**International Journal of Electrical and Computer Engineering (IJECE)**
Vol. 7, No. 4, August 2017, pp. 1882~1891
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i4.pp1882-1891 

# FPGA-based Design System for a Two-Segment Fibonacci LFSR Random Number Generator

**Zulfikar, Yuwaldi Away, Rafiqa Shahnaz Noor**
Department of Electrical and Computer Engineering, Syiah Kuala University, Banda Aceh, Indonesia

**Article Info**

***Article history:***

Received Jan 19, 2017
Revised Apr 28, 201
Accepted May 12, 2017

***Keyword:***

Fibonacci LFSR
FPGA
Galois LFSR
Matlab
Random number

**ABSTRACT**

For a long time, random numbers have been used in many fields of application. Much work has been conducted to generate truly random numbers and is still in progress. A popular method for generating random numbers is a linear-feedback shift register (LFSR). Even though a lot of work has been done using this method to search for truly random numbers, it is an area that continues to attract interest. Therefore, this paper proposes a circuit for generating random numbers. The proposed circuit is designed to produce different sequences of numbers. Two segments of Fibonacci LFSR are used to form a generator that can produce more varied random numbers. The proposed design consists of blocks: segment 1, segment 2, and a clock controller. The system produces random numbers based on an external clock. The clock signal for the first segment is that of the external clock, whereas that for the second segment is modified by the clock controller. The second stage (segment 2) is executed only after every $2^{n1}-1$ clock cycles. The proposed design can generate different sequences of random numbers compare to those of the conventional methods. The period of the proposed system is less than that of the original Fibonacci LFSR. However, the period is almost equal to the original one when the system is realized in 32-bit or 64-bit form. Finally, the proposed design is implemented on a field-programmable gate array (FPGA). It occupies more area and runs at a lower frequency compared with the original Fibonacci LFSR. However, the proposed design is more efficient than the segmented leap-ahead method concerning space occupancy.



***Corresponding Author:***

First Author,
Departement of Electrical and Computer Engineering,
Syiah Kuala University,
Jl Syech Abdul Rauf No. 7, Darussalam, Banda Aceh, Indonesia, 23111.
Email: zulfikarsafrina@unsyiah.ac.id

## 1. INTRODUCTION

For a long time, random numbers have been used in many fields of application, such as gambling, games, cryptography, computer simulations, statistical sampling, and completely randomized design. In real life, dice, playing cards, or tossed coins are often used for gambling. A die has six sides, each of which is marked with a different number of dots. When rolled, which face comes up on top is unpredictable. Hence, a sequence of die rolls can be simulated using random numbers, which is the basis of many real games that are played on computers.

Random numbers are usually generated as a sequence of (ideally) unpredictable numbers within a certain range and based upon a given initial seed value [1-3]. This initial value plays a crucial role in determining the resulting (possibly very long) sequence of numbers and whether those numbers are genuinely random. In the field of cryptography, the seed is often used as a password; whenever the seed is recognized, the encrypted data will be revealed. Therefore, the seed should be such that an unauthorized user cannot decrypt the data [3-6].

A random-number generator with a predefined seed produces what are known as pseudorandom numbers, usually by computer or machine. In that case, the seed is already determined and stored somewhere in the program. However, truly random numbers are generated based on an unknown or unspecified seed that is often set based on momentary conditions such as a clock time, a temperature, or the value of some other physical parameter.

Much work has gone into finding truly random numbers. Purushottam designed 8-bit and 16-bit linear-feedback shift register (LFSR) generators that are based on polynomial feedback. Whenever the feedback changes, so do the resulting sequence of random numbers [7], the length of which is limited by the length of the LFSR. This means that longer sequences of random numbers require more circuitry, which in turn runs more slowly and causes more delays [8].

In previous work, we proposed a very simple method of generating non-uniform random numbers using MATLAB [9]. Recently, a linear congruential generator (LCG) was successfully implemented in a field-programmable gate array (FPGA) [10-12]. Je and Seong generated barely correlated uniform random numbers by using a two-segment Galois leap-ahead LFSR. The period of the resulting random number generator was approximately 2.5 times that of a single-segment Galois leap-ahead LFSR [13].

The present paper presents a different method for generating random numbers, one that is based on a Fibonacci LFSR. The generator is formed from two segments of Fibonacci LFSR, resulting in more varied random numbers over a longer period. In section 2, we present some fundamental aspects of random numbers and their connection with FPGAs. In section 3, we give a short but precise method for generating random numbers. In section 4, we discuss other random-number methods and compare them with our proposed method. We also show how to implement the proposed generator in an FPGA. Finally, in Section 5, we draw certain conclusions from our results.

