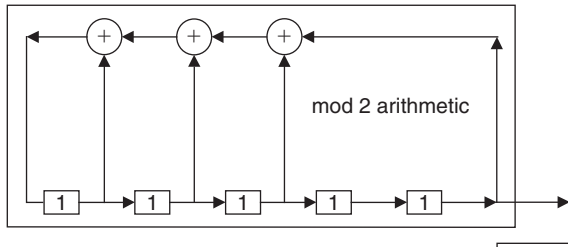
100110000

At the 10th clock pulse, the generator output ‘0’ and its states change to:



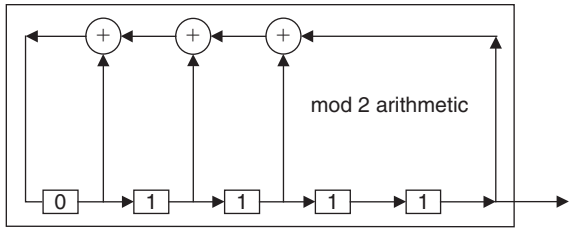
0100110000

At the 11th clock pulse, the generator output ‘0’ and its states change to:



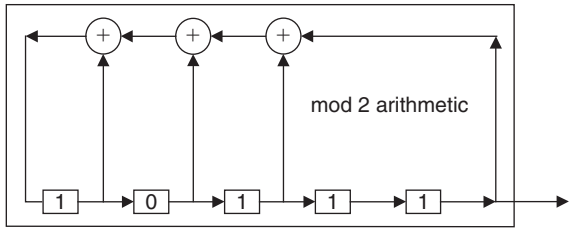
00100110000

At the 12th clock pulse, the generator output '1' and its states change to:



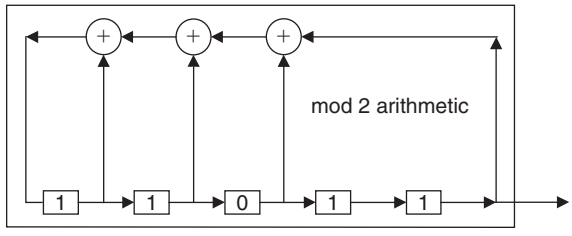
100100110000

At the 13th clock pulse, the generator output '1' and its states change to:



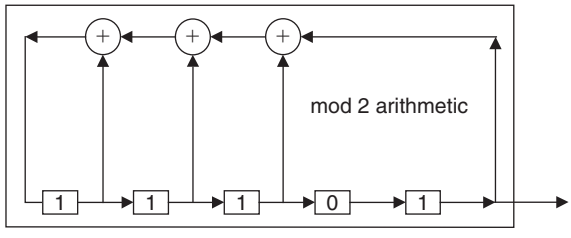
1100100110000

At the 14th clock pulse, the generator output '1' and its states change to:



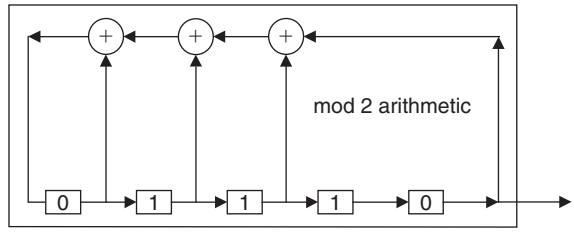
11100100110000

At the 15th clock pulse, the generator output '1' and its states change to:



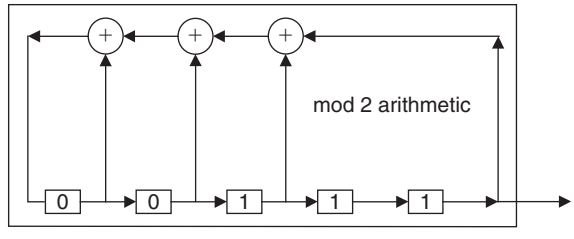
111100100110000

At the 16th clock pulse, the generator output '1' and its states change to:



