International Journal of Electrical and Computer Engineering 3:3 2008

On the Construction of m-Sequences via Primitive Polynomials with a Fast Identification Method

Abhijit Mitra

***Abstract*—The paper provides an in-depth tutorial of mathematical construction of maximal length sequences (m-sequences) via primitive polynomials and how to map the same when implemented in shift registers. It is equally important to check whether a polynomial is primitive or not so as to get proper m-sequences. A fast method to identify primitive polynomials over binary fields is proposed where the complexity is considerably less in comparison with the standard procedures for the same purpose.**

***Keywords*—Finite field, irreducible polynomial, primitive polynomial, maximal length sequence, additive shift register, multiplicative shift register.**

I. Introduction

FINITE fields are ubiquitous in computer science and communications. They help form a foundation for such areas as coding theory and cryptography, as well as they are a fundamental building block in discrete mathematics [1]. Yet finite fields are often covered in EE or CS course curriculum only in brief for some specific purpose. This paper thus deals with a broad but gentle introduction to finite fields, i.e., Galois fields. One significant application of finite fields is to generate sequences: in particular, maximal length sequences (m-sequences, in short). Properties of finite fields transfer readily to certain properties of m-sequences like correlation, span, linear complexity, and so forth. As a standard tool for computer scientists and engineers, m-sequences especially have their roles as pseudo noise sequences with a thorough application in spread spectrum communications. Treatments of m-sequences are, however, often focused only on the generation of the sequence in terms of shift registers while leaving out the basic relationship of m-sequences and finite fields that may be germane or even essential to an application. We focus on the relation in this paper and show the instrumentality of primitive polynomials over a finite field in generating m-sequences. Also, we propose a fast method to identify primitive polynomials over binary fields where the complexity is significantly less in comparison with the standard procedures.

The organization of the paper is as follows. In Section 2, we deal with the fundamentals of algebraic operations to introduce the notion of Galois field and primitive polynomials that is needed for our purpose. Section 3 shows the construction of m-sequences from primitive polynomials and how to map the same when implemented in shift registers. The standard method along with the proposed simplified approach for identifying a primitive polynomial is given in Section 4. The paper is concluded in Section 5 by summarizing the important concepts discussed herein. An easy way of practically generating Gaussian sequences from such m-sequences is discussed in the Appendix.

Manuscript received August 01, 2008.

A. Mitra is with the Department of Electronics and Communication Engineering, Indian Institute of Technology (IIT) Guwahati, Guwahati – 781039, India. E-mail: a.mitra@iitg.ernet.in.

II. Fundamentals of Algebraic Operations

We discuss here about finite fields, starting from the very basic notion of fields, along with the construction of such fields with the help of primitive polynomials. As we explain later, these primitive polynomials play an important role in generating m-sequences.

A. Fields

A field is an algebraic structure [2] in which the operations of addition, subtraction, multiplication, and division (except by zero) can be performed, and satisfy the usual rules. More precisely, a field is a set $\mathcal{F}$ with two binary operations $+$ (addition) and . (multiplication) defined on it, in which the following laws hold.

(A1) $a + (b + c) = (a + b) + c$ (associative law for addition)
(A2) $a + b = b + a$ (commutative law for addition)
(A3) There is an element 0 (zero) such that $a + 0 = a\ \forall a$
(A4) $\forall a$, there is an element $-a$ such that $a + (-a) = 0$
(M1) $a.(b.c) = (a.b).c$ (associative law for multiplication)
(M2) $a.b = b.a$ (commutative law for multiplication)
(M3) There is an element 1 ($\neq 0$) such that $a.1 = a\ \forall a$
(M4) $\forall a \neq 0$, there is an element a^{-1} such that $a.a^{-1} = 1$
(D) $a.(b + c) = (a.b) + (a.c)$ (distributive law)

Using the notion of elementary group theory (i.e., an Abelian group is a commutative group; homomorphism is a function that maps the elements of one group to another group with certain property; isomorphism is a homomorphism that is also a bijection mapping; and, automorphism is an isomorphism that maps a group onto itself), we can condense these nine axioms into just three [3]:
(1) The elements of $\mathcal{F}$ form an Abelian group with the operation $+$ (called the additive group of $\mathcal{F}$);
(2) The non-zero elements of $\mathcal{F}$ form an Abelian group under the operation . (called the multiplicative group of $\mathcal{F}$);
(3) Multiplication by any non-zero element is an automorphism of the additive group.

