International Journal of Sciences:
Basic and Applied Research
(IJSBAR)

ISSN 2307-4531
(Print & Online)

http://gssrr.org/index.php?journal=JournalOfBasicAndApplied

International Journal of
Sciences:
Basic and Applied
Research

ISSN 2307-4531
(Print & Online)

Published by:
GSSRR

Visit: www.gssrr.org

Double Feedback LFSR Parallel Output Generation

Robert Apikyan[a*], Hovhannes Gomtsyan[b], Vahe Bayadyan[c]

[a,b]*Institute of Information and Telecommunication Technologies and Electronics, NPUA, Yerevan, Armenia*

[c]*Tum School of Management, Technical University of Munich, Munich, Germany*

[a]*Email: apikyan41@gmail.comt*

[b]*Email: hovhannes.gomcyan@politechnic.am*

[c]*Email: bayadvahe@gmail.com*

Abstract

LFSR shift registers provide the mechanism that allows generating PRN (pseudo-random numbers) sequences. LFSR consists of sequentially connected memory cells where each cell can apply one of the two values 0 or 1. These cells are often called flipflops. In each step, the value from the last flipflop is passed to the next one. The first flip-flop in register applies LFSR's feedback value. The feedback value is the modulo addition of current values from specified flipflops in position. The output code of LFSR is a sequence of ones and zeros. Mostly these outputs or PRN sequences are used for signal modulation in CDMA. For example, in GPS, the broadcasting signal from satellites is modulated with unique PRN numbers for each satellite to allow receivers to identify data with broad-casting satellite numbers. The identification process is performed based on the correlation between locally generated PRN numbers and received signals from satellites. The pick of correlation value means that the signal is identified. Also, the LFSR’s are using in cryptography, and the encryption and decryption time depends on LFSR’s output generation time. The parallel generation method will short this time. The PRN code generation is a sequential process, making it parallel will short the overall signal identification and correlation time for CDMA usage and the encryption and decryption time in cryptography . This article as a unit of discussion will be two feedback based on LFSR’s parallel output generation, and the definition of generic formula that will allow us to define upcoming states of double feedback LFSR without sequential processing.

Keywords: LFSR; PRN sequences; parallel generation; Java.

1. Introduction

For generating n length output, single LFSR need's to run n cycles. Here, the time of generation always depends on the required output's n length. Shorting the time of the output generation allows making fast encryption and decryption of data. One of the ways to short the time of generation is using multiple LFSR's and the generation process could be done in a parallel manner. The main problem here is, how to set up each LFSR's initial state for a given position, where the state is LFSR's flipflop values for a given moment. Generally speaking, for LFSR's parallel output generation, we need to know it's state for a given moment. In the next section, we will discuss LFSR's upcoming state definition.

2. LFSR's upcoming state definition methodology

Consider a simple LFSR with 4 flipflops and two feedbacks Figure 1.

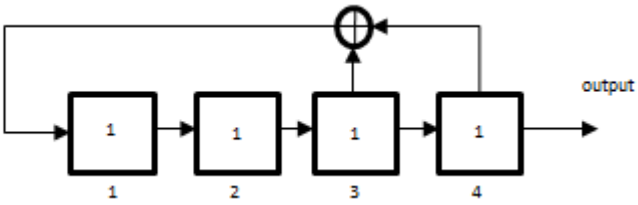

Figure 1: LFSR with double feedback.

As shown in Fig. 1 LFSR's feedbacks are equal to $f_1 = 3$ $f_2 = 4$, and initial state is set to $S_0 = \{1, 1, 1, 1\}$. On each step, the value from the previous flipflop will be applied to the next one and the value from 4th flipflop will be emitted as an output value. For calculating the feedback value it performs a modulo addition of values from 3rd and 4th flipflops and passes the result to 1st [1]. The modulo addition table is given in table 1.

Table 1: Table type styles (Table caption is indispensable).

f_1	f_2	Result
0	0	0
1	1	0
0	1	1
1	0	1

In general n length LFSR can produce a maximum 2n-1 length of the unique output, after 2^n-1 cycle it will repeat the output values, however, the output code length could be shorter then 2^n-1, it depends on chosen feedback flipflops positions [2]. The LFSR from figure 1 will produce the maximum length sequence, which means the output length of generated values will be 2^4-1 = 15. The output values for the first cycle are displayed in figure 2.

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

Figure 2: Maximum output values for 5 length LFSR with $f_1 = 3$ $f_2 = 4$

The generation process of LFSR output values is sequential, as the value of each flip-flop is depended from the previous flip-flop. In general to make some linear process parallel we need to define its state at a given moment. So In case of LFSR, it states defines the flipflops or the values of the registers. For LFSR from figure 1 the state values are shown in table 2.

