International Journal of Computer Science and Informatics

Volume 3 | Issue 4 Article 11

April 2014

LINEAR FEEDBACK SHIFT REGISTER BASED UNIQUE RANDOM NUMBER GENERATOR

HARSH KUMAR VERMA
Department of Computer Science and Engineering, Dr. B. R. Ambedkar National Institute of Technology Jalandhar, India, vermah@nitj.ac.in

RAVINDRA KUMAR SINGH
Department of Computer Science and Engineering, Dr. B. R. Ambedkar National Institute of Technology Jalandhar, India, ravindra1987singh@gmail.com

Follow this and additional works at: https://www.interscience.in/ijcsi

 Part of the Computer Engineering Commons, Information Security Commons, and the Systems and Communications Commons

Recommended Citation
VERMA, HARSH KUMAR and SINGH, RAVINDRA KUMAR (2014) "LINEAR FEEDBACK SHIFT REGISTER BASED UNIQUE RANDOM NUMBER GENERATOR," *International Journal of Computer Science and Informatics*: Vol. 3 : Iss. 4 , Article 11.
Available at: https://www.interscience.in/ijcsi/vol3/iss4/11

This Article is brought to you for free and open access by Interscience Research Network. It has been accepted for inclusion in International Journal of Computer Science and Informatics by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

LINEAR FEEDBACK SHIFT REGISTER BASED UNIQUE RANDOM NUMBER GENERATOR

HARSH KUMAR VERMA[1] & RAVINDRA KUMAR SINGH[2]

[1,2]Department of Computer Science and Engineering, Dr. B. R. Ambedkar National Institute of Technology Jalandhar, India
E-mail : vermah@nitj.ac.in, ravindra1987singh@gmail.com

Abstract - Linear Feedback Shift Register based Unique Random Number Generator is an enhancement of Random Number generator with the additional property that any number generated by a unique random number generator can't be duplicated. As per users demand for not duplicated random numbers in some applications like transferring a random number over the network on the behalf of actual character of the message for security point of view, existence of unique random number generators are very essential. In this paper LFSR [1] (Linear Feedback Shift Register) is used to implement the proposed concept of unique random number generator. Using LFSR is a faster approach for generating random sequences because it requires only X-OR operations and shift registers that's why its implementation is very easy in software as well as in hardware [2, 3]. We can easily modify the LFSR and produce different random sequences. So it is the best option for using LFSR in unique random number generator.

Keywords - *Random Number Generators, Linear Feedback Shift Register (LFSR), Unique Random Number Generators.*

I. INTRODUCTION

A random number generator is a computational or physical device (or software) designed to generate a sequence of numbers or symbols that lack any pattern [4]. There are two principal methods used to generate random numbers. One measures some physical phenomenon that is expected to be random and then compensates for possible biases in the measurement process. The other uses computational algorithms that produce long sequences of apparently random results, which are in fact completely determined by a shorter initial value, known as a seed or key. Coin flipping is an example of the first principal while Pseudo Random Number is the example of the second principal [4].

A "random number generator" based solely on deterministic computation cannot be regarded as a "true" random number generator, since its output is inherently predictable. A true random system would have no restriction on the same item appearing two, three or more times in succession or in the sequence of numbers [5]. Whereas Unique Random number generator generates the sequence of numbers in which no one can be duplicated.

In the field of cryptography, Random number generators are very useful as it facilitate the ability to run the same sequence of random numbers again by starting from the same *random seed*. So long as the *seed* is secret. Sender and receiver can generate the same set of numbers automatically to use as keys [5]. Random number generators have a vital applications in gambling, completely randomized design, statistical sampling, computer simulation, and other areas where producing an unpredictable result is desirable [6].