Depending upon the number of elements in it, a field is called either a *finite* or an *infinite* field. The examples of *infinite* field include $\mathbb{Q}$ (set of all rational numbers), $\mathbb{R}$ (set of all real numbers), $\mathbb{C}$ (set of all complex numbers) etc. On the other hand, a field with distinct q elements is called a finite field,

TABLE I
ADDITION AND MULTIPLICATION TABLES FOR $GF(3)$

$+$	0	1	2	.	0	1	2
0	0	1	2	0	0	0	0
1	1	2	0	1	0	1	2
2	2	0	1	2	0	2	1

or, *Galois*[1] *field* and is denoted by $GF(q)$. We discuss about construction of such fields, which is of our interest of this article, as well as the application of such fields below.

B. Galois Fields

Every *Galois* field contains a subfield that has a prime number of elements and this field is called the *prime subfield* or the *basefield*. Mathematically, this can be represented as: assuming q to be a prime number, the integers modulo q (denoted by $\mathbb{Z}_q$) form a *Galois* basefield $GF(q)$, i.e., its elements are the congruence classes of integers (mod q), with addition and multiplication induced from the usual integer operations. In other words, the elements of $GF(q)$ are $\{0, 1, 2, ..., q-1\} \in \mathbb{Z}_q$ and these integers follow all the nine properties (A1)-(D) as given above with respect to mod q. The simplest possible example is $GF(2)$ which is a binary basefield with elements $\{0, 1\}$ with $+$ and . defined on it as $0{+}0 = 0$, $0{+}1 = 1{+}0 = 1$, $1 + 1 = 2 = 0$ (mod 2), $0.0 = 0$, $0.1 = 1.0 = 0$, $1.1 = 1$. The example of a ternary basefield $GF(3)$ is shown in Table 1 with addition and multiplication tables with respect to mod 3 where $\{0, 1, 2\}$ have been used as representatives of the congruence classes.

With the above definition and examples of *Galois* field, the immediate question that comes into reader's mind is, whether such a field can exist for any general finite integer t, where t is not a prime number. We deal with this answer in the following.

The characteristic of a field, not necessarily finite, is the value of prime q such that adding any element to itself q times results in 0. If no such q exists, then the field is said to be characteristic 0. For example, $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$ are all characteristic 0 fields. In fact, the set of numbers in $GF(q)$ generated by repeatedly adding 1 is closed under both addition, multiplication, subtraction and division, and is isomorphic to $\mathbb{Z}_q$. This is not true for any t. For example, $\mathbb{Z}_6$ cannot be a *Galois* field as in $\mathbb{Z}_6$, we have $2.3 = 0$ (mod 6) which shows that neither 2 nor 3 can have inverses. However, following some elementary results in group theory, one can find that if $a \neq 0$ (mod q), where $a \in GF(q)$, then $a^{q-1} = 1$. Multiplying both sides by a shows that $a^q - a = 0$ for $a \neq 0$. Since this is also true for $a = 0$, we see that every element in $\mathbb{Z}_q$ satisfies the following polynomial equation

$$x^q - x = 0 \tag{1}$$

which has q roots with each root being exactly the elements in $\mathbb{Z}_q$. Putting $q = q^m$ above yields the equation $x^{q^m} - x = 0$. This, in turn, means construction of a $GF(t)$ is possible with the above closure property, if and only if $t = q^m$ where m is any positive integer. A *Galois* field $GF(q^m)$ is therefore said to be an extension of basefield $GF(q)$, with q^m elements and with addition and multiplication done in the usual way with respect to mod q. It then follows that $GF(q^m)$ has a m element basis over $GF(q)$. That is, there exist m distinct elements $u_0, ..., u_{m-1} \in GF(q^m)$ such that each of the q^m elements of $GF(q^m)$ can be expressed as $a_0u_0 + ... + a_{m-1}u_{m-1}$ for some (unique) coefficient $a_i \in GF(q)$. As $GF(q^m)$ is a prime power field, let us assume for the moment that its elements are generated by power addition and thus, $\{1, u^1, ..., u^{m-1}\}$ form the basis of $GF(q^m)$ for some u. Then,

[1]Evariste Galois, 1811-1832, who although dying young and only publishing a handful of papers, wrote seminal works in group and field theory.

