Indian Journal of Science and Technology

Computing and Listing of Number of Possible m-sequence Generators of Order n

A. Ahmad[1*], S. S. Al-Busaidi[1], A. Al Maashri[1], M. Awadalla[1], M. A. K. Rizvi[2] and N. Mohanan[2]

[1]Faculty, Department of Electrical and Computer Engineering, College of Engineering, Sultan Qaboos University, P. O. Box 33, Postal Code 123; Muscat, Sultanate of Oman; afaq@squ.edu.om; albusaid@squ.edu.om, amaashari@squ.edu.om, medhatha@squ.edu.om

[2]M.Sc. Engineering Student, Department of Electrical and Computer Engineering, College of Engineering, SultanQaboos University, P. O. Box 33, Postal Code 123; Muscat, Sultanate of Oman; m099399@squ.edu.om, m100202@squ.edu.om

Abstract

Design of maximal length sequence (m-sequence) generators of order n has many controlling parameters. In the design process of the generators it is essential to ensure that the generator characteristic polynomial corresponds to a primitive polynomial. The complexity of the search problem of primitive polynomials of order n grows as n increases and hence restricts the listing of all parameters of m-sequence generators of order n. This paper presents a computational procedure to determine the number of possible generators of order n. The paper provides a list of all possible m-sequence generators for up to n = 100.

Keywords: m-Sequence, LFSR, Primitive Polynomial, Prime Factors, Mersenne Numbers, MATLAB.

1. Introduction and Problem Definition

Maximal length sequences (m-sequence) are also known as Pseudorandom Noise (PN) sequences. Maximal length sequences are of great importance in a variety of applications such as Direct Sequence Spread Spectrum (DSSS), Built-in Self-Test (BIST), Decryption – Encryption System (DES) and error detection, just to mention a few [1–12].

Systems in these applications typically use the basic hardware named Linear Feedback Shift Register (LFSR) to generate m-sequences [1–13]. A simple explanation of the LFSR structure and operation is given with respect to the structure shown in Figure 1 as follows. As shown in Figure 1 an LFSR is made up of two parts. These parts are a shift register and a feedback function. The shift registers, which can store one bit, are D – type Flip-Flops (FFs) that are connected as a chain. Moreover, each D-FF is also connected to a clock. At each clock cycle, a new bit is loaded into the first shift register, D-FF$_1$, of the D-FF chain. Considering that D-FF$_1$ is the leftmost register then the remaining bits within all other shift registers are shifted to the neighboring right register at every clock cycle. Furthermore, there exists a feedback function, which is simply the Exclusive-OR (XOR) logic operation of a number of bits that are held within a prescribed number of registers. Registers that are involved in the feedback operation are all connected to the XOR operator. Alternatively, the sequence of connections which are involved in the XOR operation logic is referred to a sequence of feedback taps (C_0 C_1 C_2 ….. C_i …. C_{n-1} C_n). The updated left most bit state of D-FF$_1$ is computed as a function of the existing feedback taps of the LFSR. The output of the LFSR is then read out at the output of rightmost shift register, one bit every clock cycle. Finally, the period of a shift register, p, is the length of the output sequence before it starts repeating [1–17].

*Corresponding author:
A. Ahmad (afaq@squ.edu.om)

Definition 1: An LFSR is a special type of Serial-In Serial-Out (SISO) shift register that, when clocked, propagates bits from the least significant to the most significant bit position through its constituent neighbouring registers, one bit every clock cycle.

Figure 1 shows an n-bit SISO shift register. The key element of SISO shift registers is the D-type FFs. The sequence $\{q_1, q_2, \ldots q_i, \ldots q_{n-1}, q_n\}$ represents the states of the FFs $\{D_1, D_2, \ldots, D_i, \ldots D_{n-1}, D_n\}$ respectively. Table 1 demonstrates an example of how LFSRs operates to generate an m-sequence. The m-sequence generator considered for this demonstration is shown in Figure 2.

Figure 2 depicts a 3-bit LFSR, which is constructed using an external XOR functional block. Note how the feedback – which is fed into the input to the first FF - is the result of exclusive-OR operation of the outputs of the second and third FFs. Table 1 visualizes the operation of the LFSR depicted in Figure 2. The table elaborates the next states (FF1_OUT, FF2_OUT, and FF3_OUT) and the output m-sequence S_i. The initial state of the LFSR is considered to be as follows, $q_1(0) = 1$, $q_2(0) = 0$, and $q_3(0) = 1$.

