Circuits, Systems, and Signal Processing (2023) 42:4784–4808
https://doi.org/10.1007/s00034-023-02332-3

Programmable Feedback Shift Register

Saleh Abdel-Hafeez[1,2]

Received: 19 May 2022 / Revised: 20 February 2023 / Accepted: 21 February 2023 /
Published online: 11 March 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Conventionally, a linear feedback shift register (LFSR) constitutes linear sets of sequences with predictable periods, which are considered vulnerable to intruders. Besides, an LFSR limits the sequences of scan-for-test patterns. This work introduces a private key, which is a collection of keywords, to program the feedback coefficients and initial states of the LFSR, where each keyword modulates the LFSR with a different polynomial of the same degree. That is, each polynomial generates a linear set of sequences. Consequently, the aggregate polynomial has a set of unpredictable and nonlinear sequences with high statistical randomness. The keyword, which holds the coefficients and initial states of the LFSR, can be managed in a few bits and stored in first-in first-out memory array. HSPICE simulations for 90 nm CMOS technology verify the functionality and speed of the proposed programmable feedback shift register (PFSR) of size 16-bit with 64 keywords. Results show a clock speed of 500 MHz with a power consumption of 73 μW and transistor count of 37,593, wherein the overall period has 4,194,240 unpredictable nonlinear sequences that surpass most LFSR structures. The PFSR can be suited for ASIC and reconfigurable HDL synthesis for efficient stream cipher and scan-for-test applications.

Keywords FIFO · HDL synthesis · Linear feedback shift register · Scan-for-test · Stream cipher

1 Introduction

Linear feedback shift register (LFSR) has become widely used in applications that are required to generate binary sequences of random numbers due to its simplicity in

✉ Saleh Abdel-Hafeez
s.abdelhafeez@qu.edu.sa

1 Department of Computer Engineering, Jordan University of Science and Technology, Irbid 22110, Jordan

2 Department of Computer Engineering, College of Computer, Qassim University, Buraydah, Qassim, Saudi Arabia

hardware and effectiveness in generating large distributing binary numbers with good statistical properties [3, 15, 26]. Such applications are related to tests boundary scan and functional tests coverage of hardware components about SoCs and embedded systems [4, 29]. Other applications of LFSR are related to support communication components such as cyclic redundancy check (CRC), low-density party check, and digital filters, for which LFSR considers an essential circuit [7, 31]. Additionally, LFSR has been recorded widely and successfully in the field of cryptography, where a large class of stream cipher is fully dependent on the binary sequence of LFSR [13, 25, 30].

Consequently, circuit designers of LFSRs investigate several improvements in circuit performances that are related to high-speed clock operation [12], multistage input–output configurations [21], low-power with high test coverage [5], and parallel architecture [35]. Furthermore, FPGA implementations of LFSRs consider faster development and easy to reconfigure for varieties of lengths and different seeds' taps [28, 36]. Besides, FPGA implementation offers flexibility in verifications and fast production to market [9].

Conventionally, LFSRs produce several sequences of binary numbers with high statistical randomness [6]. Additionally, LFSRs are sometimes referred to as linear recurrences, in which, the output values are given through a combination of some previous output values. Consequently, pseudorandom sequences are generated instead of true-random sequences [18]. Another drawback of the LFSR structure is related to the seeds' taps of the feedback coefficients. The coefficients remain constant while LFSRs generate the output sequences. The coefficients can be evaluated by a set of linear recurrence equations. Therefore, linear behavior dictates the LFSRs with the tendency to predict the sequences and the structure of the LFSRs, a limitation in cryptography applications [14, 20].

As a result, several outstanding attempts precluded the linear behavior within the period of sequences by introducing several nonlinear functions into the feedback coefficients while maintaining the maximum periods [17]. Briefly and not limited, [19] introduces a nonlinear feedback function in a form of m-subsequence to change the coefficient variables within the period. Another attempt uses Zech's algorithm of a cascaded connection of two LFSRs to dynamically change the coefficients while doubling the period size [33]. Further attempts use a polynomial modulator to change the coefficients dynamically while maintaining the sequence operation [11, 14, 17]. Additionally, [8] uses a pre-period type of function to keep inserting a new initial state and changing the period sequence dynamically. Others use parallel LFSRs with nonlinear output functions and good statistical properties [34]. Different directions make an effort toward a coupled linear congruential generator (CLCG) with a coupled variable input to alleviate linearity behavior and improve randomness for the fall of a large computational processing and high cost of hardware [27].