TABLE II
THE ELEMENTS OF THE FIELD $GF(2^4)$ AS POWERS OF u AND AS DIFFERENT POLYNOMIALS

u^3	u^2	u^1	1	Power of u
0	0	0	1	u^0
0	0	1	0	u^1
0	1	0	0	u^2
1	0	0	0	u^3
0	0	1	1	u^4
0	1	1	0	u^5
1	1	0	0	u^6
1	0	1	1	u^7
0	1	0	1	u^8
1	0	1	0	u^9
0	1	1	1	u^{10}
1	1	1	0	u^{11}
1	1	1	1	u^{12}
1	1	0	1	u^{13}
1	0	0	1	u^{14}
0	0	0	1	$u^{15} = u^0 = 1$

$$u^m - a_{m-1}u^{m-1} - ... - a_0 = 0 \tag{2}$$

since u^m must be a linear combination on the basis. This shows that we can multiply two arbitrary elements in $GF(q^m)$ by expressing the elements in the basis, multiplying them as polynomials in u and reducing the result by the relation in (2).

As an illustration, let us see how to construct $GF(2^4)$ such that $\{1, u^1, u^2, u^3\}$ is a basis over $GF(2)$ with the relationship

$$x^4 + x + 1 = 0. \tag{3}$$

That is, letting u be a root of this equation, we get the basic relationship $u^4 = u + 1$ with respect to mod 2. This means all the $2^4 = 16$ elements of $GF(2^4)$, $\{1, u^1, u^2, ..., u^{15}\}$, can be expressed in terms of linear combinations of $\{1, u^1, u^2, u^3\}$ for some (unique) $a_i \in GF(2)$ using (3). For example, $u^5 = u.u^4 = u.(u + 1) = u^2 + u$. Similarly, all the other elements can also be represented by a polynomial expression where the highest degree of the polynomial is 3. The polynomial expressions, as the coefficients of $\{1, u^1, u^2, u^3\}$, are given in Table 2 for this example. We observe from Table 2 that beyond u^{q^m-2}, the higher powers of u repeat the elements of $GF(q^m)$ which comes from the closure property as given in (1). It is also seen that all zero coefficients cannot be a valid expression for any u^i.

C. Primitive Polynomials over $GF(q^n)$

It is shown in (1) that *Galois* field elements can be expressed in terms of polynomials and more explicitly in (2) which also

TABLE III
A SET OF PRIMITIVE POLYNOMIALS UP TO DEGREE 30 FOR BINARY FIELDS

degree (n)	$p(x)$
1	$x+1$
2	x^2+x+1
3	x^3+x+1
4	x^4+x+1
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x+1
8	$x^8+x^6+x^5+x+1$
9	x^9+x^4+1
10	$x^{10}+x^3+1$
11	$x^{11}+x^2+1$
12	$x^{12}+x^7+x^4+x^3+1$
13	$x^{13}+x^4+x^3+x+1$
14	$x^{14}+x^{12}+x^{11}+x+1$
15	$x^{15}+x+1$
16	$x^{16}+x^5+x^3+x^2+1$
17	$x^{17}+x^3+1$
18	$x^{18}+x^7+1$
19	$x^{19}+x^6+x^5+x+1$
20	$x^{20}+x^3+1$
21	$x^{21}+x^2+1$
22	$x^{22}+x+1$
23	$x^{23}+x^5+1$
24	$x^{24}+x^4+x^3+x+1$
25	$x^{25}+x^3+1$
26	$x^{26}+x^8+x^7+x+1$
27	$x^{27}+x^8+x^7+x+1$
28	$x^{28}+x^3+1$
29	$x^{29}+x^2+1$
30	$x^{30}+x^{16}+x^{15}+x+1$

indicates that all the elements have different polynomial representations with respect to a basic relationship. We thus now investigate more into the form of polynomials for generating any $GF(q^n)$. Analogous to expression (2), let us define a general polynomial $p(x)$ of order n over a $GF(q^n)$ as

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0 \tag{4}$$

where, as earlier, all the coefficients a_i, $i = 0, 1, ..., n$ are members of $GF(q)$, i.e., integers ranging from 0 to $q-1$ with $a_n = a_0 = 1$. The polynomial of (4) is called *irreducible* in $GF(q^n)$ if $p(x)$ cannot be factored into a product of lower-degree polynomials. An *irreducible* polynomial $p(x) \in GF(q^n)$ of degree n is said to be *primitive* if the smallest positive integer l for which $p(x)$ divides $h(x) = x^l - 1$ is $l = q^n - 1$. For binary fields ($GF(2^n)$), a set of primitive polynomials up to $n = 30$ is shown in Table 3, which is quite sufficient for most of the purposes. Primitive polynomials of much higher degree can be found in [4].