## 2. BACKGROUND THEORY

### 2.1. Random Numbers

Random numbers are numbers that appear random or unpredictable over a particular period, usually resulting from a random-number generator. There are two types of the generator based on how the random numbers are generated: a pseudorandom number generator (PRNG), or a truly random number generator (TRNG). Whereas a PRNG uses a known predefined seed, a TRNG uses a seed from some momentary physical situation, such as a clock time or a temperature.

Random numbers are used in applications ranging from electronic games to gambling to cryptography. Highly secure systems are often based on random numbers, which therefore have to be as random as possible. Truly random numbers can be achieved by choosing the right seed. Two parameters that are often used to determine the quality of random numbers are as follows [1], [3]:

a. Sequence length: The more random the numbers, the longer the sequence before they repeat.
b. Unpredictability: Numbers may be repeated in a random-number sequence whose length exceeds the sequence length. However, a good generator will produce different number sequences in such cases [14].

Most generators produce random numbers that are distributed uniformly. However, some applications require non-uniformly distributed numbers. For this, there are many generators that produce random numbers based on exponential, normal, or other distributions. However, the numbers so distributed tend to be ones that have been modified from a uniformly distributed sequence. More numbers are required in such cases, but some of these are ignored to fit with the specified non-uniform distribution [9].

### 2.2. Uniform Random Numbers

A uniform random number distribution is one type of pseudorandom number distribution, in addition to others such as Gaussian and exponential. A uniform random number generator generates a set of random numbers that are distributed uniformly in the sense that a sufficiently long sequence of numbers so generated would correspond to a uniform distribution. In other words, each number has the same probability of occurring. Uniform random numbers can be generated as formulated in Equation (1) [1-3]:

$$X_{n+1} = a + b\,X_n \pmod{m} \quad (1)$$

where

$n$ = *0,1,2….*
$a$ = shifting constant ($a < m$)
$b$ = multiplier constant ($b < m$)
$m$ = modulus ($> 0$)
$X_0$ = initial value (integer $\geq 0$, $X_0 < m$)

### 2.3. Linear Feedback Shift Register

A linear feedback shift register (LFSR) is a type of shift register that is built from D flip-flops and XOR gates. The term "linear" refers to the input linearity of its previous condition or state. A seed is required for the initial value at the beginning of the sequence of random numbers. Each flip-flop stores a state that is either high or low (in value). Whenever the clock goes high, every stored value is shifted to another flip-flop. To produce a long sequence of truly random numbers, feedback from the outputs of individual flip-flops is required [15].

Usually, an LFSR is represented by a polynomial of modulus 2, and the feedbacks are tapped according to the polynomial formula. Table 1 gives the maximum period of an LFSR based on its number of bits, which depends on the polynomial feedback.

Table 1. Correlation between LFSR maximum period and feedback/number of bits

| Number of bits | Period | Feedback |
|---|---|---|
| 2 | 3 | $X^2+X+1$ |
| 3 | 7 | $X^3+X^2+1$ |
| 4 | 15 | $X^4+X^3+1$ |
| 5 | 31 | $X^5+X^3+1$ |
| 6 | 63 | $X^6+X^5+1$ |
| 7 | 127 | $X^7+X^6+1$ |
| 8 | 255 | $X^8+X^6+X^5+X^4+1$ |
| 18 | 262,143 | $X^{18}+X^{11}+1$ |
| 19 | 524,287 | $X^{19}+X^{18}+X^{17}+X^{14}+1$ |
| 20–168 | [16] | [16] |
| 2–786, 1024, 2048, 4096 | [17] | [17] |

An LFSR may be implemented using either a Fibonacci or a Galois model, both of which reproduce the same amount (period) of random numbers. Figure 1 shows an example of a Fibonacci LFSR, in which the taps are bits that might influence the input of the LFSR state. The maximum length of an LFSR sequence is $2^n-1$ possible states, except those in which all the bits are zero.

In a Galois structure, the values (states) are changed when shifted from one position to the next. However, the taps are subjected to an XOR operation with the output bit before they are saved in the next position. The next input bit is the new output bit. Therefore, when the output bit is zero, all the states in the register are shifted to the next positions, and the input state becomes zero. Figure 2 shows the configuration of a Galois LFSR structure.