It is this feedback function that causes the LFSR to loop through a repetitive sequence of values. The choice of feedback connections, the initial state and the value of n, all determine the number of elements in a given sequence before the sequence repeats. The number of elements, in turn, determines the sequence length. This length is known as the periodicity, p, of the LFSR [2, 4, 6, 18–24].

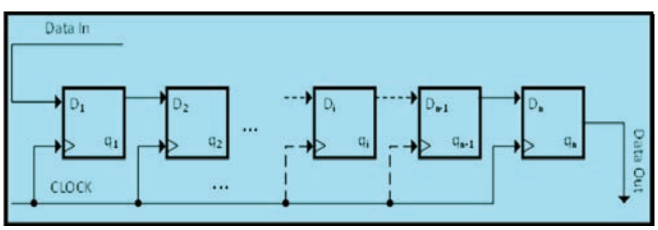

Figure 1. An n-bit SISO shift register.

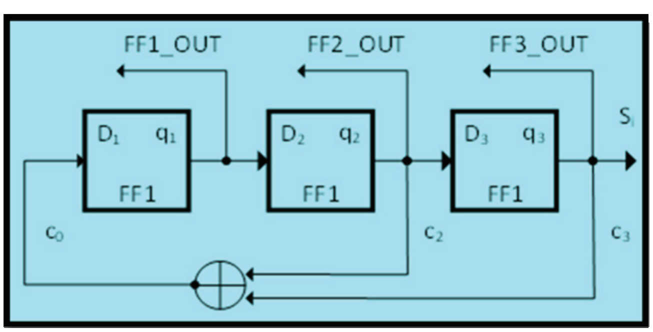

Figure 2. A 3-bit LFSR (External EOR type).

Definition 2: The period p of an n-bit LFSR may vary from $p = 1$ to 2^n-1.

Definition 3: The sequence having length of 2^n-1 is known as maximal length sequence (m-sequence).

Any binary sequence can be represented in a polynomial form over the Galois Field 2, GF(2). As such, the feedback connection vector of an LFSR can be represented by a polynomial that is technically referred to as a characteristic polynomial. Equations (1) define a general form of a characteristic polynomial which can be denoted as D(x).

$$\mathrm{D}(x) = (c_0 \times x^0) + (c_1 \times x^1) + \ldots + (c_n \times x^n) \tag{1}$$

For any LFSR structure of length n, a list of all characteristic polynomials can be generated to encompass every possible connection. To demonstrate this, Table 2 lists all the characteristic polynomials, NP, for an LFSR of length 3.

Theorem 1: In an n-bit LFSR a sequence generator can be referred to as an m-sequence generator only if its characteristic polynomial is primitive.

Table 1. Next state m-sequences for the structure of LFSR of Figure 2

Clock	q_1 (FF1_OUT)	q_2 (FF2_OUT)	q_3 (FF3_OUT)	Si
0	1	0	1	
1	1	1	0	
2	1	1	1	
3	0	1	1	…11101001110
4	0	0	1	
5	1	0	0	
6	0	1	0	
7	1	0	1	Repeats

Table 2. Number of possible generators (NP) and number of possible m-sequence generators in LFSR of Figure 2

n	*NP*	*NPP*
3	4	2
	$1+(x^3)$	$1+(x^2)+(x^3)$
	$1+(x^2)+(x^3)$	$1+(x)+(x^2)+(x^3)$
	$1+(x)+(x^3)$	
	$1+(x)+(x^2)+(x^3)$	

Table 2 also demonstrates that out of the 4 possible characteristic polynomials only 2 can be used to generate m-sequences. It can therefore be assumed that the number of primitive characteristic polynomials, NPP, is a subset of NP, by which the relation NPP < NP should hold.

An interesting relation on the complexity search for primitive characteristic polynomials can also be established. As n increases, the search complexity for primitive characteristic polynomials of order *n* grows exponentially. Consequently, the search success decays similar to the exponential decay of a signal. The nature of this search success decay rate is depicted in Figure 3. The search of primitive polynomials from the list of all possible combinations of generator polynomials can be prohibitive for large n. To this extent, this paper presents a computational efficient procedure of determining *NPP* when n is large.