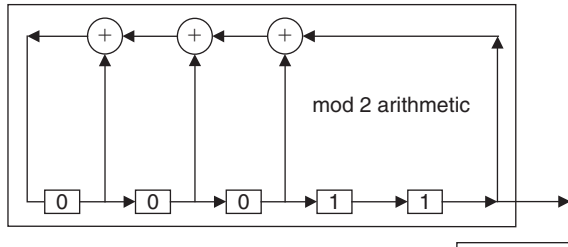
1111100100110000

At the 17th clock pulse, the generator output '0' and its states change to:



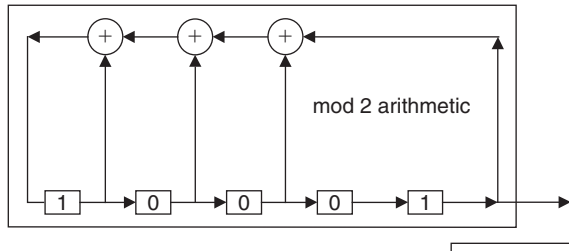
01111100100110000

At the 18th clock pulse, the generator output '1' and its states change to:



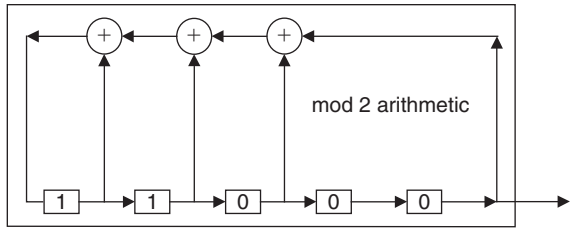
101111100100110000

At the 19th clock pulse, the generator output '1' and its states change to:



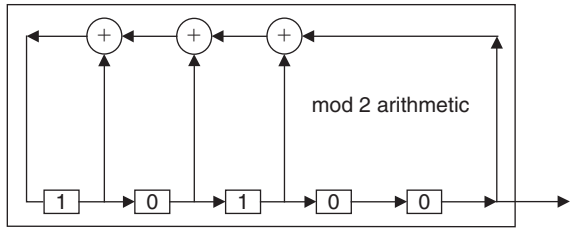
1101111100100110000

At the 20th clock pulse, the generator output ‘1’ and its states change to:



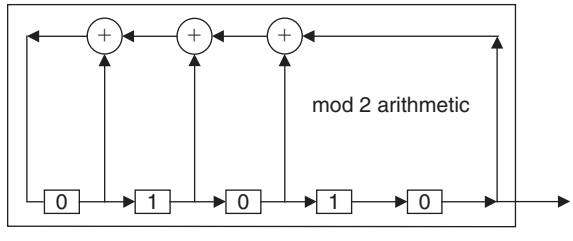
11101111100100110000

At the 21st clock pulse, the generator output ‘0’ and its states change to:



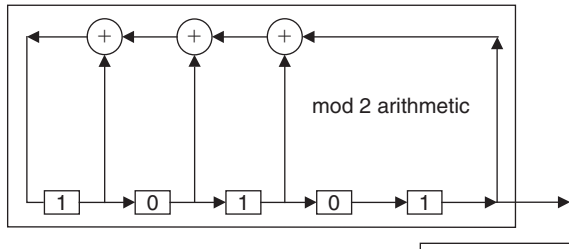
01110111100100110000

At the 22nd clock pulse, the generator output ‘0’ and its states change to:



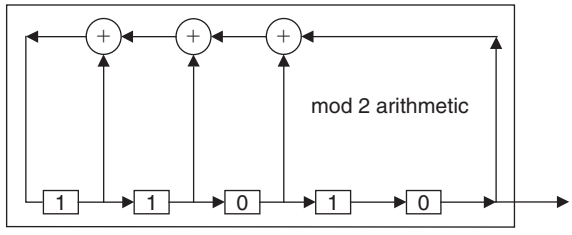
00111011111100100110000

At the 23rd clock pulse, the generator output ‘0’ and its states change to:



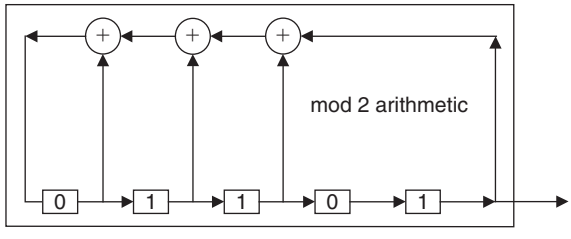
00011101111100100110000

At the 24th clock pulse, the generator output '1' and its states change to:



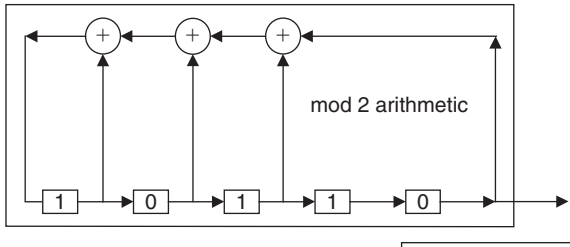
100011101111100100110000

At the 25th clock pulse, the generator output '0' and its states change to:



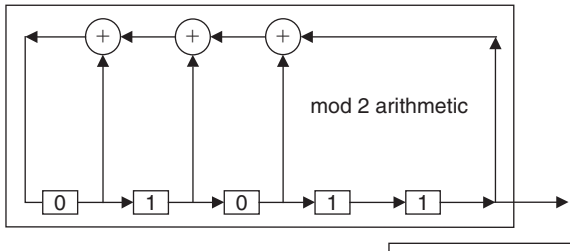
010001110111100100110000

At the 26th clock pulse, the generator output '1' and its states change to:



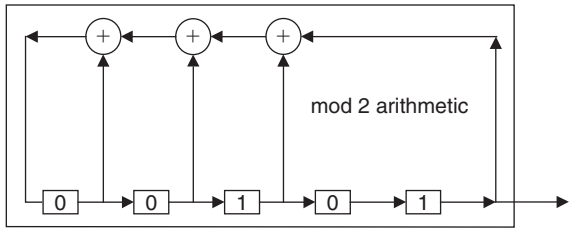
101000111011100100110000

At the 27th clock pulse, the generator output '0' and its states change to:



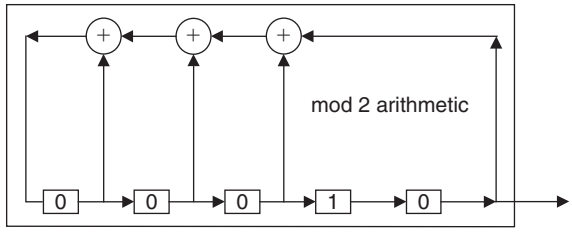
010100011101111100100110000

At the 28th clock pulse, the generator output ‘1’ and its states change to:



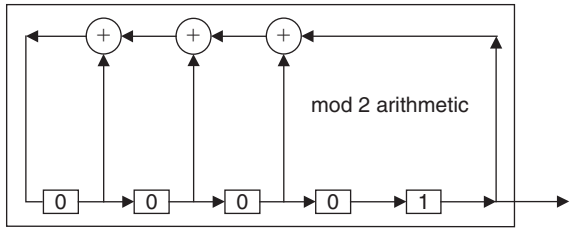
10101000111011111100100110000

At the 29th clock pulse, the generator output ‘1’ and its states change to:



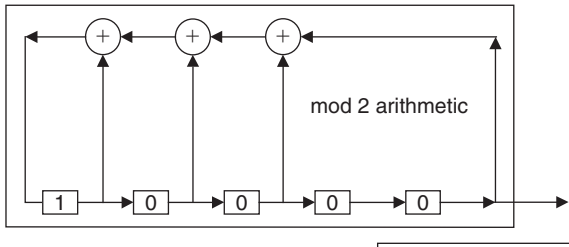
11010100011101111100100110000

At the 30th clock pulse, the generator output ‘0’ and its states change to:



01101010001110111100100110000

At the 31st clock pulse, the generator output ‘1’ and its states change to:



10110101000111011111100100110000

The binary LFSR revert to its initial conditions and the binary sequence is:

10110101000111011111100100110000

Now shift left one place to account for multiplication by 2 and we get the sequence after the multiplier:

01101010001110111110010011000 01

Now add the two sequences to get:

13002302111333200230122111100 01

The desired sequence is obtained using (4.48):

$$\begin{aligned}\{\hat{a}_n\} \;=\; & \mathrm{j}, -\mathrm{j}, 1, 1, -1, -\mathrm{j}, 1, -1, \mathrm{j}, \mathrm{j}, \mathrm{j}, -\mathrm{j}, -\mathrm{j}, -\mathrm{j}, -1, 1, 1, -1, -\mathrm{j}, 1, \mathrm{j}, \\ & -1, -1, \mathrm{j}, \mathrm{j}, \mathrm{j}, \mathrm{j}, 1, 1, 1, \mathrm{j} \\ & \frac{\pi}{2}, \frac{3\pi}{2}, 2\pi, 2\pi, \pi, \frac{3\pi}{2}, 2\pi, \pi, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}, \frac{3\pi}{2}, \frac{3\pi}{2}, \frac{3\pi}{2}, \pi, 2\pi, 2\pi, \pi, \frac{3\pi}{2}, \\ & 2\pi, 2\pi, \pi, \pi, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}, \frac{\pi}{2}, 2\pi, 2\pi, 2\pi, \frac{\pi}{2}\end{aligned}$$

4.6.2 Polyphase sequences

Polyphase sequences are finite complex sequences with constant amplitude and time discrete phases. These sequences have desirable periodic and aperiodic auto- and cross-correlation properties, which make them optimal for applications in radar, system identification and spread-spectrum communications.

A polyphase sequence of length N can be defined by N complex elements $\hat{a}_i$ given by:

$$\hat{a}_i = a_i \, e^{j\theta_i} \tag{4.49}$$

where θ_i represents the N phases given by the sequence:

$$\{\theta_0, \theta_1, \theta_2, \ldots\ldots\ldots\ldots\ldots \theta_{N-1}\} \tag{4.50}$$

$j = \sqrt{-1}$ and a_i is the magnitude of i^{th} element in the complex polyphase sequence. The sequence symbols are the complex N^{th} root of unity where N is a prime number for finite Galois field as demonstrated in Section 4.6.1.

The literature is rich with works on polyphase sequences proposing their use as signature sequences in multi-user spread-spectrum systems but most of the proposed techniques are still open for research; especially the practical issues concerning the implementation of these sequences and the performance of the multi-user spread-spectrum systems which use the polyphase code sequences.

A polyphase sequence with elements of magnitude 1 is called a *Baker sequence* if the maximum magnitude of all side lobes of their aperiodic autocorrelation function is less than or equal to 1. Generally, optimization algorithms are used to search for these sequences (Krone and Sarwate, 1984). However, the number of polyphase Baker sequences discovered is 31, each of length 31 (Fan and Darnell, 1995) and because of their small family size, they have minimal use in multiple access communication system applications.

Frank polyphase sequences with a relatively large set size ($=\sqrt{N}-1$) are denoted as $F=\{\hat{a}^{(1)},\hat{a}^{(2)},\ldots\ldots\ldots,\hat{a}^{(r)},\ldots\ldots\ldots,\hat{a}^{(q-1)}\}$. Each sequence is of length $N=q^2$. The elements of the sequence are the qth root of unity and are given by Frank (1963):

$$\hat{a}^{(r)}=\{a_0^{(r)},a_1^{(r)},\ldots\ldots\ldots\ldots,a_{N-1}^{(r)}\} \tag{4.51}$$