In alignment with the aforementioned attempts, we proposed the programmable feedback shift register (PFSR) that seeds new feedback coefficients and the initial states based on the degree m-bit of LFSR and the number of keywords. Therefore, the PFSR produces nonlinear sequences with multiple different linear periods since every keyword generates a distinct linear set of sequences within its period. A good analogy for this is to approximate a nonlinear function with several piecewise linear functions. The purpose of the keywords is to change the feedback coefficients from one set of sequences to another set of sequences, resulting in a non-periodic overall set of sequences and a non-predictable set of equations for better security and wider sequences. The major contributions of the proposed work are outlined with the following key features:

1. The main aim of the proposed PFSR design is to generate overall nonlinear unpredicted pseudo-random binary sequences that cannot be modulated with a linear set of equations and enjoys high statistical behavior.
2. The other objective of the PFSR of degree m-bit is to generate a large number of sequences. Since every keyword can modulate a number (i.e., length or period) of sequences of a total $(2^m - 1)$, the maximum number of generated sequences is equal to the number of keywords multiplied by $(2^m - 1)$. Thus, the PFSR aims to replace LFSR whenever the efficiency of cryptosystems and scan-for-test systems are needed.
3. Every keyword comprises feedback coefficients ($P_i s$) and initial states that seed the network structure of the LFSR with degree m-bit. Thus, every keyword has new $P_i s$ and initial states that generates a new set of sequences, wherein the overall sum of keywords generates an unpredicted and nonlinear set of sequences. In contrast, an LFSR with degree m-bit has fixed feedback coefficients ($P_i s$) and fixed initial states, such that the structure can be predicted by the coefficients $P_i s$ as well as the output states' sequences due to its deterministic period as defined by the following equation [25, 30]:

$$S_{i+m} = \sum_{J=0}^{m-1} P_j . S_{i+j} \mod 2; \quad S_i, P_j \in \{0, 1\}; i = 0, 1, 2, \ldots \tag{1}$$

4. The CMOS design of the proposed PFSR with degree m-bit has a FIFO that stores the keywords and a simple LFSR with degree m-bit. The FIFO operates at the low phase of the clock to initialize the network structure of the LFSR, wherein the LFSR processes the sequences at the high phase of the clock. Thus, the two-phase clocking system is used to shorten the critical path delay, and to avoid the cumbersome design related to setup/hold timing criteria [1]. Consequently, the design is stable and can be implemented with the low-cost standard CMOS library, an attractive feature for FPGA synthesis and ASIC designs.
5. The proposed PFSR is difficult to intrude by a side-channel attack through the use of reverse hardware since the initial states and the coefficients are dynamically changing by encoding new keywords [16], despite the fact that the reverse hardware attack can determine the structure of the PFSR and its degree. Furthermore, the decryption function has the same PFSR characteristics as the encryption function, in contrast to other nonlinear sequence generators that require high-cost reverse decryption functions.

In summary, Sect. 2 exploits the principle of programming coefficients and their impact on the nonlinearity and non-periodic pseudorandom numbers. Section 3 demonstrates the complete circuit architectural design of the proposed PFSR from input to output. Section 4 records the simulations and results with some performance features. Section 5 realizes comparisons with recent works. The conclusion is given in Sect. 6.