At this point, another question might come to reader's mind: whether there exist irreducible and primitive polynomials of degree n for each n. The answer to this question is 'Yes'. Note that, not only one, there exist more than one irreducible and primitive polynomials for any degree $n > 1$. These are usually calculated using Mobius functions and Euler φ functions, respectively. For example, $\beta_2(n)$ is defined as the number of primitive polynomials of degree n ($\in GF(2^n)$) and the mathematical relation of this with Euler function $\varphi(2^n-1)$ is $\beta_2(n) = \frac{\varphi(2^n-1)}{n}$ where $\varphi(2^n-1)$ is defined as the number of positive integers not exceeding 2^n-1 and coprime to 2^n-1. However, discussion about this is beyond the scope of this article. Interested readers can find the rather complicated proofs on these in [5].

With the help of primitive polynomials, we can construct a class of linear recurring sequences, called m-sequence [6]-[7], which is highly important in a number of applications mainly including spread spectrum communications. We describe below the generation process of such sequences by primitive polynomials.

III. CONSTRUCTION OF MAXIMAL LENGTH SEQUENCES

Let us assume a class of finite length sequences that are recurring and thus periodic in nature. By a periodic sequence, we mean a countably infinite list of values $\mathbf{s} = (s_0, s_1, ...)$ such that $s_{i+N} = s_i \ \forall\ N \geq 1$ and $i \geq 0$, where N refers to the period of the sequence. Our aim is, however, to find out a relation between such linear recurrence and primitive polynomials so as to generate m-sequences from primitive polynomials.

A. Linear Recurrence

In general, any linear recurrence of order n is given by

$$s_{i+n} = \sum_{k=0}^{n-1} a_k s_{i+k}, \forall i \geq 0. \tag{5}$$

Let us assume that $s_i \in GF(2^n)$ for computational purposes and $a_k \in GF(2)$, i.e., we are mainly interested in binary field and its extensions. It is clear that in (5), specifying the initial values $(s_{n-1}, ..., s_1, s_0)$ completely specifies the sequence. Indeed, specifying any n consecutive values specifies all remaining values and, assuming $a_0 = 1$, all previous values down to s_0. Since there is only a finite number of possible combinations of $s_{n-1}, ..., s_0$, or, finite states, any sequence satisfying (5) must repeat and the period of such a nonzero sequence can be at most $2^n - 1$ as there are 2^n possible states (the all-zeros state cannot occur unless the sequence is itself all-zeros).

As an example, consider the relation $s_{i+4} = s_{i+1} + s_i$ with the initial state $(0, 0, 0, 1)$. The resulting sequence is binary and has a period of 15 with the first 15 values as 100010011010111. The values repeat itself after this period.

B. Sequences via Characteristic Polynomials

Let us take a characteristic polynomial of the linear recurrence (5) as

$$f(x) = x^n - \sum_{k=0}^{n-1} a_k x^k \tag{6}$$

by replacing s_{i+k} with x^k. The relation $f(x) = 0$ is called the characteristic equation of this. Let u be a root of this equation, meaning $u^n = \sum_{k=0}^{n-1} a_k u^k$. Multiplying both sides by u^i, we get

$$u^{i+n} = \sum_{k=0}^{n-1} a_k u^{i+k} \tag{7}$$

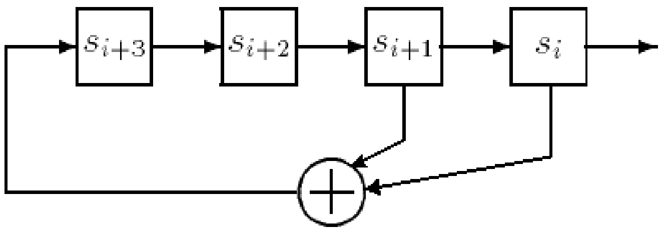

Fig. 1. The additive shift register structure.