2. Controlling Parameters for m-sequence Generators and Mathematics

The parameters governing the sequence period of a generator are: 1) the order n, 2) the initial state, and 3) the used characteristic polynomial. Table 1 shows an example where n = 3, initial state: [q_1 = 1, q_2 = 0, and q_3 = 1], and characteristic polynomials:

$$1+(x^3),$$
$$1+(x^2)+(x^3),$$
$$1+(x)+(x^3), \text{ and}$$
$$1+(x)+(x^2)+(x^3).$$

The sequence periods (*p*) generated by the characteristic polynomials shown above are 3, 7, 7 and 4, respectively. Note that each of the sequence generators is governed by a different state equation. In general, the state equation of a sequence generator is defined by Equation (2).

$$q(t+1)=[A]*q(t) \tag{2}$$

where, system matrix of the generator is [*A*], while q(*t*+1) and q(*t*) are next and present states of the generator, respectively. The structure of matrix [*A*] for an *n* order can be defined in Equation (3).

$$\begin{bmatrix} q_1(t+1) \\ q_2(t+1) \\ . \\ . \\ . \\ q_{n-1}(t+1) \\ q_n(t+1) \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & . & . & . & c_{n-1} & c_n \\ 1 & 0 & . & . & . & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & . & . & . & 0 & 0 \\ 0 & 0 & . & . & . & 1 & 0 \end{bmatrix} \begin{bmatrix} q_1(t) \\ q_2(t) \\ . \\ . \\ . \\ q_{n-1}(t) \\ q_n(t) \end{bmatrix} \tag{3}$$

where
$$c_j = \begin{cases} 0\, or\, 1, & for\, 1 \le j \le n-1 \\ 1, & for\, j = n \end{cases} \tag{4}$$

In Equation (4), the values of c_j show the existence or absence of a feedback connection from the j-th stage of the LFSR. Equation (2) can be written as:

$$[q(t+1)]=[A][q(t)] \tag{5}$$

If $[q]=[q(0)]$ represents a particular initial loading of the LFSR, then the sequence of states through which the LFSR will pass during successive times is given by

$$[q(t)],[A][q(t)],[A]^2[q(t)],[A]^3[q(t)],...$$

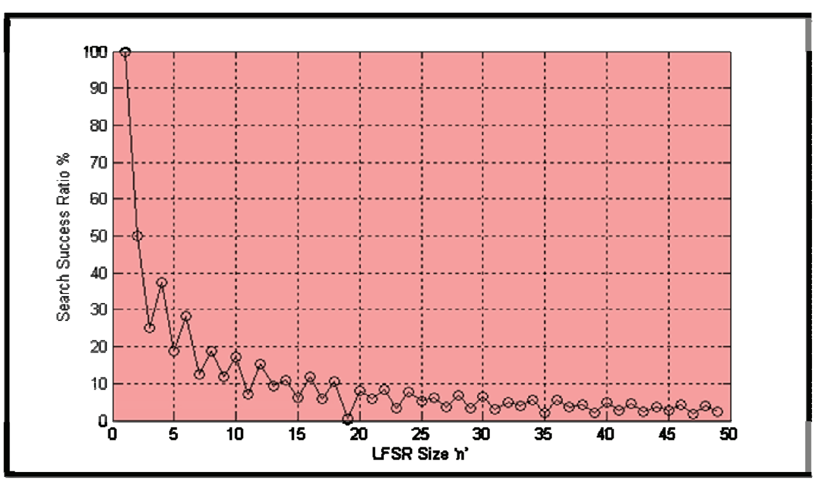

Figure 3. Search success rate for m-sequence generator in an-bit LFSR.

Let the matrix 'period' be the smallest integer p for which $[A]^p = I$, where I is an identity matrix. Then $[A]^p [q(t)] = [q(t)]$ for any non-zero initial vector $[q(0)]$, indicating the 'cycle length (or period)' of the LFSR is p.

The cycle length for $[q(0)] = 0$ is always 1, independent of matrix $[A]$. Thus, on the basis of this property of periodicity of LFSR and Equation (5), it follows that:

$$[q(t)] = [q(t+p)] = [A]^p [q(t)] \tag{6}$$

The following corollaries cover the periodicity properties of an LFSR and its relation with its corresponding primitive characteristic polynomial. These are used in the proposed algorithm.

Corollary 1: If $p = m = 2^n - 1$ is a prime number, then the characteristic polynomial corresponds to that connection of LFSR will be primitive, if and only if $[A]^m = 1$.

Corollary 2: If $p = m = 2^n - 1$ is not a prime number, and $[A]^{pi} = 1$, where p_i is a divisor of p, then the characteristic polynomial corresponds to that connection of LFSR cannot be primitive.

The determination of the primitive polynomial comprises of two folds; 1) the use of the Euler phi-function φ(.), and 2) the search for primes. The Euler function has the property that its value for an integer m is the product of the values of the Euler phi-function at the prime powers that occur in the factorization of m. The Euler phi-function can be computed on the basis of the prime factorization of m. The following theorems and Lemma are embodied in the proposed algorithm for finding primitive polynomials.