2 Keywords Principle

Conventionally, an m-bit LFSR consists of clocked storage elements (D-type flip-flops (DFFs)) and a network of feedback paths that manages by switches of coefficients $P_i s$ that direct the outputs of DFFs to XOR-sum logics as clearly illustrated in [25]. The number of DFFs represents the degree (m) of an LFSR, where the degree determines the number of sequences before being repeated; that is, the maximum length (period) is given by [25]:

$$L = 2^m - 1. \tag{2}$$

Assume the LFSR is initially loaded with values $S_0,\ldots, S_{m-1}$. The next output bit S_m of the LFSR, which is also the input to the left-most flip-flop, can be computed by the XOR-sum of the products of DFFs outputs and the corresponding feedback coefficients; thus, expanding Eq. (1):

$$S_m = S_{m-1}P_{m-1} + \ldots + S_1P_1 + S_0P_0 \quad \bmod 2$$

Intuitively, the general output sequences of the linear set of equations can be inspected from the above equation as follows:

$$\begin{aligned} S_{m+1} &= S_m P_{m-1} + \cdots + S_2P_1 + S_1P_0 \quad \bmod\ 2 \\ S_{m+2} &= S_{m+1} P_{m-1} + \cdots + S_3P_1 + S_2P_0 \quad \bmod\ 2 \\ &\ldots \\ &\ldots \\ S_{2m-1} &= S_{2m} P_{m-1} + \cdots + S_{m+1}P_1 + S_mP_0 \quad \bmod\ 2 \end{aligned} \tag{3}$$

As a result, Eq. (3) shows m linear sets of equations between the states and the next states, where $P_i s$ have constant values through all linear equations. Thus, Eq. (3) uniquely determines the coefficient $P_i s$. However, if the coefficients $P_i s$ change (variant) with the new states, then Eq. (3) becomes invalid and cannot be recognized for evaluating the coefficients $P_i s$, as illustrated by the next example.

2.1 Proof of Nonlinearity

For ease of understanding and brevity of discussion, let's start with the following simple example (Example #1) which is illustrated in Fig. 1. Let's start with the degree 2-bit LFSR structure that has two DFFs, where each DFF has a separate Set-Reset (SET$_i$, RES$_i$) for complete flexibility in setting the initial state. Besides, the feedback network consists of two controlling paths that gadget with multiplexers logic of select

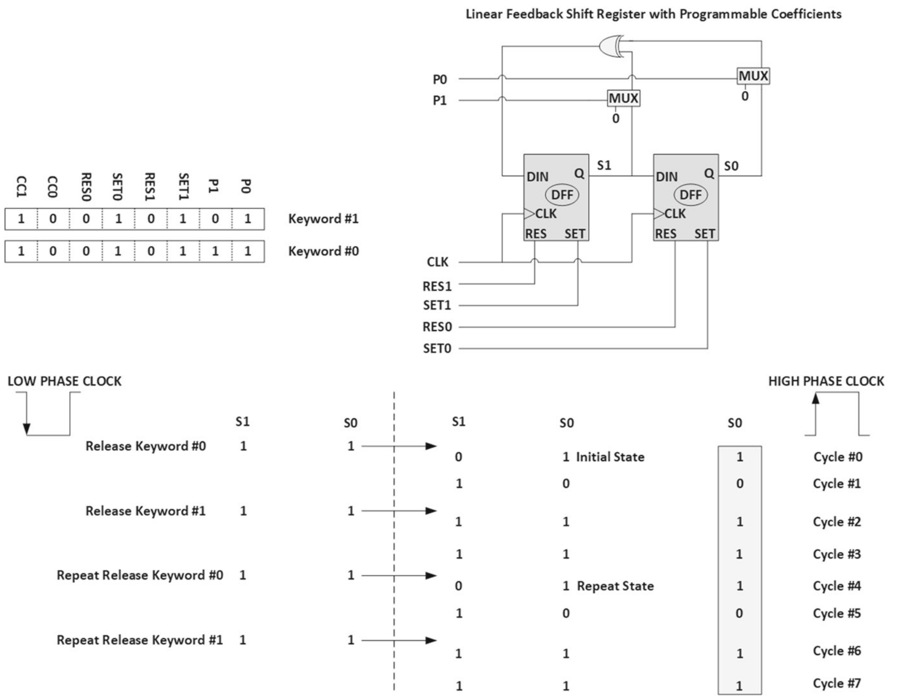


Fig. 1 Programmable feedback shift register (PFSR) based on Example #1

P_is. Consequently, each keyword contains the P_is, the SET$_is$, and the RES$_is$ values that seed the LFSR structure. Additionally, each keyword includes extra bits that determine the number of sequences that LFSR should go through before the next keyword is used.