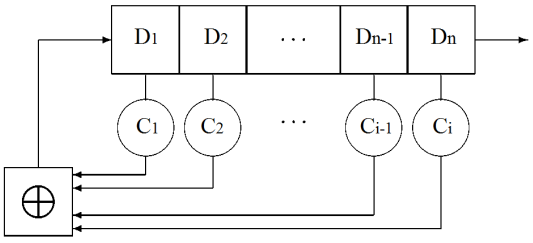


Figure 1. Fibonacci LFSR

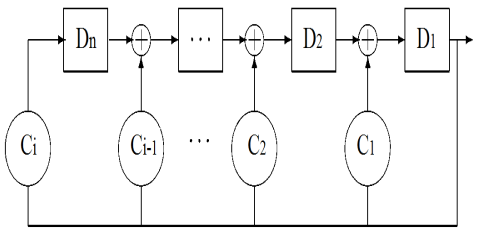


Figure 2. Galois LFSR

## 3. TWO-SEGMENT FIBONACCI LFSR

Figure 3 shows a block diagram of the proposed two-segment Fibonacci LFSR. The design consists of a clock controller and segments 1 and 2, the latter two of which are independent of each other. The system produces a random number every time an external clock goes high.

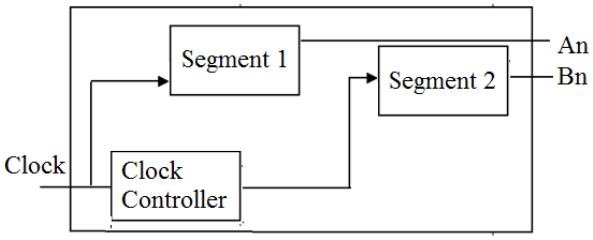


Figure 3. Block diagram of proposed two-segment Fibonacci LFSR

The two segments may consist of the same number or different numbers of D flip-flops. For instance, a 6-bit generator could be constructed from the following of flip-flops in the first segment and second segment as 2 and 4, 3 and 3, or 4 and 2, respectively.

The clock controller is used to supply a different clock signal to each segment. The controller circuit is designed based on the numbers of flip-flops in segments 1 and 2. For example, if the segment 1:2 flip-flop combination is 3:3, the clock frequency supplied to segment one will be seven times faster than that to segment 2. In general, the second stage (segment 2) will be executed (clock goes high) after the first stage produces a period of the random. However, there is no zero in the period. This design will be discussed in section 4.

The initial seed values of the proposed design are based on the numbers of flip-flops in both stages. If these numbers are the same, the same seed is used for both stages. Otherwise, different seeds are used to match the different numbers of flip-flops.

### 3.1. Design of 6-bit two-segment LFSR

As mention above, the circuit for a 6-bit two-segment LFSR can be built with different numbers of flip-flops in its segments. However, let us begin by considering the case of equal numbers of flip-flops in both segments, namely 3:3. Figure 4 shows the circuit design for one segment. According to the polynomial formula in Table 1 for generating 3-bit random numbers, three flip-flops and a XOR gate are required. The feedback of input flip-flop $D_1$ is XORed between the outputs of the second ($D_2$) and third ($D_3$) flip-flops. Implementing a 3:3 6-bit two-segment generator requires two circuits of the type shown in Figure 4, but with different clock inputs. The second segment responds only when the first stage has generated one period of random numbers.

Let us consider another model by taking the combination of 2:4. Two flip-flops are used in the first segment, whereas four are required in the second. Figure 5 shows a circuit realization of the design. Clock 1 is initially from the external clock without any modification. However, Clock 2 is an output of the clock controller circuit. Because the first stage contains only two flip-flops, the clock controller ensures that the clock signal supplied to the second stage is three times slower than the original one.