Lemma 1: There exists (exist) a prime divisor (or divisors) for every positive integer greater than one.

Theorem 2: If m is a composite integer, then m has a prime factor not exceeding the prime integer value of $\sqrt{m}$.

Theorem 3: Let $m = p_1^{a_1} p_2^{a_2} ... p_i^{a_i} ... p_k^{a_k}$ be the prime (p_i) of power (a_i) factorization of the positive integer m. Then

$$\varphi(\mathrm{m}) = \mathrm{m}(1 - 1/p_1)(1 - 1/p_2)...(1 - 1/p_k) \tag{7}$$

Theorem 4: The total number of possible primitive polynomials (NPP) of order n is given by

$$NPP = \frac{(m)}{n} \tag{8}$$

3. Computing Factors of *m*, ((*m*) and *NPP* [25–32]

Two algorithms – designated A.1 and A.2 – are used to compute factors of m, φ(m), and NPP. These two algorithms are presented in pseudo code format as follows:

	A.1 Algorithm: Computing prime factors of p p
	Input: n
	output: p, prime factors of p (p_i), number of prime factors (k), and exponents of each prime factor(e_i)
1	*Read n and do the following*
2	*Compute p* = 2^n-1;
3	*check is p prime or not, if yes GOTO step* 8;
4	*Find* p_i;
5	*Compute k;*
6	*Find* e_i;
7	*Return with p,* p_i*, k and* e_i
8	*Return with p,* p_i *= p, k = 1, and* e_i *= 1*

	A.2 Algorithm: Computing NPP
	Input: n
	output: NPP
1	*Function: A.1; generates p, k, and* e_i
2	*Use the Equations* (7) *and* (8) *to compute NPP*
3	*Return with NPP*

The factors for n = 1 to 100 are listed in Table 3 and 4; whereas the values of φ(m) and *NPP* are shown in Tables 5 and 6. Factorisations are given from smallest to largest factor, with a period '.' in the table 'f' indicates the number of factors.

4. Conclusions

We developed an algorithm and succeeded in getting the values of *NPP* for large values of n. This paper represent our efforts in transforming our observations into algorithms that are capable of determining the values of *NPP* for large n. Using these algorithm, we generated NPP lookup tables, which show the factors of m-sequence period m and *NPP*. As an example let n = 4, gives *NP* as 4 while NPP is computed as 2. The factors for m = 15 suggests that the periods of sequence generators of size n = 4 may be any value from factors {1, 3, 5, 15}. Only two of the generators are giving m-sequence. These tables offer fruitful information that can be utilized to judge whether

Table 3. The Factors of *m* (for *n* = 1 to 50)[Exponents are represented with '^']

n	*m*	*p*	*f*	Factors
1	1	Y	1	1
2	3	Y	1	3
3	7	Y	1	7
4	15	N	2	3×5
5	31	Y	1	31
6	63	N	3	3×3×7 → 3^2.7
7	127	Y	1	127
8	255	N	3	3×5×17
9	511	N	2	7×73
10	1023	N	3	3×11×31
11	2047	N	2	23×89
12	4095	N	5	3×3×5×7×13 → 3^2.5.7.13
13	8191	Y	1	8191
14	16384	N	3	3×43×127
15	32768	N	3	7×31×151
16	65536	N	4	3×5×17×257
17	131072	Y	1	131071
18	262144	N	6	3×3×3×7×19×73 → 3^3.7.19.73
19	524288	Y	1	524287
20	1048576	N	6	3×5×5×11×31×41 → 3.5^2.11.31.41
21	2097152	N	4	7×7×127×337 → 7^2.127.337
22	4194304	N	4	3×23×89×683
23	8388608	Y	2	47×178481
24	16777216	N	7	3×3×5×7×13×17×241 → 3^2.5.7.13.17.241
25	33554432	N	3	31×601×1801
26	67108864	N	3	3×2731×8191
27	134217728	N	3	7×73×262657
28	268435456	N	6	3×5×29×43×113×127
29	536870912	Y	3	233×1103×2089
30	1073741824	N	7	3×3×7×11×31×151×331 → 3^2.7.11.31.151.331
31	2147483648	Y	1	2147483647
32	4294967296	N	5	3×5×17×257×65537
33	8589934592	N	4	7×23×89×599479
34	17179869184	N	3	3×43691×131071
35	34359738368	N	4	31×71×127×122921
36	68719476736	N	10	3×3×3×5×7×13×19×37×73×109 → 3^3.5.7.13.19.37.73.109
37	1.37439E+11	Y	2	223×616318177
38	2.74878E+11	N	3	3×174763×524287
39	5.49756E+11	N	4	7×79×8191×121369
40	1.09951E+12	N	8	3×5×5×11×17×31×41×61681→3.5^2.11.17.31.41.61681
41	2.19902E+12	Y	2	13367×164511353
42	4.39805E+12	N	8	3×3×7×7×43×127×337×5419 → 3^2.7^2.43.127.337.5419
43	8.79609E+12	Y	3	431×9719×2099863
44	1.75922E+13	N	7	3×5×23×89×397×683×2113
45	3.51844E+13	N	6	7×31×73×151×631×23311
46	7.03687E+13	N	4	3×47×178481×2796203