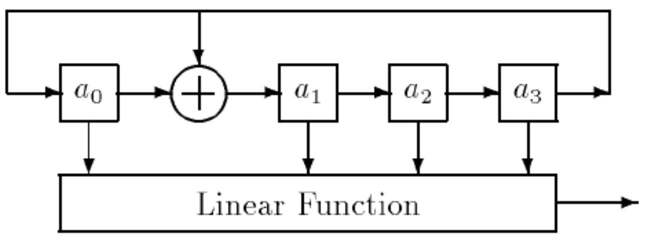

Fig. 2. The multiplicative shift register structure.

which shows that the sequence defined by $s_i = u^i$ satisfies the recurrence in (5). Now, if we compare equation (6) and (4), we find that they are equivalent as, in a characteristic 2 field, minus signs can be changed to plus signs. Therefore, if $f(x)$ is irreducible, then it will have n distinct roots: $\{1, u^1, u^2, ..., u^{n-1}\}$. Further, if $f(x)$ is a primitive polynomial, then, from the definition, the order of a root u must be $(2^n - 1)$ which is the maximum possible period of the output sequence given by the recurrence (5). In other words, if the characteristic equation of an order n linear recurrence sequence is the associated primitive polynomial, the sequence period becomes maximum. Now if we look at the problem given by the recurring relation $s_{i+4} = s_{i+1} + s_i$, it refers to the primitive polynomial of (3) and thus the sequence must be a maximum period sequence. If initial state is given by the first row of Table 2 (i.e., $(0, 0, 0, 1)$), the output binary sequence will then be given by the first 15 values of the corresponding fourth column of the same table, after which the sequence will be repeated. With this knowledge, we can then define the m-sequence as follows.

Definition 1: A maximal length sequence is a $(2^n - 1)$ length sequence that satisfies a linear recurrence, defined over $GF(2)$, given by any corresponding primitive polynomial of degree n.

C. Generating M-Sequence

The goal up to now has been to give a mathematical formula for m-sequences which we did via (5)-(7). In actual applications, of course, m-sequences are generated via primitive polynomials (as given in Table 3 as an example) by the more familiar method of linear feedback shift registers (LFSRs), depending upon the nature of the application. Here we present a brief account of two standard registers for generating m-sequences: the *additive form* which is related to linear recurrences, and the *multiplicative form* which is related to linear functions of field elements.

1) Additive shift register: Recall that a state of any periodic sequence s satisfying the recurrence (5) is an n-tuple of n consecutive values of s. If we consider two consecutive states, say

$$(s_i, s_{i+1}, ..., s_{i+n-1}) \rightarrow (s_{i+1}, s_{i+2}, ..., s_{i+n}) \quad (8)$$

then we see the second state is obtained by shifting all values to left by one bit and appending the value obtained from (5). This operation can be implemented in a LFSR which is shown in Fig. 1. Here $n = 4$ and the feedback is $s_{i+4} = s_{i+1} + s_i$. The type of shift register in Fig.1 is called an *additive* shift register. Its main characteristic is that it implements the recurrence exactly and the feedback bit is given by a linear combination of some of the bits. In practice, however, if there are a large number of taps, the fan-in to the additive function may be impractical. The multiplicative register alleviates this problem.

2) Multiplicative shift register: The *multiplicative* form, used to generate m-sequences, is based on multiplication in the finite field. With the knowledge that any linear function applied to its elements, i.e., to the register state, must yield an m-sequence, we get such a shift register as shown in Fig. 2. The top portion of the register generates the field elements (where we put in the initial element (0, 0, 0, 1)). These are then transformed to the sequence bits by some linear function. This function is often called a mask. The multiplicative register may be more practical, especially for simple linear functions, e.g., reading only a few bits of the register state. If we look at Table 2 again with initial state (0, 0, 0, 1) and u as the primitive root defined by $x^4 + x + 1$, then such a multiplicative register gives the successive states as sequential powers of u, i.e., u^i for $i = 0, 1, ..., 14$, after which the states get repeated.