Table 1 summarizes the keyword parameters with their associated definition and abbreviation based on Fig. 1. In this example, the counting cycle has two bits since the LFSR comprises two DFFs with a maximum counting period of three sequences. Example #1 is chosen for two keywords, where each keyword has a size of 8-bit with

Table 1 Keyword parameters

Parameter	Definition
P0	Programmable feedback of path “0”
P1	Programmable feedback of path “1”
SET0	The initial state of the first DFF for set
RES0	The initial state of the first DFF for reset
SET1	The initial state of the second DFF for set
RES1	The initial state of the second DFF for reset
CC0	The first bit of the counter cycle
CC1	The second bit of the counter cycle

hypothetical values recorded in Fig. 1. The values for the two keywords are stored in memory devices such as FIFO. Figure 1 illustrates Example #1 based on each clock cycle showing how the keywords impacted the sequences of the LFSR during the low phase and the high phase of the clock. The operation is processed as follows:

1. Fetch the keyword#0 from the FIFO during the low phase of the clock
2. Operate LFSR based on the value of keyword#0 for two sequences (i.e., CC1 = 1, CC0 = 0) during the high phase of the clock.
3. Fetch the keyword#1 from the FIFO during the low phase of the clock
4. Operate LFSR based on the value of keyword#1 for two sequences (i.e., CC1 = 1, CC0 = 0) during the high phase of the clock.

1- Repeat process 1 to 4 until the data are completed.

$$\begin{aligned} S_i &: 1\,0\,1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1\,0\,1\,1 \rightarrow \text{Generated from Figure 1} \\ X_i &: 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \rightarrow \text{Hypothetically chosen message text} \\ Y_i &: 0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0 \rightarrow \text{XOR - Logic generates the cipher text} \end{aligned}$$

Now, let's examine the generated sequence of Fig. 1 on the stream cipher and observe if we can get a linear set of equations capable of decrypting the cipher back to text. Let's assume the known plaintext for the two-byte X_is as simple as it gets to intrude, which is all ones. Subsequently, the generated cipher text (Y_i) can result from XOR logic as demonstrated below for the two bytes of the plaintext of data:

In general, the intruder has the information of the Y_i sequence and the first text header X_i, and thus tries to figure out the structure of the sequence S_i. Therefore, using the set of linear equations given in Eq. (3), we have:

$$\left.\begin{aligned} S_3 &= P_2S_2 + P_1S_1 + P_0S_0 \\ S_4 &= P_2S_3 + P_1S_2 + P_0S_1 \\ S_5 &= P_2S_4 + P_1S_3 + P_0S_2 \end{aligned}\right\} \bmod 2 \rightarrow \left\{\begin{array}{ll} 1 = P_2 + P_0 & \bmod 2 \\ 1 = P_2 + P_1 & \bmod 2 \\ 0 = P_2 + P_1 + P_0 & \bmod 2 \end{array}\right.$$

Thus, the intruder hopes that the derived coefficients $P2 = 0$, $P1 = 1$, and $P0 = 1$ with the initial state "101" determine the structure of the LFSR as clearly depicted in Fig. 2. Consequently, applying the sequence S_i of the LFSR depicted in Fig. 2 on the cipher text Y_i, we have:

$$\begin{aligned} S_i &: 1\,0\,1\,1\,1\,0\,0\,1\,0\,1\,1\,1\,0\,0\,1\,0 \rightarrow \text{Generated from Figure 2} \\ Y_i &: 0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0 \rightarrow \text{Intruder capture of cipher text} \\ X_i &: 1\,1\,1\,1\,1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,0 \rightarrow \text{XOR - Logic generates the message text} \end{aligned}$$

The sequence S_i that the intruder derived from the LFSR of Fig. 2 does not provide the message text X_i. Therefore, the linear set of equations given in Eq. (3) cannot be implied to derive the variant structure of LFSR due to its dynamic changes of feedback coefficients along with initial states during the generation of sequences.