Many such applications of randomness have led to the development of several different methods for generating random data. Many of these have existed since ancient times; including dice, coin flipping, and the shuffling of playing cards [7]. But those are not sufficient enough to fulfill are requirements so now a day's some other techniques are also used like: linear congruential generator, middle square method, probability density function, inversion method, acceptance-rejection method, hash function based random number generator, linear feedback shift register method etc [4].

II. LINEAR FEEDBACK SHIFT REGISTER

A Linear Feedback Shift Register is a shift register whose input state is a linear function of its previous state [8]. The only linear functions of single bits are XOR and inverse-XOR; thus it is a shift register whose input bit is driven by the exclusive-or (XOR) of some bits of the overall shift register value [9].

The L-bit initial value of LFSR is called **seed** where L is called its length, the stream values produced by the register is completely determined by previous state [10]. It can produce various random sequences by varying the **taps** [11].The bit position that affects next state is called tap. This is illustrated as follows [9].

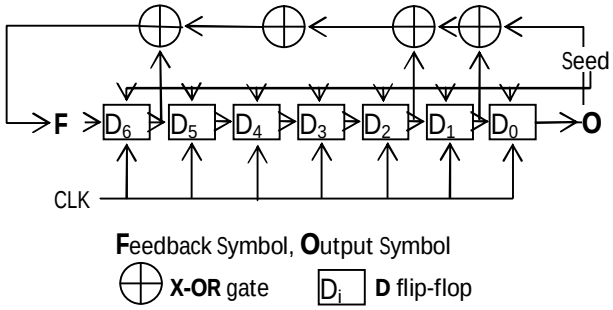

Fig. 1 : LFSR for Tap = [0, 1, 2, 4, 6], length (L) = 7

In this circuit, at each pulse, the state of the flip-flop is shifted to the next one down the line and also computes Boolean function of the state of the flip-flops [9, 12].

If Tap = {0, 2, 3, 5, 6} and Seed = 51 whose binary equivalent is 0110011. Then-

Table I : Generated Sequence By Lfsr

Feedback Symbol	State of Shift Register							Output Symbol
	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
	0	**1**	***1***	**0**	**0**	***1***	**1**	
0	0	0	*1*	1	0	*0*	1	1
0	0	0	*0*	1	1	*0*	0	1
0	0	0	*0*	0	1	*1*	0	0
1	1	0	*0*	0	0	*1*	1	0
0	0	1	*0*	0	0	*0*	1	1
0	0	0	*1*	0	0	*0*	0	1
0	0	0	*0*	1	0	*0*	0	0
1	1	0	*0*	0	1	*0*	0	0
0	0	1	*0*	0	0	*1*	0	0
1	1	0	*1*	0	0	*0*	1	0
0	0	1	*0*	1	0	*0*	0	1
0	0	0	*1*	0	1	*0*	0	0
1	1	0	*0*	1	0	*1*	0	0
0	0	1	*0*	0	1	*0*	1	0
1	1	0	*1*	0	0	*1*	0	1
1	1	1	*0*	1	0	*0*	1	0
0	0	1	*1*	0	1	*0*	0	1
0	0	0	*1*	1	0	*1*	0	0
1	1	0	*0*	1	1	*0*	1	0
0	0	1	*0*	0	1	*1*	0	1
0	0	0	*1*	0	0	*1*	1	0
1	1	0	*0*	1	0	*0*	1	1
1	1	1	*0*	0	1	*0*	0	1
1	1	1	*1*	0	0	*1*	0	0
0	0	1	*1*	1	0	*0*	1	0
1	1	0	*1*	1	1	*0*	0	1
1	1	1	*0*	1	1	*1*	0	0
0	0	1	*1*	0	1	*1*	1	0
1	1	0	*1*	1	0	*1*	1	1
1	1	1	*0*	1	1	*0*	1	1
1	1	1	*1*	0	1	*1*	0	1