$$a_n^{(r)}=\hat{a}_{iq+k}^{(r)}=e^{j\frac{2\pi}{q}rik}=\alpha^{rik} \tag{4.52}$$

where $\alpha=e^{j\frac{2\pi}{q}}$ and $0\le n\le N-1, i<q$, q is any integer, and $\gcd(r,q)=1$. It has been shown in Frank (1963) that the periodic ACF and CCF of Frank sequences are as follows:

$$\begin{aligned}\mathrm{ACF}(\tau) &= N \quad \text{for } \tau=0\\ &= 0 \quad \text{for } \tau\neq 0\end{aligned} \tag{4.53}$$

$$\text{and } \mathrm{CCF}(\tau)=\sqrt{N} \quad \text{for q is odd.} \tag{4.54}$$

Chu (1972) polyphase sequences exist for every integer N. The sequences $\{a_n^{(r)}\}$ of length N and for $(0\le n\le N)$ are defined by:

$$a_n^{(r)} = e^{j\frac{\pi}{N}r(1+n)n+mn} \quad \text{for N odd} \tag{4.55}$$

$$= e^{j\frac{\pi}{N}rn^2+mn} \quad \text{for N even} \tag{4.56}$$

where m is any integer and gcd(r,N) = 1.

It has been shown (Chu, 1972) that the periodic ACF and CCF of the Zadoff-Chu sequences are as follows:

$$\begin{aligned}\mathrm{ACF}(\tau) &= N \quad \text{for } \tau=0\\ &= 0 \quad \text{for } \tau\neq 0\end{aligned} \tag{4.57}$$

$$\text{and } \mathrm{CCF}(\tau)=\sqrt{N} \quad \text{for } \gcd(r-s,N)=1 \text{ and N is odd.} \tag{4.58}$$

A comprehensive treatment of the complex sequences and the behaviour of the correlation functions is given in Fan and Darnell (1995), Chapters 7 to 11.

The Frank-Zadoff-Chu (FZC) polyphase sequences have excellent correlation properties. The maximum out-of-phase periodic autocorrelation and the maximum periodic

cross-correlation (C_{max}) of the Gold, quadriphase, and ZFC sequences are given in Lam and Ozluturk (1992) as:

$$C_{max} = \begin{cases} \sqrt{2(N+1)} + 1 & \text{Gold sequences} \\ \sqrt{N + \sqrt{2(N+1)} + 2} & \text{Quadriphase sequences} \\ \sqrt{N} & \text{FZC sequences} \end{cases} \tag{4.59}$$

where N is the length of the sequence. For example, where $N = 128$, the maximum correlations are 17.06, 14.00, and 11.31 for the three classes of sequences. However, the degradation in system performance does not depend on periodic correlation but on the odd correlation as well. Chu (1972) and Lam and Ozluturk (1992) discuss how multi-user spread-spectrum systems with FZC code sequences exhibit a slightly better BER performance under large SNR values, i.e. show lower multiple access interference.

Oppermann and Vucetic (1997) presented a new family of complex valued sequences for use in multi-user CDMA spread-spectrum systems which offer a choice of a wide range of correlation properties. The i^{th} element of the Oppermann-Vucetic sequence μ_i is defined by:

$$\mu_i = (-1)^{Mi}\, e^{\frac{j\pi(M^m i^p + i^n)}{N}} \quad 1 \le i \le N, 1 \le M \le N \tag{4.60}$$

where $j = \sqrt{-1}$, and M is an integer that is relatively prime to N. The variables m, n, and p are any real numbers that specify the sequence set. However, the sequences in the family are constant amplitude complex numbers, each sequence in the set differs only in phase terms. When $p = 1$, each sequence in the set will have the same autocorrelation function magnitude. When N is prime, the maximum set size for any combination of parameters of $N - 1$ will be achieved.

The performance of CDMA systems using sequence (4.57) is superior to any other known large set of sequences (Oppermann and Vucetic, 1997). However, methods of constructing these sequences are open problems for research. Oppermann (1997) has identified a subfamily of spreading sequences within the sequences presented in Oppermann and Vucetic (1997), which offer larger sets of orthogonal sequences.