As a result, the proposed PFSR structure shown in Fig. 1 generates a set of non-linear sequences that cannot be retrieved by the linear recursion equation (Eq. 3)

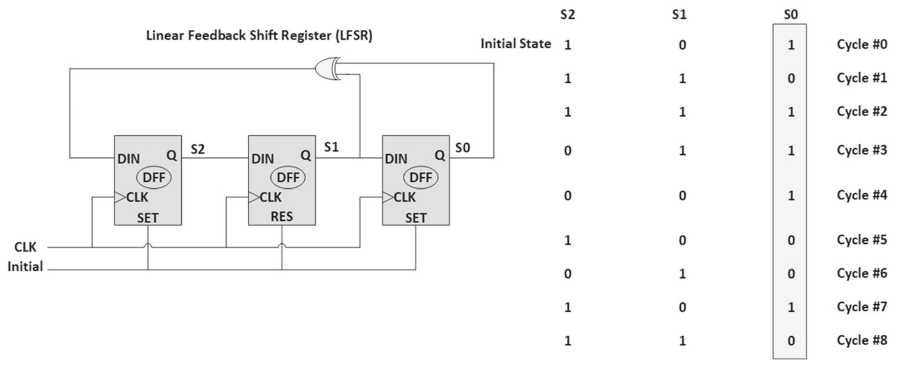


Fig. 2 Linear feedback shift register (LFSR) based on Example #1

since the keywords, which hold the coefficients, change during the time of generating the sequences. In such a case, the proposed PFSR module has several polynomial equations of degree m, where each polynomial is associated with the given keyword coefficients. That is, each polynomial releases a distinct pattern of sequences with high statistical properties. Notice that the PFSR module in Fig. 1 uses only two keywords, and the generated set of sequences is still nonlinear. The number of keywords can be increased depending on the following:

1. Cost of the FIFO memory array,
2. Complexity requirements of nonlinear order and size of the overall period.

Furthermore, the size of the keyword in bits can be measured based on the degree of the PFSR module. Referring to Example #1, which is demonstrated in Fig. 1, the PFSR with degree $m = 2$ has a keyword of size:

$$\text{Keyword size} = m + 2 * m + m = 4 * m \text{ bits.} \tag{4}$$

That is detailed as:

$$\begin{aligned}\text{Keyword size} = {} & \text{Feedback} - \text{Coefficients } (P0,\ P1) + \text{Set} \\ & - \text{Reset}(\text{SET0},\ \text{RES0},\ \text{SET1},\ \text{RES1}) \\ & + \text{Count} - \text{Cycle } (\text{CC0}, \text{CC1}) = 4 + 8 + 4 = 16\,\text{bits.}\end{aligned}$$

2.2 Proof of Randomness and Periodicity

LFSRs have good statistical properties for which the coefficients are chosen properly to give a maximum length [25, 30]. Moreover, an LFSR for a given degree m can produce sequences of different lengths (periods), depending on the feedback coefficients and initial state of the LFSR, such that LFSRs are often specified by irreducible polynomials using the following notation: An LFSR with a feedback coefficient vector

Table 2 Different lengths by various polynomials of 3-bit LFSR

Polynomial	Coefficients	Initial states	Length
$P(X) = X^3 + X + 1$	$P2 = 1, P1 = 1, P0 = 0$	DFF2 = 0, DFF1 = 1,DF0 = 0	7
$P(X) = X^3 + X^2 + 1$	$P2 = 1, P1 = 0, P0 = 1$	DFF2 = 0, DFF1 = 1,DFF0 = 1	7
$P(X) = X^3 + 1$	$P2 = 1, P1 = 0, P0 = 0$	DFF2 = 1, DFF1 = 1,DFF0 = 0	3
$P(X) = X^3 + X^2 + X + 1$	$P2 = 1, P1 = 1, P0 = 1$	DFF2 = 1, DFF1 = 0,DFF0 = 1	1