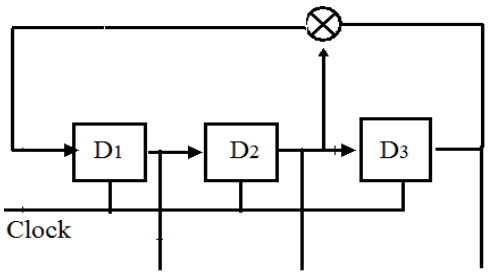


Figure 4. Design of a 3:3 3-bit Fibonacci LFSR: one segment

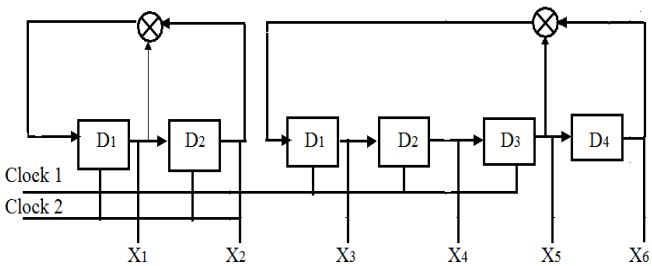


Figure 5. Design of a 2:4 6-bit two-segment Fibonacci LFSR: both segments

### 3.2. Design of 8-bit two-segment LFSR

More combinations may be used in the case of a bigger circuit. For example, an 8-bit generator can be formed from segment 1:2 flip-flop combinations of 2:6, 3:5, 4:4, 5:3, or 2:6. Let us consider the 3:5 case, for which two different Fibonacci circuits are required (a 3-bit segment 1 and a 5-bit segment 2). The circuit of segment 1 is similar to the one shown in Figure 4. The design of the second stage is shown in Figure 6,

based on 5-bit polynomial feedback. The feedback that drives the input of the first flip-flop is the XOR of the outputs of $D_3$ and $D_5$.

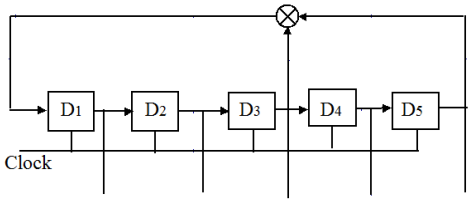


Figure 6. Design of a 5-bit Fibonacci LFSR segment

### 3.3. Clock Controller

In the proposed model, the second stage does not change until the first stage has generated a period of random numbers. The period of the first segment ($P_1$) is based on its number of flip-flops ($n_1$) as shown in Equation (2).

$$P_1 = 2^{n_1} - 1 \quad (2)$$

Therefore, to control the second segment, we have to apply a clock signal to the second segment every $P_1$ times that the first clock is supplied from outside the system. Let us consider the case of a 2:4 6-bit two-segment design as shown in Figure 7. The initial seed value is set to be decimal 17. The initial values of segments 1 and 2 are "01" (decimal 1) and "0001" (decimal 1), respectively. When the first clock is applied to this generator, the first segment is executed but the second part remains unchanged. This scheme continues until the first layer has generated one period ($2^2-1 = 3$) of random numbers. When the third clock cycle is applied, the second segment responds by changing its value to "0010" (decimal 2). The second segment responds again after another three clock cycles.

In other words, the first layer responds to each clock cycle, whereas the second layer responds to every $P_1$ clock cycles. Based on this, the circuit of the clock controller can be realized by a synchronous counter of modulus $P_1$. Therefore, the design requires a total of ($n + n_1$) D flip-flops in general, where $n$ is the combined number of flip-flops in both segments and $n_1$ is the number in the clock controller.

| Clock | Segment 1 A2 | Segment 1 A1 | Segment 2 B4 | Segment 2 B3 | Segment 2 B2 | Segment 2 B1 | Random Number | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 1 | 17 | Seed |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 49 | No change |
| 2 | 1 | 0 | 0 | 0 | 0 | 1 | 33 | |
| 3 | 0 | 1 | 0 | 0 | 1 | 0 | 18 | |
| 4 | 1 | 1 | 0 | 0 | 1 | 0 | 50 | No change |
| 5 | 1 | 0 | 0 | 0 | 1 | 0 | 34 | |
| 6 | 0 | 1 | 0 | 1 | 0 | 0 | 20 | |
| 7 | 1 | 1 | 0 | 1 | 0 | 0 | 52 | |

one period

Figure 7. Illustration of random-number generation process

## 4. RESULTS AND DISCUSSION

### 4.1. Sequence of Random Numbers

The period of the proposed two-segment Fibonacci LFSR generator is shorter than that of the original one. However, the former produces different sequences of random numbers. Table 2 and Table 3 list some of the random numbers generated by the original one-segment (Table 2) and the proposed 2:4 two-segment (Table 3) 6-bit Fibonacci LFSR. Table 4 lists some of the random numbers generated by the proposed 3:3 two-segment 6-bit Fibonacci LFSR, which provides more numbers than does the 2:4 generator.