IV. Identification of Primitive Polynomials with Reduced Complexity

As stated in the last section, it is necessary to check any polynomial whether it is a primitive one or not before employing it for constructing m-sequence purpose. Although a list is given in Table 3, it is, however, only one possible solution set and the reader should be reminded that there may exist other primitive polynomials as well for any degree n. Below, we discuss about a standard procedure to check any polynomial over binary fields whether it is a primitive one or not, and subsequently we propose a simplified method which would be helpful for most of the practical cases.

A. Standard Identification Procedure

To check whether any general polynomial $p(x)$ with order n, as given in (4), is a primitive polynomial or not over $GF(2)$, we can follow the stepwise algorithm as given in [8].
Step 1: To check whether $p(x)$ is irreducible.
(Divide $p(x)$ by any polynomial over $GF(2)$ of order $1 < m \leq \lfloor\sqrt{n}\rfloor$ and check all divisions have non-zero remainders.)
Step 2: To check whether irreducible $p(x)$ is primitive.

(Divide $h(x) = x^l - 1$ by $p(x)$ for the range $n \leq l < 2^n - 1$ and check all divisions have non-zero remainders.)

Although Step 1 is a basic scheme and one can apply Berlekamp's reducibility criterion [3] if available, Step 2, however, is relatively complex. To check for complexity, let us start with an example of code division multiple access (CDMA), a popular communication scheme used in many practical applications, where the polynomial over $GF(2)$ generating the short code for this purpose is of order 15 or higher. The computational cost of determining $r(x) = h(x)/p(x)$ is $\mathcal{O}(l)$, i.e., it increases linearly with the degree l of $h(x)$. The summation $\sum_{l=n}^{2^n-1} l$ yields approximately $\mathcal{O}(2^{2n})$. This, when applied to the CDMA example, gives a value of $\mathcal{O}(2^{30}) \approx 10^9$, which is quite high as expected. We propose a method that attempts to alleviates this problem with a simplified computational approach.

B. Proposed Simplified Method

We observe that reformulating Step 2 of the primitive polynomial check procedure enables to use a simple modulo arithmetic by reducing the computational cost to one division by $p(x)$ per iteration if the power of x can be reduced. This is possible simply by factorizing $h(x)$. Expressing $h(x) = x^l - 1$ as $(x-1)(x^{l-1} + x^{l-2} + ... + 1)$, we can alternately write

$$\frac{h(x)}{p(x)} = \frac{(x-1)(d(x)+1)}{p(x)} = \frac{(x-1)d(x)}{p(x)} + \frac{(x-1)}{p(x)} \qquad (9)$$

where $d(x) = x^{l-1} + x^{l-2} + ... + x$. Therefore, if we replace Step 2 with a check whether $d(x)/p(x)$ results in a remainder $r(x) \neq 1$ for all $n \leq l-1 \leq 2^n - 3$ and $r(x) = 1$ for $l - 1 = 2^n - 2$, then we can reach a simplified solution. Using mod 2 arithmetic, one can easily verify that this is identical to the original expression. As this procedure reduces the degree of numerator polynomial used for all iterations by one, it thus keeps the computation per iteration one division less than the standard one. It results in a total computational cost of almost $\mathcal{O}(2^n)$, which is significantly lesser than the standard procedure when the degree of the polynomial is high. We summarize the proposed algorithm in Table 4.

Note that another nice simple identification method is proposed in [9] which reformulates Step 2 as a check whether $x^l/p(x)$ results in a remainder $r(x) \neq 1$ for all $n \leq l \leq 2^n - 1$ with a more efficient use of intermediate results from the previous iterations. The students are encouraged to simulate both these two simplified methods along with the standard procedure to get an idea of complexity reduction for the examples like CDMA.

V. Concluding Remarks

We have discussed at length the mathematical construction of m-sequences via primitive polynomials and how to map the same when implementing in shift registers. As m-sequences are widely used in many applications, it is thus equally important to check whether a polynomial is primitive or not so as to get proper m-sequences. In this paper, we have proposed a fast method to identify a primitive polynomial over binary field which has a computational cost of $\mathcal{O}(2^n)$ in comparison with $\mathcal{O}(2^{2n})$ of the standard procedures. The proposed algorithm, in the context of the CDMA example, can be easily taken up by the undergraduate students as an assignment in any 'advanced communication' course in order to check the reduction in complexity.