An interesting class of polyphase sequences for prime length (N) greater than 3 is suggested in Fan et al. (1994). The size of the set is equal N and r^{th} polyphase sequence $a^{(r)}$ is defined as:

$$a^{(r)} = \{a_0^{(r)}, a_1^{(r)}, \ldots\ldots\ldots, a_{N-1}^{(r)}\} \tag{4.61}$$

$$a_n^{(r)} = \alpha^{\frac{n(n+1)(n+2)}{6} + rn}$$

$$\alpha = e^{\frac{j2\pi v}{N}} \quad 0 \le n \quad r < N \tag{4.62}$$

where α is the N^{th} root of unity and r, n and v are integers relatively prime to N. It was shown in Fan and Darnell (1995) that the periodic autocorrelation is a two-valued function given by:

$$\begin{aligned} |R_{r,r}(\tau)| &= N \qquad \tau = 0 \\ &= \sqrt{N} \quad \tau \neq 0 \end{aligned} \tag{4.63}$$

and the periodic cross-correlation is a two-valued function given by:

$$\begin{aligned} |R_{r,s}(\tau)| &= 0 \qquad \tau = 0 \\ &= \sqrt{N} \quad \tau \neq 0 \end{aligned} \tag{4.64}$$

However, neither the implementation of these sequences nor their performance in a multi-user spread-spectrum system were discussed in Fan et al. (1994).

4.7 Summary

In this chapter we have presented the theory and implementation of the pseudo-random code sequences commonly used in spread-spectrum systems. We started by introducing basic algebra concepts that are used in the sequence analysis in Section 4.2, followed by the arithmetic of the binary polynomials in Section 4.3. Computation of the elements of Galois's field, important in the generation of the sequences, was presented in Section 4.4, while Section 4.5 presented the generation and decimations of the most common PN-sequences, called the m-sequences. These sequences have many applications in coding theory, spread-spectrum techniques and cryptography. Methods using two preferred m-sequences for deriving the Gold and Kasami sequences that have reduced cross-correlation between any pair in the set were also presented in Sections 4.5.5 and 4.5.6. Orthogonal sequences such as the Walsh-Hadamard and orthogonal Gold sequences are used for logical chanallization and were described in Sections 4.5.7 and 4.5.8.

Recent interest in complex code sequences such as quadriphase and polyphase sequences were considered in Section 4.6 where we examined in detail the use of quadriphase and polyphase sequences as users' signatures.

Problems

4.1 Consider polynomials $P_1(x)$ and $P_2(x)$ with coefficients drawn from Galois field GF(4) such that:

$$\begin{aligned} P_1(x) &= 2x + 3x^2 + x^4 \\ P_2(x) &= 1 + 3x^3 + x^4 \end{aligned}$$

Evaluate the following mathematical expressions:

i. $P_1(x) + P_2(x)$
ii. $P_1(x) - P_2(x)$

iii. $P_1(x) \cdot P_2(x)$

iv. $\dfrac{P_1(x)}{P_2(x)}$

4.2 Find the elements that arise from the addition of each pair of elements of the polynomials in GF(2^3).

4.3 Find the elements in GF(2^3) using the primitive polynomial $P(x) = 1 + x^2 + x^3$.

4.4 Consider the *Galois feedback implementation* of the sequence generator shown in Figure 4.4 with the generator polynomial g(x) given by:

$$g(x) = 1 + x^3 + x^4$$

Assume the initial load of the register be 0001.

i. Find the output periodic sequence.

ii. What would the maximum possible period be?

4.5 Find the sequence at the output of the Fibonacci generator shown in Figure 4.5 with polynomial given by:

$$g(x) = 1 + x + x^4$$

Assume the initial load is 0001.