Table 3. (*Continued*)

47	1.40737E+14	Y	3	2351×4513×13264529
48	2.81475E+14	N	10	3×3×5×7×13×17×97×241×257×673 → 3^2.5.7.13.17.97.241.257.673
49	5.6295E+14	N	2	127×4432676798593
50	1.1259E+15	N	7	3×11×31×251×601×1801×4051

Table 4. The Factors of *m* (for *n* = 51 to 100) [Exponents are represented with'^']

n	*m*	*p*	*f*	Factors
51	2.2518E+15	N	5	7×103×2143×11119×131071
52	4.5036E+15	N	7	3×5×53×157×1613×2731×8191
53	9.0072E+15	Y	3	6361×69431×20394401
54	1.80144E+16	N	9	3×3×3×3×7×19×73×87211×262657 →3^4.7.19.73.87211.262657
55	3.60288E+16	N	6	23×31×89×881×3191×201961
56	7.20576E+16	N	8	3×5×17×29×43×113×127×15790321
57	1.44115E+17	N	4	7×32377×524287×1212847
58	2.8823E+17	N	6	3×59×233×1103×2089×3033169
59	5.76461E+17	Y	2	179951×3203431780337
60	1.15292E+18	N	13	3×3×5×5×7×11×13×31×41×61×151×331×1321 →3^2.5^2.7.11.13.31.41.61.151.331.1321
61	2.30584E+18	Y	1	2305843009213693951
62	4.61169E+18	N	3	3×715827883×2147483647
63	9.22337E+18	N	7	7×7×73×127×337×92737×649657 7^2.73.127.337.92737.649657
64	1.84467E+19	N	7	3×5×17×257×641×65537×6700417
65	3.68935E+19	N	3	31×8191×145295143558111
66	7.3787E+19	N	9	3×3×7×23×67×89×683×20857×599479 →3^2.7.23.67.89.683.20857.599479
67	1.47574E+20	Y	2	193707721×761838257287
68	2.95148E+20	N	7	3×5×137×953×26317×43691×131071
69	5.90296E+20	N	4	7×47×178481×10052678938039
70	1.18059E+21	N	9	3×11×31×43×71×127×281×86171×122921
71	2.36118E+21	Y	3	228479×48544121×212885833
72	4.72237E+21	N	14	3×3×3×5×7×13×17×19×37×73×109×241×433×38737 →3^3.5.7.13.17.19.37.73.109.241.433.38737
73	9.44473E+21	Y	3	439×2298041×9361973132609
74	1.88895E+22	N	5	3×223×1777×25781083×616318177
75	3.77789E+22	N	7	7×31×151×601×1801×100801×10567201
76	7.55579E+22	N	7	3×5×229×457×174763×524287×525313
77	1.51116E+23	N	4	23×89×127×581283643249112959
78	3.02231E+23	N	8	3×3×7×79×2731×8191×121369×22366891 →3^2.7.79.2731.8191.121369.22366891
79	6.04463E+23	Y	3	2687×202029703×1113491139767
80	1.20893E+24	N	10	3×5×5×11×17×31×41×257×61681×4278255361 →3.5^2.11.17.31.41.257.61681.4278255361
81	2.41785E+24	N	6	7×73×2593×71119×262657×97685839
82	4.8357E+24	N	5	3×83×13367×164511353×8831418697
83	9.67141E+24	Y	2	167×57912614113275649087721

Table 4. (*Continued*)