TABLE IV
The Proposed Primitive Polynomial Identification Algorithm

```
1. Initialization:
Formulate $d(x) = x^{l-1} + x^{l-2} + ... + x$
and let $r(x) = d(x)$ mod $p(x)$
2. Iterative Processing:
(a) FOR $l - 1 = n$ to $2^n - 3$
        Evaluate $r(x)$
        IF $r(x) == 1$ (mod 2)
        THEN return 'non-primitive polynomial'
        BREAK-FORLOOP
        ELSE $l - 1 \leftarrow l - 1 + 1$ and Go to (a)
    END-FORLOOP
    Go to (b)
(b) FOR $l - 1 = 2^n - 2$
        IF $r(x) == 1$ (mod 2)
        THEN return 'primitive polynomial'
        ELSE return 'non-primitive polynomial'
```

VI. Appendix: An Easy Way to Generate Gaussian Sequences from Maximal Length Sequences

A frequently used sequence in communication and signal processing applications is Gaussian sequence, which can be easily generated from the discussed m-sequences and such an experiment can be taken up in the DSP laboratory to check its feasibility with fixed/floating point processors. First, with the help of the proposed simple primitive polynomial identification method, a $(2^n - 1)$ length m-sequence can be generated with a simple n bit additive LFSR. Next, such m-sequence generation technique can be repeated for 7 or 8 times with 7 or 8 different initial seeds. Note that the same primitive polynomial should be used in each case. The n bit binary values of any m-sequence can be converted to integer values using sign-magnitude rule of conversion (i.e., treating MSB as the sign and the rest as the magnitude). These integer values should then be normalized within the range $[+1, -1]$. If we now add the values of all these 7 or 8 m-sequences sample-wise, we would get almost a Gaussian sequence according to Central Limit Theorem [10]. This easy practical technique is used in almost all the communication system hardware to generate a Gaussian random variable.

References

[1] S. Blackburn, "A Note on Sequences with the Shift and Add Property," *Designs, Codes, and Crypt.*, vol. 9, pp. 251-256, 1996.

[2] I. D. Alanen and D. E. Knuth, "Tables of Finite Fields," *Sankhya*, Series A, vol. 26, pp. 305-328, 1964.

[3] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.

[4] E. J. Watson, "Primitive Polynomials (mod 2)," *Mathematics of Computation*, vol. 16, pp. 368-369, 1962.

[5] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*. Boston, MA: Kluwer Academic, 1987.

[6] D. E. Carter, "On the Generation of Pseudo-Noise Codes," *IEEE Trans. Aerosp. Electron. Sys.*, vol. 10, pp. 898-899, 1974.

[7] S. Golomb, *Shift Register Sequences, Revised edition*. Laguna Hills, CA: Aegean Park Press, 1982.

[8] D. E. Knuth, *The Art of Computer Progamming*. Reading, MA: Addison-Wesley, 1968.

[9] K. Krogsgaard and T. Karp, "Fast Identification of Primitive Polynomials over Galois Fields: Results from a Course Project," in *Proc. 2005 IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Philadelphia, PA, USA, 2005, pp. V553-V556.

[10] H. Tijms, *Understanding Probability: Chance Rules in Everyday Life*. Cambridge: Cambridge University Press, 2004.

Abhijit Mitra was born in Serampore, India, in 1975. He received the B.E.(Honors) degree from the Regional Engineering College, Durgapur, India, in 1997, the M.E.Tel.E. degree from Jadavpur University, India, in 1999 and the Ph.D. degree from the Indian Institute of Technology, Kharagpur, India, in 2004, all in electronics and communication engineering.

Since 2004, he has been with the Department of Electronics and Communication Engineering at the Indian Institute of Technology, Guwahati, India, as an Assistant Professor. He visited Indian Statistical Institute (ISI), Kolkata, as a Visiting Scientist during June-July and December 2007. He is the recipient of URSI Young Scientist Award and INAE Summer Fellowship for Young Engineering Teachers, both for 2008. He has also been elected as an Associate of Indian Academy of Sciences (IAS) for 2008-2011. His research interests include adaptive signal processing and signal processing applications in wireless communications with the primary emphasis on low complexity realizations.

Dr. Mitra has been a member of IEEE since 2003 and presently serves as a reviewer of many IEEE Transactions. He is also a member of Indian Science Congress Association and Institution of Electronics and Telecommunication Engineers, India.