4.6 Show that the m-sequence obtained from the primitive polynomial $(1 + x + x^4)$ is [00 01 00 11 01 01 11 1].

4.7 A sequence u generated by a primitive polynomial of degree 6 is decimated by $q = 27$ to generate sequence $\upsilon = u(q)$. What would be the period of sequence υ?

4.8 An m-sequence u of period $N = 15$ is decimated by $q = 7$ to generate sequence $\upsilon = u(7)$. Given u = [00 10 01 10 10 11 11 0]. What would be sequence υ?

4.9 Four sequences, 7-bits each, are chosen from a set of Gold sequences. Compute the cross-correlation between each pair for time shift (τ) between the sequences to be 0, 1, 2, 3, 4, 5 and suggest a format that eliminates the cross-correlation for zero time shift.

$$G_1 = [01\ 10\ 10\ 1]$$

$$G_2 = [00\ 10\ 01\ 0]$$

$$G_3 = [11\ 00\ 11\ 0]$$

$$G_4 = [10\ 11\ 10\ 0]$$

4.10 Consider the primitive polynomial h(x) of degree $m = 4$ which is dividing $(x^{2^m - 1} - 1)$. The shift registers are loaded with the following initial states: $(a_{n-1}, a_{n-2}, a_{n-3}, a_{n-4}) = (1, 0, 0, 0)$. Generate the quadriphase sequences using mod 4 arithmetic LFSR generator.

References

Chu, D.C. (1972) Polyphase codes with good periodic correlation properties, *IEEE Transactions on Information Theory*, 18, 531–532.

Fan, P. and Darnell, M. (1995) *Sequence Design for Communications Applications*, Publisher: Taunton: Research Studies Press Ltd, page 118.

Fan, P.Z., Darnell, M. and Honary, B. (1994) New class of polyphase sequences with two-valued auto- and cross-correlation functions, *Electronics Letters*, 30(13), 1031–1032.

Frank, R.L. (1963) Polyphase codes with good non-periodic correlation properties, *IEEE Transactions on Information Theory*, IT-9, 43–45.

Hanzo, L., Munster, M., Choi, B. J. and Keller, T. (2003) *OFDM and MC-CDMA for Broadband Multi-user Communications, WLANs and Broadcasting*, John Wiley & Sons.

Krone, S. M. and Sarwate, D.V. (1984) Quadriphase sequences for spread-spectrum multiple access communication, *IEEE Transactions on Information Theory*, IT-30(3), 520–529.

Kumar, P.V., Helleseth, T., Calderbank, A.R. and Hammons, A.R. (1996) Large families of quaternary sequences with low correlation, *IEEE Transactions on Information Theory*, 42(2), 579–592.

Lam, A.W. and Ozluturk, F.M. (1992) Performance bound for DS/SSMA communications with complex signature sequences, *IEEE Transactions on Communications*, 40(10), 1607–1614.

Oppermann, I. and Vucetic, B.S. (1997) Complex spreading sequences with a wide range of correlation properties, *IEEE Transactions on Communications*, 45(3), 365–375.

Oppermann, I. (1997) Orthogonal complex-valued spreading sequences with a wide range of correlation properties, *IEEE Transactions on Communications*, 45(11), 1379–1380.

Ozluturk, F.M., Tantaratana, S. and Lam, A.W. (1995) Performance of DS/SSMA Communications with MPSK signalling and complex signature sequences, *IEEE Transactions on Communications*, 43(2/3/4), Feb/Mar/April.

Peterson, R.L., Ziemer, R.E. and Borth, D.E. (1995) *Introduction to Spread-Spectrum Communications*, Prentice Hall Inc.

Proakis, J.G. (1995) *Digital Communications*, McGraw-Hill International Editions.

Sarwate, D.V. and Pursley, M.B. (1980) Cross-correlation properties of pseudorandom and related sequences, *Proceedings of the IEEE*, 68(5), 593–619.

Welch, L.R. (1974) Lower bounds on the maximum cross-correlation of signals, *IEEE Transactions on Information Theory*, IT-20, 397–399.