Table 2. List of random numbers generated by original 6-bit Fibonacci LFSR

| No. | Random number | No. | Random number | No. | Random number | No. | Random number |
|---|---|---|---|---|---|---|---|
| 1 | 32 | 18 | 19 | 35 | 18 | 52 | 53 |
| 2 | 1 | 19 | 39 | 36 | 37 | 53 | 42 |
| 3 | 2 | 20 | 15 | 37 | 11 | 54 | 21 |
| 4 | 4 | 21 | 30 | 38 | 22 | 55 | 43 |
| 5 | 8 | 22 | 61 | 39 | 45 | 56 | 23 |
| 6 | 16 | 23 | 58 | 40 | 27 | 57 | 47 |
| 7 | 33 | 24 | 52 | 41 | 55 | 58 | 31 |
| 8 | 3 | 25 | 40 | 42 | 46 | 59 | 63 |
| 9 | 6 | 26 | 17 | 43 | 29 | 60 | 62 |
| 10 | 12 | 27 | 35 | 44 | 59 | 61 | 60 |
| 11 | 24 | 28 | 7 | 45 | 54 | 62 | 56 |
| 12 | 49 | 29 | 14 | 46 | 44 | 63 | 48 |
| 13 | 34 | 30 | 28 | 47 | 25 | 1 | 32 |
| 14 | 5 | 31 | 57 | 48 | 51 | 2 | 1 |
| 15 | 10 | 32 | 50 | 49 | 38 | 3 | 2 |
| 16 | 20 | 33 | 36 | 50 | 13 | 4 | 4 |
| 17 | 41 | 34 | 9 | 51 | 26 | 5 | 8 |

Table 3. List of random numbers generated by the proposed 2:4 two-segment 6-bit Fibonacci LFSR

| No. | Random number | No. | Random number | No. | Random number | No. | Random number |
|---|---|---|---|---|---|---|---|
| 1 | 17 | 14 | 51 | 27 | 37 | 40 | 28 |
| 2 | 49 | 15 | 35 | 28 | 27 | 41 | 60 |
| 3 | 33 | 16 | 22 | 29 | 59 | 42 | 44 |
| 4 | 18 | 17 | 54 | 30 | 43 | 43 | 24 |
| 5 | 50 | 18 | 38 | 31 | 23 | 44 | 56 |
| 6 | 34 | 19 | 29 | 32 | 55 | 45 | 40 |
| 7 | 20 | 20 | 61 | 33 | 39 | 1 | 17 |
| 8 | 52 | 21 | 45 | 34 | 31 | 2 | 49 |
| 9 | 36 | 22 | 26 | 35 | 63 | 3 | 33 |
| 10 | 25 | 23 | 58 | 36 | 47 | 4 | 18 |
| 11 | 57 | 24 | 42 | 37 | 30 | 5 | 50 |
| 12 | 41 | 25 | 21 | 38 | 62 | 6 | 34 |
| 13 | 19 | 26 | 53 | 39 | 46 | 7 | 20 |

As seen from Table 2, the original 6-bit Fibonacci LFSR produces a full period of random numbers. The numbers colored red have appeared before at the beginning of the sequence. The original system produces $2^6-1 = 63$ numbers. Numbers 32, 1, 2, 4, and 8 are a repetition of the ones that first appeared. Meanwhile, the proposed 2:4 Fibonacci generator provides slightly fewer numbers: $(2^2-1)(2^4-1) = 45$ (see Table 3). Similarly, the proposed 3:3 generator produces fewer random numbers than does the original, albeit a few more than does the 2:4 generator.

Table 4. List of random numbers generated by the proposed 3:3 two-segment 6-bit Fibonacci LFSR

| No. | Random number | No. | Random number | No. | Random number | No. | Random number |
|---|---|---|---|---|---|---|---|
| 1 | 9 | 14 | 34 | 27 | 51 | 40 | 62 |
| 2 | 17 | 15 | 13 | 28 | 35 | 41 | 54 |
| 3 | 41 | 16 | 21 | 29 | 15 | 42 | 38 |
| 4 | 25 | 17 | 45 | 30 | 23 | 43 | 12 |
| 5 | 57 | 18 | 29 | 31 | 47 | 44 | 20 |
| 6 | 49 | 19 | 61 | 32 | 31 | 45 | 44 |
| 7 | 33 | 20 | 53 | 33 | 63 | 46 | 28 |
| 8 | 10 | 21 | 37 | 34 | 55 | 47 | 60 |
| 9 | 18 | 22 | 11 | 35 | 39 | 48 | 52 |
| 10 | 42 | 23 | 19 | 36 | 14 | 49 | 36 |
| 11 | 26 | 24 | 43 | 37 | 22 | 1 | 9 |
| 12 | 58 | 25 | 27 | 38 | 46 | 2 | 17 |
| 13 | 50 | 26 | 59 | 39 | 30 | 3 | 41 |

In the original Fibonacci LFSR, zero is not produced because at least one flip-flop must always be in the high (logical 1) state. However, more numbers are excluded in the proposed structure. Now, at least one flip-flop must always be in the high state in both segments. In other words, at least two flip-flops must always be in the high state in the proposed generator.