84	1.93428E+25	N	14	3×3×5×7×7×13×29×43×113×127×337×1429×5419×14449 →3^2.5.7^2.13.29.43.113.127.337.1429.5419.14449
85	3.86856E+25	N	3	31×131071×9520972806333758431
86	7.73713E+25	N	5	3×431×9719×2099863×2932031007403
87	1.54743E+26	N	6	7×233×1103×2089×4177×9857737155463
88	3.09485E+26	N	10	3×5×17×23×89×353×397×683×2113×2931542417
89	6.1897E+26	Y	1	618970019642690137449562111
90	1.23794E+27	N	13	3×3×3×7×11×19×31×73×151×331×631×23311×18837001 →3^3.7.11.19.31.73.151.331.631.23311.18837001
91	2.47588E+27	N	5	127×911×8191×112901153×23140471537
92	4.95176E+27	N	9	3×5×47×277×1013×1657×30269×178481×2796203
93	9.90352E+27	N	3	7×2147483647×658812288653553079
94	1.9807E+28	N	6	3×283×2351×4513×13264529×165768537521
95	3.96141E+28	N	5	31×191×524287×420778751×30327152671
96	7.92282E+28	N	13	3×3×5×7×13×17×97×193×241×257×673×65537×22253377 →3^2.5.7.13.17.97.193.241.257.673.65537.22253377
97	1.58456E+29	Y	2	11447×13842607235828485645766393
98	3.16913E+29	N	5	3×43×127×4363953127297×4432676798593
99	6.33825E+29	N	8	7×23×73×89×199×153649×599479×33057806959
100	1.26765E+30	N	14	3×5×5×5×11×31×41×101×251×601×1801×4051×8101×268501 →3.5^3.11.31.41.101.251.601.1801.4051.8101.268501

Table 5. The Values of *NPP* (n= 1 to 50) [Exponents are represented with '^']

n	$NP=2^{n-1}$	$\varphi(m)$	*NPP*
1	1	0	0
2	2	2	1
3	4	6	2
4	8	8	2
5	16	30	6
6	32	24	4
7	64	126	18
8	128	128	16
9	256	432	48
10	512	600	60
11	1024	1936	176
12	2048	1152	96
13	4096	8190	630
14	8192	10584	756
15	16384	27000	1800
16	32768	32768	2048
17	65536	131070	7710
18	131072	62208	3456
19	262144	524286	27594
20	524288	384000	19200

Table 5. (*Continued*)

21	1048576	1524096	72576
22	2097152	2640704	120032
23	4194304	8210080	356960
24	8388608	4423680	184320
25	16777216	32400000	1296000
26	33554432	44717400	1719900
27	67108864	1.13E+08	4202496
28	134217728	88510464	3161088
29	268435456	5.34E+08	18407808
30	536870912	3.56E+08	11880000
31	1073741824	2.15E+09	69273666
32	2147483648	2.15E+09	67108864
33	4294967296	6.96E+09	2.11E+08
34	8589934592	1.15E+10	3.37E+08
35	17179869184	3.25E+10	9.29E+08
36	34359738368	1.16E+10	3.22E+08
37	68719476736	1.37E+11	3.7E+09
38	1.37439E+11	1.83E+11	4.82E+09
39	2.74878E+11	4.65E+11	1.19E+10
40	5.49756E+11	3.79E+11	9.47E+09
41	1.09951E+12	2.2E+12	5.36E+10
42	2.19902E+12	1.39E+12	3.3E+10
43	4.39805E+12	8.77E+12	2.04E+11
44	8.79609E+12	5.52E+12	1.25E+11
45	1.75922E+13	2.85E+13	6.34E+11
46	3.51844E+13	4.59E+13	9.98E+11
47	7.03687E+13	1.41E+14	2.99E+12
48	1.40737E+14	7.31E+13	1.52E+12
49	2.81475E+14	5.59E+14	1.14E+13
50	5.6295E+14	6.56E+14	1.31E+13

Table 6. The Values of *NPP* (n = 51 to 100)[Exponents are represented with '^']

n	$NP\text{-}2^{n-1}$	$\varphi(m)$	*NPP*
51	1.1259E+15	1.91E+15	3.75E+13
52	2.2518E+15	2.34E+15	4.5E+13
53	4.5036E+15	9.01E+15	1.7E+14
54	9.0072E+15	2.85E+15	5.28E+13
55	1.80144E+16	3.29E+16	5.99E+14
56	3.60288E+16	3.35E+16	5.99E+14
57	7.20576E+16	1.24E+17	2.17E+15
58	1.44115E+17	1.88E+17	3.24E+15

Table 6. (*Continued*)