Table 2: LFSR states n = 4, f_1 = 3, f_2 =4

State	f_1	f_2	f_3	f_4	State	f_1	f_2	f_3	f_4
S_1	1	1	1	1	S_9	1	1	0	0
S_2	0	1	1	1	S_{10}	0	1	1	0
S_3	0	0	1	1	S_{11}	1	0	1	1
S_4	0	0	0	1	S_{12}	0	1	0	1
S_5	1	0	0	0	S_{13}	1	0	1	0
S_6	0	1	0	0	S_{14}	1	1	0	1
S_7	0	0	1	0	S_{15}	1	1	1	0
S_8	1	0	0	1	S_{16}	1	1	1	1

As an example, we can see that for stating generation from 4th bit of output value in figure 2, LFSR's registers need to be set up with S_4 state from table 2. Here we can see that S_4 is modulo addition of S_1 and S_2 which could be written as the first row in equation 1. The LFSR's two modulo addition of LFSR states will give one of the upcoming states.

$$\begin{aligned} S_4 &= S_1 \oplus S_2 \\ S_5 &= S_2 \oplus S_3 \\ S_6 &= S_3 \oplus S_4 \\ S_7 &= S_4 \oplus S_5 \\ &\dots \end{aligned} \tag{1}$$

In general equation 1 could be rewritten as follows.

$$S_k = S_i \oplus S_j \tag{2}$$

Where i and j are required state indexes that will be modulo added to each other to define k state. For LFSR from figure 1 we can see the following pattern that j = i + 1 and k = j +2 and Equation 2 could be rewritten as follows.

$$S_{j+2} = S_i \oplus S_{i+1} \tag{3}$$

Now we can define S_{i+3} for each given i, but the equation 3 will work for only LFSRs with length 4 where f_1 = 3 and f_2 = 4, for other LFSRs with different length and f_1, f_2 values equation 3 will not work. Let's change the first feedback value, now f_1 = 2. The length of generated output sequences will be equal to 6 as shown in figure 3, which means that the LFSR is not maximum length.

1 1 1 1 0 0…

Figure 3: Output values for 5 length LFSR with $f_1 = 2$ $f_2 = 4$

The states for 5 length LFSR with $f_1 = 2$ $f_2 = 4$ are shown in table 3.

Table 3: LFSR states n = 4, $f_1 = 2$, $f_2 = 4$

State	f_1	f_2	f_3	f_4
S_1	1	1	1	1
S_2	0	1	1	1
S_3	0	0	1	1
S_4	1	0	0	1
S_5	1	1	0	0
S_6	1	1	1	0
S_7	1	1	1	1

From table 3 we can define the following pattern for each state.

$$\begin{aligned} S_5 &= S_1 \oplus S_3 \\ S_6 &= S_2 \oplus S_4 \\ S_7 &= S_3 \oplus S_5 \\ S_8 &= S_4 \oplus S_6 \end{aligned} \tag{4}$$

Let's define the dependency between i, j and k from equation (4) value. We can see that for each i j = i + 2 and k = j + 2, from this the general state equation will be as fallows.

$$S_{j+2} = S_i \oplus S_{i+2} \tag{5}$$

By comparing equation 3 and equation 5 we can see that the only difference is in j index value. For first one j = i+1 and for second one j = i+2. It's easy to see that this difference depends from f_1 and f_2 feedback positions, more the difference value could be defined as $j = i + f_2 - f_1$ and overall the equation 5 could be rewritten as follows.

$$S_{i + f2} = S_i \oplus S_{i + f2 - f1} \tag{6}$$

Now let's change the feedback positions $f_1 = 1$ and $f_2 = 2$. First six states values are displayed in table 4.

Table 4: LFSR states n = 4, $f_1 = 1$, $f_2 = 2$

State	f_1	f_2	f_3	f_4
S_1	1	1	1	1
S_2	0	1	1	1
S_3	1	0	1	1
S_4	1	1	0	1
S_5	0	1	1	0
S_6	1	0	1	1

As we can see from table 4 states $S_3 = S_6$, which means that the length of the output code sequence will be 4. The important part here is that S_1 and S_2 states will never be repeated whole generation, that's why they can't be used in equation 6. In equation 6 the argument states must be repeatable, which means that S_i and $S_{i+f2-f1}$ might be periodically repeated by the register. And we can see that the count of non-repeatable states depends on the second feedback position, in this case, $f_2 = 2$. Here we need some statement that will skip non-repeatable states, and that statement is k >= n or by representing the same with i we can say fallowing.

$$i >= n - f_2 \quad (7)$$

Now we can define a formula with a statement that will define double feedback shift register upcoming state for a given moment of time.

For a given n length double feedback LFSR, where f1 [0,n], f2 [0,n], f1 < f2 and picked $i >= n - f_2$ then $S_{i+f2} = S_i \oplus S_{i+f2-f1}$.

Figure 4: Generic formula for defining upcoming state of n length, double feedback LFSR.