Table 5 lists the periods of various combinations of the proposed two-segment Fibonacci LFSR compared to the original one. The calculations of the period $P$ of the proposed generator are based on Equations (3) and (4):

$$P = P_1 P_2 \quad (3)$$

substitute $P_1$ and $P_2$ from Equation (2) so that

$$P = (2^{n1}-1)(2^{n2}-1) \quad (4)$$

where

$n_1$: number of flip-flops in segment 1
$n_2$: number of flip-flops in segment 2

Based on the data in Table 5, the periods of the proposed two-segment Fibonacci LFSR generator are less than those of the original one. The proposed generators produce maximum periods when the numbers of flip-flops in both segments are equal ($n_1 = n_2$).

Table 5. List of periods of original one-segment and proposed two-segment Fibonacci LFSR

| | 6 bit | 8 bit | 18 bit | 32 bit | 64 bit |
|---|---|---|---|---|---|
| Original | 63 | 255 | 262,143 | $4.2949 \times 10^9$ | $1.844674407 \times 10^{19}$ |
| 2:4 or 4:2 | 45 | - | - | - | - |
| 3:3 | 49 | - | - | - | - |
| 2:6 or 6:2 | - | 189 | - | - | - |
| 3:5 or 5:3 | - | 217 | - | - | - |
| 4:4 | - | 225 | - | - | - |
| 9:9 | - | - | 261,121 | - | - |
| 16:16 | - | - | - | $4.2948 \times 10^9$ | - |
| 32:32 | - | - | - | - | $1.844674406 \times 10^{19}$ |

The proposed 6-bit generator produces 45 and 49 random numbers for the combinations of 2:4 (or 4:2) and 3:3, respectively; the period of the 3:3 generator is 77.78% that of the original one. For the 8-bit generator, the proposed 4:4 one provides a period that is 88.23% that of the original LFSR generator. For the 18-bit design, the proposed two-segment LFSR generator is capable of generating a period that is 99.61% that of the original LFSR generator. Therefore, for higher numbers of bits (32 and 64 bits are commonly used in applications), the difference in period between the designed and original generators is vanishingly small, as shown in Figure 8.

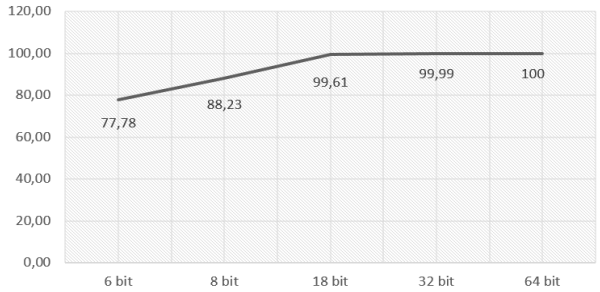


Figure 8. Maximum period of the proposed two-segment LFSR generator as a percentage of the period of the original generator plotted against the number of bits

### 4.2. FPGA Implementation

We simulated the proposed two-segment Fibonacci LFSR generator in FPGA software to compare it with other designs. Figure 9 shows the simulated behavior of a 4:4 8-bit two-segment Fibonacci LFSR. In the figure, lfsr_out[3:0] represents the output numbers of the first segment, and lfsr_out1[3:0] represents the numbers produced by the flip-flop in the second part. The blue-highlighted lfsr_out12[7:0] is the random-number output of the proposed generator, which is a combination of the output of both segments.

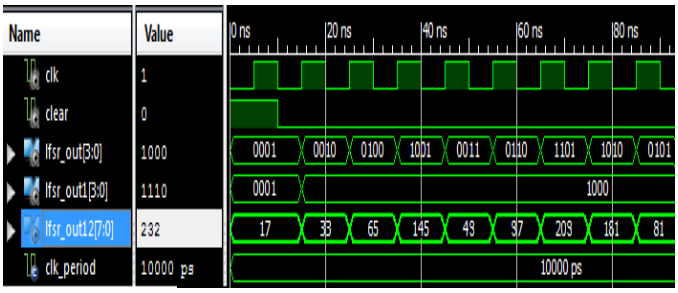


Figure 9. Simulated behavior of the proposed 4:4 8-bit two-segment Fibonacci LFSR

In other to verify the proposed design manually, we implemented it on a BASYS 2 FPGA board that was equipped with a relatively inexpensive Spartan 3 integrated circuit. Onboard LEDs were used to verify manually random numbers produced by the proposed design, as shown in Figure 10.