59	2.8823E+17	5.76E+17	9.77E+15
60	5.76461E+17	2.17E+17	3.61E+15
61	1.15292E+18	2.31E+18	3.78E+16
62	2.30584E+18	3.07E+18	4.96E+16
63	4.61169E+18	6.61E+18	1.05E+17
64	9.22337E+18	9.21E+18	1.44E+17
65	1.84467E+19	3.57E+19	5.49E+17
66	3.68935E+19	2.61E+19	3.96E+17
67	7.3787E+19	1.48E+20	2.2E+18
68	1.47574E+20	1.56E+20	2.3E+18
69	2.95148E+20	4.95E+20	7.18E+18
70	5.90296E+20	6.59E+20	9.42E+18
71	1.18059E+21	2.36E+21	3.33E+19
72	2.36118E+21	7.46E+20	1.04E+19
73	4.72237E+21	9.42E+21	1.29E+20
74	9.44473E+21	1.25E+22	1.69E+20
75	1.88895E+22	3.11E+22	4.14E+20
76	3.77789E+22	4E+22	5.27E+20
77	7.55579E+22	1.42E+23	1.84E+21
78	1.51116E+23	1.14E+23	1.46E+21
79	3.02231E+23	6.04E+23	7.65E+21
80	6.04463E+23	4.15E+23	5.19E+21
81	1.20893E+24	2.04E+24	2.52E+22
82	2.41785E+24	3.18E+24	3.88E+22
83	4.8357E+24	9.61E+24	1.16E+23
84	9.67141E+24	4.31E+24	5.13E+22
85	1.93428E+25	3.74E+25	4.4E+23
86	3.86856E+25	5.15E+25	5.98E+23
87	7.73713E+25	1.32E+26	1.52E+24
88	1.54743E+26	1.46E+26	1.66E+24
89	3.09485E+26	6.19E+26	6.95E+24
90	6.1897E+26	2.56E+26	2.84E+24
91	1.23794E+27	2.45E+27	2.7E+25
92	2.47588E+27	2.57E+27	2.79E+25
93	4.95176E+27	8.49E+27	9.13E+25
94	9.90352E+27	1.31E+28	1.4E+26
95	1.9807E+28	3.81E+28	4.01E+26
96	3.96141E+28	2.05E+28	2.13E+26
97	7.92282E+28	1.58E+29	1.63E+27
98	1.58456E+29	2.05E+29	2.09E+27
99	3.16913E+29	5.04E+29	5.09E+27
100	6.33825E+29	3.65E+29	3.65E+27

the generators need to be used or not. Hence these tables are of great help for engineers, scientists and researchers practicing / working their skills in the fields of DSSS, BIST and data security.

7. Acknowledgement

The acknowledgements are due to authorities of Sultan Qaboos University (Sultanate of Oman) for providing generous research support grants and environments for carrying out the research works.