3. Parallel generation methodology

Let's use the equation 6 for the parallel generation process. As we can see the equation requires two initially defined states S_i and $S_{i+f2-f1}$. Considering this, the parallel generation algorithm will wait for these two initials stats initialization and then define the upcoming state for starting a parallel generation on the new thread. Besides, the algorithm will check for index validity with equation 7. The general block diagram of the parallel generation algorithm is shown in figure 5. The parallel generation program starts from i, j, k indices definition. First it search for valid i index that will satisfy $i >= n - f_2$ statement. Whenever the statement is satisfied, the program starts j and k indices definition based on i. Next, it starts the loop that will work from i to k. As the j is always between i and k and it's required for S_k state definition, j will be the point where the program will start the new parallel process. The program will make a check on each iteration for index = j. When the statement returns true it starts the new parallel process and passes already defined k value to it. In the new process, the program recursively starts the definition of new j and k indexes based on i, where i is equal to k that is passed from the last process. After starting the new parallel process, the current process will continue generating code sequences until the end of its iteration. Each process will check the generated code length if it's value has achieved the required length, which means the end of the generation, and the program will be terminated.

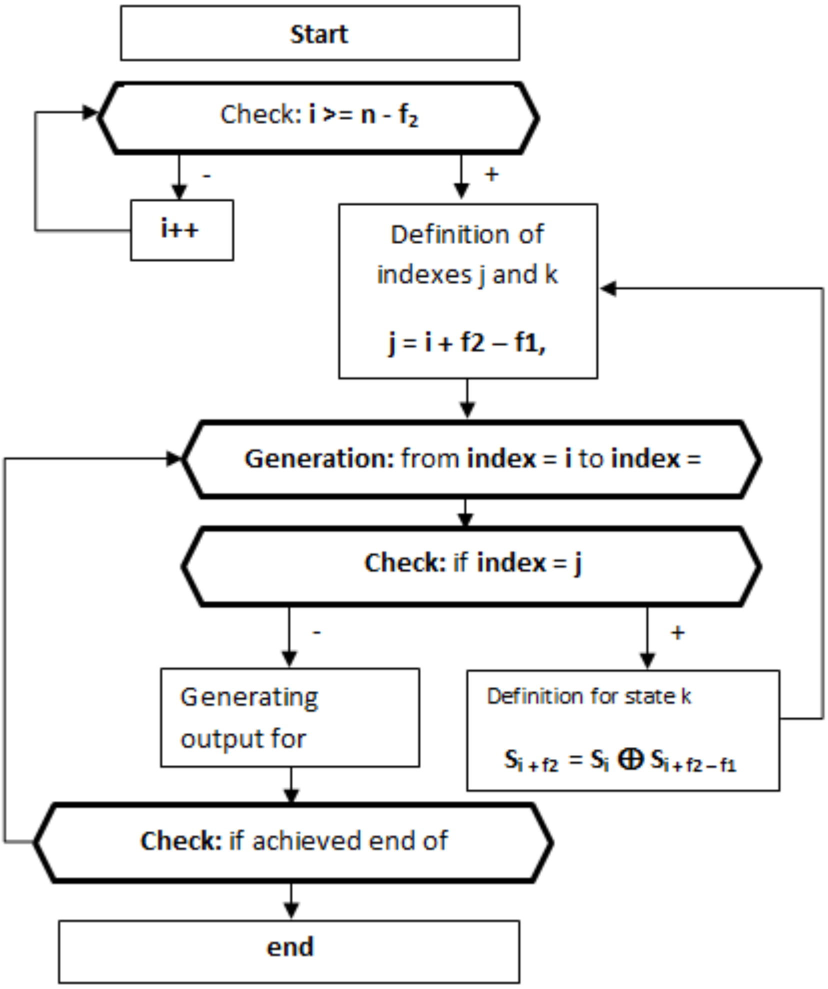

Figure 5: LFSR parallel generation algorithm

4. Parallel generation program

The program was written in Java its source code is located in GitHub repository [3]. The program is the implementation of parallel generation algorithm that is described in the previous section. Java's Thread pool executors have been used for parallel generation. For starting the program find the ParallelTwoFeedbackLFRS.java [4] file in the project and run the main() method. After program execution, the output code sequence will be printed in the console. By default, the program has a configuration to generate 1024 code sequences from LFSR that has 33 registers, where feedback values are f1 = 32 and f2 = 33. These configurations could be changed by modifying lfsrRunLength, lfsrLength, f1 and f2 parameters in the main() method.

5. Conclusion

As a result of the discussions above now we have a formula that allows generating upcoming states of double feedback LFSRs. Based on this formula the parallel generation algorithm is defined and a program, that implements that algorithm.

References

Thesis:

[1] James Bao-Yen Tsui, a Wiley Interscience publication "Fundaments of Global Positioning System Receivers: A Software Approach", 93 –99. http://twanclik.free.fr/electricity/electronic/pdfdone7/Fundamentals%20of%20Global%20Positioning%20System%20Receivers.pdf

Book:

[2] John F. Brendle Jr. (2000) Pseudorandom Code Generation for Communication and Navigation System Application, 17-23. https://apps.dtic.mil/dtic/tr/fulltext/u2/a336311.pdf

Internet:

[3] Program reference in GitHub - https://github.com/RobertApikyan/GpsGenerator/tree/parallel_lfsr_generation

[4] ParallelTwoFeedbackLFRS.java class reference in GitHub - https://github.com/RobertApikyan/GpsGenerator/blob/parallel_lfsr_generation/src/src/main/ParallelTwoFeedbackLFSR.java