Feedback Symbol	State of Shift Register							Output Symbol
	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
	0	**1**	***1***	**0**	**0**	***1***	**1**	
1	1	1	*1*	1	0	*1*	1	0
0	0	1	*1*	1	1	*0*	1	1
0	0	0	*1*	1	1	*1*	0	1
0	0	0	*0*	1	1	*1*	1	0
1	1	0	*0*	0	1	*1*	1	1
1	1	1	*0*	0	0	*1*	1	1
1	1	1	*1*	0	0	*0*	1	1
1	1	1	*1*	1	0	*0*	0	1
1	1	1	*1*	1	1	*0*	0	0
0	0	1	*1*	1	1	*1*	0	0
1	1	0	*1*	1	1	*1*	1	0
0	0	1	*0*	1	1	*1*	1	1
0	0	0	*1*	0	1	*1*	1	1
0	0	0	*0*	1	0	*1*	1	1
0	0	0	*0*	0	1	*0*	1	1
0	0	0	*0*	0	0	*1*	0	1
0	0	0	*0*	0	0	*0*	1	0
1	1	0	*0*	0	0	*0*	0	1
1	1	1	*0*	0	0	*0*	0	0
0	0	1	*1*	0	0	*0*	0	0
1	1	0	*1*	1	0	*0*	0	0
0	0	1	*0*	1	1	*0*	0	0
1	1	0	*1*	0	1	*1*	0	0
0	0	1	*0*	1	0	*1*	1	0
1	1	0	*1*	0	1	*0*	1	1
1	1	1	*0*	1	0	*1*	0	1
1	1	1	*1*	0	1	*0*	1	0
0	0	1	*1*	1	0	*1*	0	1
0	0	0	*1*	1	1	*0*	1	0
1	1	0	*0*	1	1	*1*	0	1
1	1	1	*0*	0	1	*1*	1	0
0	**0**	**1**	***1***	**0**	**0**	***1***	**1**	1

Generated sequence will be (1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 1 0 0 1 0 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1)

LFSR can also be used as a random number generator [13, 14]. By using the range (R) of random number it can be determined that how many bits (B) will be grouped together to represent a random number by the formula-

$$2^B >= R > 2^{B-1} \quad (1)$$

If the range of random number is 0 to 63 then number of bits will be 6 to represent the random number. Generated sequences of random numbers are repeated to generate the required count of random numbers [15].

The binary value of produced random number from above LSFR is 110011, 000010, 001010, 010110,

010011, 101101, 111000, 111110, 100000 and 011010 while 101 is left, it will be used in next repetition on the top of generated sequence. So next values will be 101110, 011000, 010001, 010010, 110010, 011101, 101111, 000111, 110100, 000011 and 010101, these numbers will be repeated again and again to generate the specified count of random numbers [15].

But the fastest moving era of computer science demands the non repeating random numbers in some applications. At those situations the existing approach can not satisfy the demand, that's why LFSR Based Unique Random Number Generators came in the focus.

III. LFSR BASED UNIQUE RANDOM NUMBER GENERATOR

The proposed algorithm ensures to generate the specified count of unique random numbers by using linear feedback shift register with some associated rules.

A. Basic Terms

There are some basic terms required in LFSR based Unique Random Number Generator.

1) Required Registers(R)

Number of required registers in LFSR based Unique Random Number Generator is depends on the number of letters exist in that number system. If there are N letters in any number system then it will need "R" registers. Where-

$$2^R >= N > 2^{R-1} \quad (2)$$

Ex: ascii codes require 7 registers, (2^7 = 128). Because there are total 128 letters in ascii number system. A letter may be alphabet (case sensitive), numeral or a special symbol.

2) Relevant Values

A relevant value is assigned to all letters exist in the number system. Ex: ascii codes are that relevant numbers for ascii number system which contains 128 letters.

3) Seed Sequence

Seed sequence is also depends upon the keyword. Keyword's relevant values stores in first come first serve manner by removing duplicates is called **Seed Sequence**.