8. References

1. Ahmad A (2012). Better PN generators for CDMA application – a Verilog-HDL implementation approach, International Journal of Information Engineering (IJIE), vol 2(1), 6–11.
2. Ahmad A, Al-Abri D et al. (2012). Adding pseudo-random test sequence generator in the test simulator for DFT approach, Journal of Computer Technology and Applications (JCTA), vol 3(7), 463–470.
3. Ahmad A, and Hayat L (2011). Selection of polynomials for cyclic redundancy check for the use of high speed embedded – an algorithmic procedure, Transactions on Computers (WSEAS), vol 10(1), 16–20.
4. Ahmad A (2010). A simulation experiment on a built-in self test equipped with pseudorandom test pattern generator and Multi-Input Shift Register (MISR), International Journal of VLSI Design & Communication Systems, vol 1, No. 4, 1–12.
5. Ahmad A, and Al-Balushi J (2009). How to design an effective Serial Input Shift Register (SISR) for data compression process of built-in self-test methodology, Proceedings 4th International Design and Test Workshop (IDT'09) held at King Abdul Aziz City for Science and Technology (KAASAT), Riyadh, Saudi Arabia. 372–379.
6. Al-Naamany A M, and Ahmad A (2003). Development of a strong stream ciphering technique using non-linear fuzzy logic selector, Mobile and Wireless Communications, IFIP — The International Federation for Information Processing, vol 106, 199–206.
7. Ahmad A, Al-Musharafi M J et al. (2002). Design and study of a strong stream crypto-system model for e-commerce, Proceeding ICCC '02 Proceedings of the 15th International Conference on Computer Communication, vol 1, 619–630.
8. Jamil T, and Ahmad A (2002). An investigation in to the application of linear feedback shift registers for steganography, Proceedings IEEE SoutheastCon2002, Columbia, SC, USA, 239–244.
9. Ahmad A, Al-Musharafi M J et al. (2001). An NLFSR based sequence generator for stream ciphers, Proceedings (SETA'01) - An International Conference on Sequences & Their Applications, Norway (Bergen), 11–13.
10. Golic J D (2000). Cryptanalysis of three mutually clock-controlled stop/go shift registers, IEEE Transactions on Information Technology, vol 46, No. 3, 1081–1090.
11. Ahmad A (1997). Achievement of higher testability goals through the modification of shift register in LFSR based testing, International Journal of Electronics (UK), vol 82, No. 3, 249–260.
12. Chen H W, Aine C J E et al. (1996). Nonlinear analysis of biological systems using short m-sequences and sparse-simulation techniques, Annals of Biomedical Engineering, vol 24, 513–536.
13. Golomb S W (1982). Shift Register Sequences, Aegean Park Press, Revised Edition.
14. Ahmad A, Nanda N K et al. (1990). Are primitive polynomials always best in signature analysis?, IEEE Design & Test of Computers (USA), vol 7, No. 4, 36–38.
15. Blum L, Blum M et al. (1986). A simple unpredictable pseudo-random number generator, SIAM Journal of Computing, vol 15, No. 2, 364–383.
16. Ahmad A, Nanda N K et al. (1988). A critical role of primitive polynomials in an LFSR based testing technique, IEE Electronics Letters (UK), vol 24(15), 953–955.
17. Ahmad A, Al-Busaidi S S et al. (2013). Study on cyclic cross-correlation behavior of maximal length pseudo-random binary sequences, Indian Journal of Industrial and Applied Mathematics (Taylor & Francis), vol 4(1), 33–43.
18. Ahmad A, and Al-Abri D (2012). Design of a pseudo-random binary code generator via a developed simulation model, International Journal on Information Technology (ACEEE - Journal), vol 2(1), 33–36.
19. Ahmad A (2011). Investigation of some quite interesting divisibility situations in a signature analyzer implementation, Transactions on Circuits and Systems (WSEAS), vol 10 (9), 299–308.
20. Ahmad A (1998). An algorithmic generation of sparse primitive polynomials of order n, International Wireless and Telecommunication Symposium (IWATS'98), Shah Alam (Malaysia).
21. Ahmad A, and Elabdalla A M (1997). An efficient method to determine linear feedback connections in shift registers that generate maximal length pseudo-random up and down binary sequences, Computer & Electrical Engineering (Elsevier), vol 23(1), 33–39.
22. Ahmad A, and Al-Maashri A (2008). Investigating some special sequence length generated through an external exclusive-NOR type LFSRs, International Journal Electrical and Computer Engineering, vol 34(1), 270–280.

23. Ahmad A (2003). Realization of a stream cipher with better security and higher reliability goals, 2nd International Conference on Quality, Reliability and Information Technology - Trends and Future Directions, 18–21, 77–78.
24. Ahmad A, Al-Busaidi S et al. (2003). Measurement techniques of LFSR sequences, Proceedings International Symposium on Wireless Systems and Networks (ISWSN'03), King Fahad University of Petroleum - Dahran, Kingdom of Saudi Arabia, 1–5.
25. Ahmad A, Nanda N K et al. (1989). The use of irreducible characteristic polynomials in an LFSR based testing of digital circuits, Proceedings of 4th IEEE International Conference of Region 10 (TENCON-89), 494–496.
26. Ahmad A, Ahmed Al-Mashari et al. (2009). On locking conditions in M-sequence generators for the use in digital watermarking, Proceedings International Conference on Methods and Models in Computer Science (ICM2CS09) held at School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi, India, 111–115.
27. Ahmad A, Al-Musharafi M J et al. (2001). Study and implementation of properties of m-sequences in MATLAB-SIMULINK – A pass / fail test tool for designs of random generators, Proceedings IEEE / IEE International Conference on Communication, Computer and Power (ICCCP'01), Oman, 191–196.
28. Ahmad A, Al-Musharafi M J et al. (2002). Study and implementation of properties of m-sequences in MATLAB-SIMULINK – A pass / fail test tool for designs of random generators, Journal of Scientific Research – Science and Technology, vol 7 (part 1), 147–156.
29. Al-Lawati A, and Ahmad A (2004). Realization of a simplified controllability computation procedure – a MATLAB-SIMULINK based tool, Sultan Qaboos University Journal for Scientific Research - Science and Technology, Oman, vol 8, 131–143.
30. Ahmad A, and Ruelens D (2013). Development of digital logic design teaching tool using MATLAB & SIMULINK, IEEE Technology and Engineering Education (ITEE), vol 8, No. 1, 7–12.
31. Ahmad A, Ruelens D et al. (2013). Development of verification tool for minimal Boolean equation, IEEE Technology and Engineering Education (ITEE), vol 8, No. 3.
32. MATLAB: Available from: http://www.mathworks.com/