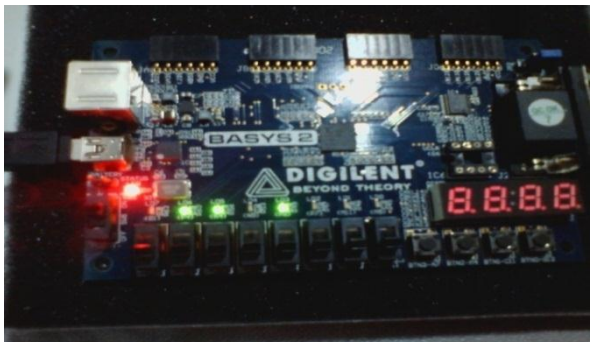


Figure 10. Hardware realization of 4:4 8-bit two-segment Fibonacci LFSR and a BASYS 2 FPGA board

Table 6 and Table 7 list the occupied areas and maximum frequencies of the original versus proposed Fibonacci LFSRs for 6 and 8 bits, respectively. The proposed generators require roughly twice the space that the original ones require. Similarly, the proposed Fibonacci LFSRs are slower than the original ones. However, this is a small cost compared with the resulting different sequences as indicated in Table 3 and Table 4. Of course, the proposed design does require additional flip-flops to control the clock signal supplied to the second segment; this circuit was discussed in section 3.3.

Table 6. Comparison of occupied slices and frequencies of different Fibonacci LFSR designs: 6-bit

| Fibonacci LFSR type | Slices | Frequency (MHz) |
|---|---|---|
| Original | 3 | 519.224 |
| 3:3 two-segment | 7 | 336.055 |
| 2:4 two-segment | 6 | 436.862 |

Table 7. Comparison of occupied slices and frequencies of different Fibonacci LFSR designs: 8-bit

| Fibonacci LFSR type | Slices | Frequency (MHz) |
|---|---|---|
| Original | 4 | 519.224 |
| 4:4 two-segment | 8 | 330.764 |
| 3:5 two-segment | 8 | 336.055 |
| 2:6 two-segment | 7 | 334.317 |

To facilitate comparison, the proposed design was implemented using Xilinx ISE on to a Virtex-II Pro chip numbered XCV2VP30. Table 8 gives another comparison of the proposed two-segment Fibonacci LFSR with the two-segment leap-ahead Galois LFSR proposed in 2013 by Lee et al. [13]. It can be seen that the proposed generator can generate the same amount of random numbers as can the generator introduced by Lee et al. regarding the area, the proposed design is better than the previous model, even though the developed model is slower than the segmented leap-ahead Galois LFSR.

Table 8. Comparison of maximum period, occupied slices, and frequency for 18-bit LFSRs

| | Max. period | Number of segments | Slices | Freq. (MHz) |
|---|---|---|---|---|
| Segmented leap-ahead Galois LFSR [13] | 261,121 | 2 | 35 | 535 |
| Proposed 4:4 two-segment Fibonacci LFSR | 261,121 | 2 | 25 | 350 |

## 5. CONCLUSION

A two-segment Fibonacci LFSR has been designed and implemented successfully. The design requires ($n + n_1$) D flip-flops. The proposed design approaches the sequence length of the original design with increasing bit size. However, the occupied area is greater and the speed is lower compared with the original one. When the model is implemented on an FPGA, the proposed two-segment Fibonacci LFSR shows the same period as that of the one developed by Lee et al. [13] but it has occupies less area. The proposed design is unable to generate certain numbers, such as "00010000" or "00000010." This is because there should be at least two flip-flops in the high state at any given iteration.

## REFERENCES


[1] A. L. Garcia, "Probability and Random Process for Electrical Engineering," 2nd ed., Addison-Wesley Publishing Company, 1994.

[2] M. Evans, N. Hasting, B. Peacock, "Statistical Distributions," 3rd ed., Wiley series in probability and statistic, 2000.

[3] R. Halprin, M. Naor, "Games for Extracting Randomness," Weizmann Institute of Science. 2009.