4) Tap Sequence

Tap sequence depends upon the keyword. Mod_R of keyword's relevant values stores in first come first serve manner by removing duplicates is called **Tap Sequence**.

5) Length (L) of Unique Random Number

Length of unique random number is totally depending on the total count of required unique random number (U), which can be calculated by the given formula-

$$2^L >= U > 2^{L-1} \quad (3)$$

B. Algorithm

These are the following steps to produce unique random numbers-

1) Unique Random Number Generator *(Keyword, N, OS[N][2], U, URnum[U], R, L, i, j, k, m, x, y, z, w, flag)*

This algorithm is used for generating U count of unique random numbers by using user's Keyword. URnum is a linear array of U length for storing U cout of unique random number where OS is a 2-D array which stores output symbols of LFSR.

{

1. [Initialize]
   y = 0, z = 0, w = 0, flag = 0;
2. [Find the number of **Required Registers**(**R**) of that number system.]
   $2^R >= N > 2^{R-1}$;
3. [Find the **Length (L)** of Unique Random Number.]
   $2^L > U >= 2^{L-1}$;
4. Find the **Seed Sequence** of that keyword.
5. Using mod_R operations find out the **Tap Sequence** of the keyword.
6. Fetch the first value from **Seed Sequence** for the **Seed** of **LFSR** and delete it from the **Seed Sequence**.
7. Use the **Tap Sequence** as the **Tap** for the LFSR.
8. Generate the Output symbols from **LFSR** (described detailed in next algorithm) until the seed gets repeated. And store it in OS**[k][0]** where **k** = 0 to number of output symbols (x) generated by LFSR while OS[k][1] will be 0 for all.
9. do

{

```
z = flag = 0;
if (OS[y][1] = = 0)
{
OS[y][1] = 1;
m = y;
for (k = L-1 to 0)
{
z = z + (2^k) * (OS[y][0])
y = ((y+1) % x);
}
for (j = 0 to w)
{
if (URnum[j] = = z)
flag = 1;
}
if ( flag = = 0 )
{
URnum[w] = z;
z = 0;
w = w + 1;
if (w = = U)
goto 13;
}
else
{
y = (m + 1) % x;
flag = 0;
```

```
}
}
else
{
y = (y + 1) % x;
}
}
until (OS[i][1] = = 0)
```

10. y = 0, x = 0;
11. if (length [Tap Sequence] > 2) then remove first value of Tap Sequence and consider it as the seed. goto 8;
12. goto 6;
13. return URnum[U];

2) LFSR (LFSRT[R], Tap[Lt], Seed[R], Lt, R, w, i, j, k)

This algorithm is used to generate the random Output Symbols by using Seed[R] and Tap[Lt], Lt represents the length of Tap. LFSRT is a linear array of R length for storing the next Seed.

```
{
w = 0;
for (i = 0 to Lt)
{
j=Tap[i];
w = X-OR (w , LFSRT[(R-1) - j]);
}
OS[k][0]=LFSRT[R-1];
OS[k][1]=0;
for (i = R-2 to 0)
{
LFSRT[i+1]=LFSRT[i];
}
LFSRT[0] = w;
k = k + 1;
} until (Seed gets repeated);
```

IV. EXAMPLE

If Keyword = "*RAVINDRA@july1987*", Number system is ascii (N = *128*) and count of Unique Random number (U) is *24* then-

Using Formula (2)-

R = *7*, because N = 128 ($2^R >= 128 > 2^{R-1}$).

Using formula (3)-

L = 5, because U = 24 ($2^L > 24 >= 2^{L-1}$).

Table II : Extracting Seed And Tap From Keyword

Letters	Relevant Values (ascii)		mod_R (ascii, R=7)	
R	≈	82	% 7	5
A	≈	65	% 7	2
V	≈	86	% 7	2
I	≈	73	% 7	3
N	≈	78	% 7	1
D	≈	68	% 7	5
R	≈	82	% 7	5
A	≈	65	% 7	2
@	≈	64	% 7	1
j	≈	106	% 7	1
u	≈	117	% 7	5
l	≈	108	% 7	3
y	≈	121	% 7	2
1	≈	49	% 7	0
9	≈	57	% 7	1
8	≈	56	% 7	0
7	≈	55	% 7	6

keyword's relative value = {82, 65, 86, 73, 78, 68, 82, 65, 64, 106, 117, 108, 121, 49, 57, 56, 55}

Seed sequence = {82, 65, 86, 73, 78, 68, 64, 106, 117, 108, 121, 49, 57, 56, 55} (by removing duplicate values) mod_R of keyword's relative value = {5, 2, 2, 3, 1, 5, 5, 2, 1, 1, 5, 3, 2, 0, 1, 0, 6}

Tap sequence = {5, 2, 3, 1, 0, 6} (by removing duplicate values)

At first attempt-

Seed: 1010010

Tap: 5, 2, 3, 1, 0, 6

Using LFSR Algorithm: by using LFSR algorithm the output symbols OS[K][0] will be-

0 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, , 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1
Total output symbols = 127

Using Unique Random Number Generator Algorithm:

By using the output symbols generated by LFSR, Unique Random Number Generator will generate following sequence of Unique Random numbers-

Table III : Generated Unique Random Numbers

Binary Values	Decimal Values
0 1 0 0 1	9
0 1 0 1 0	10
1 1 0 1 1	27
1 0 0 1 0	18
0 0 0 1 0	2
0 0 1 0 0	4
1 0 0 1 1	19
1 1 0 1 0	26
1 0 0 0 0	16
0 1 0 1 1	11

1 1 0 0 1	25
1 0 1 1 1	23
1 1 1 1 0	30
0 0 1 0 1	5
1 0 0 0 1	17
0 0 1 1 0	6
1 0 1 0 0	20
0 1 1 1 1	15
0 0 0 0 1	1
1 1 0 0 0	24

0 1 1 0 0	12
0 0 0 1 1	3
0 1 1 1 0	14
1 0 1 0 1	21

Fig 2 : Illustrate the generation of unique random numbers for the above mentioned example.