[4] Jonathan Katz, and Yehuda Lindell, "Introduction to Modern Cryptography," 2nd Edition, *CRC Press*, 2007.

[5] S. Bruce, "Applied Cryptography: Protocols, Algorithms, and Source Code in C," Wiley-India, 2007

[6] B. James, Body of Secrets, "Anatomy of the Ultra-Secret National Security Agency," Knopf Doubleday Publishing Group, Dec 18, 2007

[7] C. Y. Purushottam, R.V. Kshirsagar, *"Design of 8 and 16 Bit LFSR with Maximum Length Feedback Polynomial Using Verilog HDL,"* IRF International Conference, Pune, India, July 20th, 2014.

[8] K. Panda, P. Rajput, B. Shukla, *"FPGA implementation of 8, 16 and 32 bit LFSR with Maximum Length Feedback Polynomial using VHDL,"* Proceedings of the 2nd Int'l Conf. on Communication Systems and Network Technologies, Rajkot, India, (2012) May 11-13.

[9] Zulfikar, "Generating non Uniform Random Numbers Using Residue and Rejection Methods," *Jurnal Rekayasa Elektrika*, vol 8. no. 2, 2009.

[10] Zulfikar, Hubbul Walidainy, "Design of Linear Congruential Generator based on Wordlengths Reduction Technique into FPGA," *International Journal of Electronics Communication Computer Engineering (IJECCE)*, vol. 5 no. 4, July 2014.

[11] H. Walidainy, Zulfikar, "An Improved Design of Linear Congruential Generator based on Wordlengths Reduction Technique into FPGA," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 5, no. 1, February 2015.

[12] N. Bharatesh, S. Rohith, "FPGA Implementation of Park Miller Algorithm to Generate Sequence of 32 Bit Pseudo Random Key for Encryption and Decryption of Plain Text," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 2, no. 3, 2013 pages 99-105

[13] Lee Je-Hoon, Kim, Seong Kun., "Segmented Leap-Ahead LFSR Architecture for Uniform Random Number Generator," *International Journal of Software and Its Application*, vol. 7, no. 5, pp. 233-242, 2013.

[14] B. J. Abcunas, S. P. Coughlin, G. T. Pedro, D. C. Reisberg, "Evaluation of Random Number Generators on FPGAs," Worcester Polytechnic Institute, 2004

[15] M. A. Oorschot, P. C. Van, S. A. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1996 (5th Printing 2001).

[16] P. Alfke, "Application Note: Efficient Shift Registers, LFSR Counters, and Long Pseudo Random Sequence Generators," *XAPP 052, July 7, 1996.*

[17] *R. Ward, T.* Molteno, "Table of Linear Feedback Shift Registers," Technical Reports from the Electronics Group at the University of Otago, October 26, 2007.

## BIOGRAPHIES OF AUTHORS

**Zulfikar,** he was born in Beureunuen, Aceh, Indonesia, in 1975. He received his B.Sc. degree in Electrical Engineering from North Sumatera University, Medan, Indonesia, the M. Sc. Degree in Electrical Engineering from King Saud University, Riyadh, Saudi Arabia, in 1999 and 2011, respectively. He joined as teaching staff in the Department of Electronics at Politeknik Caltex Riau, Pekanbaru, Indonesia in 2003. He served as head of Industrial Control Laboratory, Politeknik Caltex Riau from 2003 to 2006. In 2006, he joined the Electrical Engineering Department, Syiah Kuala University. He has been appointed as head of Digital Laboratory for two successive years. His current research interests include VLSI design and System on Chips (SoC).

**Yuwaldi Away,** he was born in Tapaktuan, Aceh Selatan, Indonesia in 1964. He received his B.Eng degree in Electrical-Computer Engineering from "Sepuluh Nopember" Institute of Technology (ITS) Surabaya, Indonesia, the M.Sc degree from "Bandung" Institute of Technology (ITB) Bandung, Indonesia. He obtained his Ph.D. in Computer Technology from the National University of Malaysia. He joined as teaching staff in Syiah Kuala University start from 1990, and from 2007 until now he as a professor in Electrical Engineering. His current research interest includes the microprocessor-based system, embedded system, FPGA, optimation, and visualization.

**Rafiqa Shahnaz Noor,** she was born in Banda Aceh, Aceh, Indonesia, in 1992. She earned his B.Sc. degree Electrical Engineering from Syiah Kuala University, Banda Aceh, Indonesia. She joined as laboratory assistants in Data Laboratorium at Syiah Kuala University from 2012 to 2014.