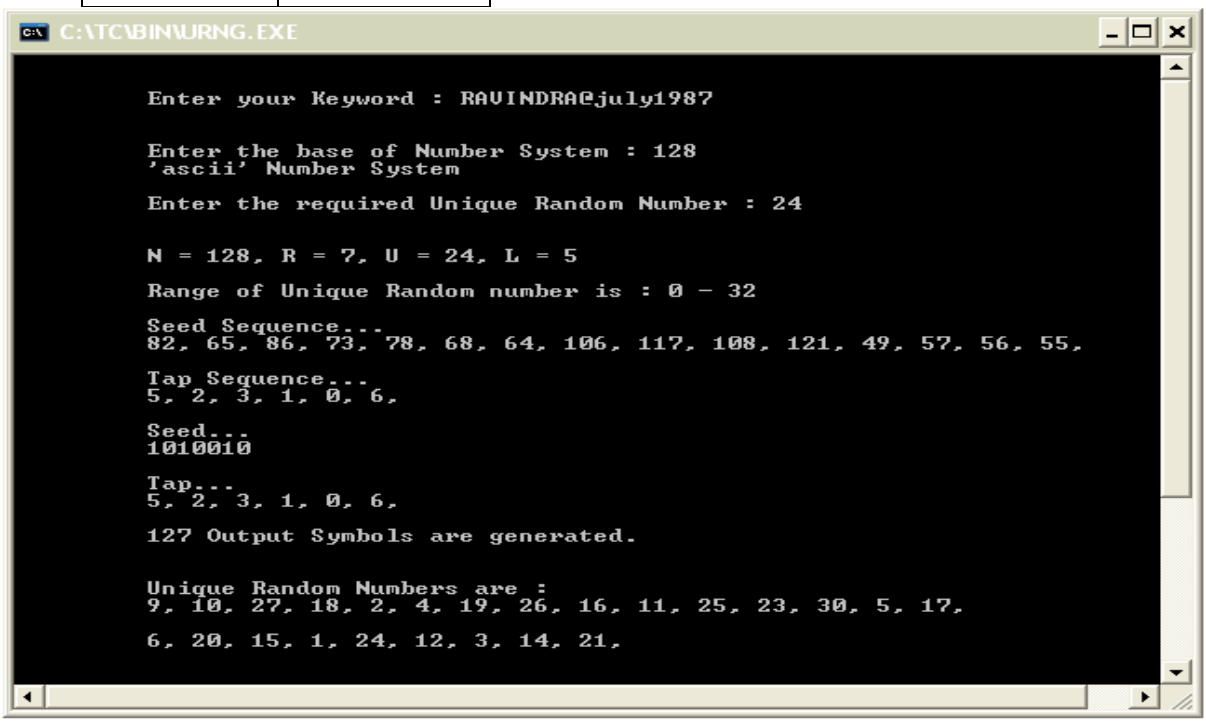

Fig. 2 : Unique Random Number Generator

V. CONCLUSION

Linear Feedback Shift Register based Unique Random Number Generator is an effective approach for generating unique random numbers on the basis of user's seed (keyword). By using C program, its performance was measured on a 3GHz Pentium®4 with 1GB of RAM running Windows XP professional Version 2002, Service pack 3. Results conclude that Linear Feedback Shift Register based Unique Random Number Generator can generate any count of Unique random numbers against the user's key. Its implementation is quite simple and performance of proposed approach is awesome.

REFERENCES

[1] Wayne Tomasi "Electronic Communications System Fundamentals through Advanced . 5th edition, Pearson Education, 2008.

[2] Rajski J, Tyszer J, "On the diagnostic properties of linear feedback shift registers", ISSN : 0278-0070, IEEE @ 06 August 2002

[3] Raina R, Marionos P, "Signature analysis with modified linear feedback shift registers (M-LFSRs)", Print ISBN: 0-8186-2150-8, IEEE @ 06 August 2002

[4] "Random number generation", Wikipedia [online], Available at : http://en.wikipedia.org/wiki/Random_number_generation.

[5] "What's this fuss about true randomness?", Random [online], Available at : http://www.random.org/

[6] "Random Number Generator", Random Number Generator [online], Available at : http://www.randomnumbergenerator.com/

[7] Simon Haykin , Communication Systems. , 4th Edition , Willey.

[8] Krishnaswamy S, Pillai H K, "On the Number of Linear Feedback Shift Registers With a Special Structure", ISSN : 0018-9448, IEEE @ 27 February 2012

[9] Murali P, Senthilkumar G, "Modified Version of Playfair Cipher Using Linear Feedback Shift Register", Print ISBN: 978-0-7695-3595-1, IEEE @ 19 June 2009

[10] "Linear Feedback Shift Registers", Available at : *http://homepage.mac.com/afj/lfsr.html.

[11] "Linear feedback shift register ", Wikipedia [online], Available at : http://en.wikipedia.org/ wiki/Linear_feedback_shift_ register.html.

[12] The Art of Electronics, 2ndEdition,Horowitzand Hill, 1989, pp. 665-667

[13] Dan Healy, "Understanding Linear Feedback Shift Registers – The Easy Way", Yikes [online], Available at : http://www.yikes.com/ ~ptolemy/lfsr_web/index.htm

[14] P. Alfke, "Efficient Shift Registers, LFSR, Counters, and Long Pseudo-Random Sequence Generators,"XAPP 052, July 7,1996 (Version 1.1)

[15] W.W. Peterson and E.J. Weldon, Jr. Error Correcting Codes, MIT press, Cambridge, MA 1972.

◈◈◈