$(P_{m-1}, \ldots, P_1, P_0)$ is represented by the polynomial [25, 30]:

$$P(X) = X^m + P^{m-1}X_{m-1} + \cdots + P^2X_2 + PX + P_0 \tag{5}$$

For instance, the LFSR represented in Fig. 2 with a coefficient ($P_2 = 0, P_1 = 1, P_0 = 1$) can alternatively be specified by the polynomial $P(3) = X^3 + X + 1$. Consequently, Table 2 records different polynomials of the same degree (m), where each polynomial has a different length (period) [25, 30]. The varieties of primitive polynomials depend on the feedback coefficients and initial states. That is, if the coefficients and initial states are chosen properly, the maximum length of sequences can be generated. In general, there are many primitive polynomials for every given degree m. For instance, literature records that there exist 69,273,666 different primitive polynomials of degree $m = 31$-bit [25]. The linear complexity test of LFSRs that is represented by a primitive polynomial with degree m is ($L/2$) since there is an (L) number of generated sequences within the period of an LFSR primitive polynomial [30]. Example #2 illustrates the linear complexity test for the LFSR with degree $m = 4$ and polynomial $P(X) = X^4 + X + 1$. Figure 3 shows the dynamic behavior of the linear complexity test for the first n of generated sequences.

On the other hand, the proposed PFSR introduces several keywords along with LFSR of degree m, wherein every keyword seeds different coefficients to the LFSR to

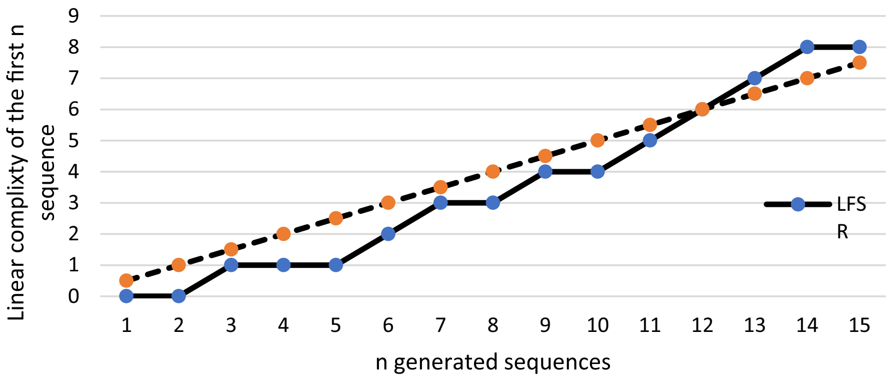


Fig. 3 Linear complexity profiles generated by $P(X) = X^4 + X + 1$ and initial state [0,1,0,0]

Table 3 Characteristic behavior of PFSR with $m = 4$ and two keywords

Keywords	Polynomial (primitive)	Initial state S3–S2–S1–S0	Counting cycle (length)
1–1–1–1–1–0–0–1–1–0–1–0–0–0–1–1	$P(X) = X^4 + X + 1$	0–1–0–0	15
1–1–1–1–0–1–1–0–1–0–0–1–1–0–0–1	$P(X) = X^4 + X^3 + 1$	1–0–0–1	15

generate a primitive polynomial with maximum length (L) as given in Eq. (1). Thus, every keyword has a good randomness property that almost enjoys $L/2$ linear complexity value. Furthermore, aggregating all polynomials' sequences results in overall good randomness properties. Table 3 illustrates the PFSR with degree $m = 4$-bit and two keywords, each representing a polynomial with a maximum length ($L = 15$) of generating sequences. Therefore, each polynomial has a linear complexity of $L/2$, such that the overall linear complexity is close to $L/2$ for a total of $L = 30$ generating sequences. Figure 4 shows the dynamic behavior of the linear complexity test for the PFSR demonstrated in Table 3. The results show the linear complexity test is slightly less than $L/2$ since the overall generating sequences are less by one sequence from the LFSR with degree m-5-bit. Employing the analysis of the linear complexity test, which is summarized in [27], the profile graph in Fig. 4 is close to $L/2$ in its overall period, in which $L = (2^4–1)*2 = 30$. Besides, the graph is observed as irregular staircases with an average height of 2 and an average length of 4 in its overall period. Additionally, the linear complexity is close to the graph's minimal overall period, which is $T = 30$. As a result, the linear complexity profile for the proposed PFSR is considered appropriate and fair for statistical pseudo-randomness representation.

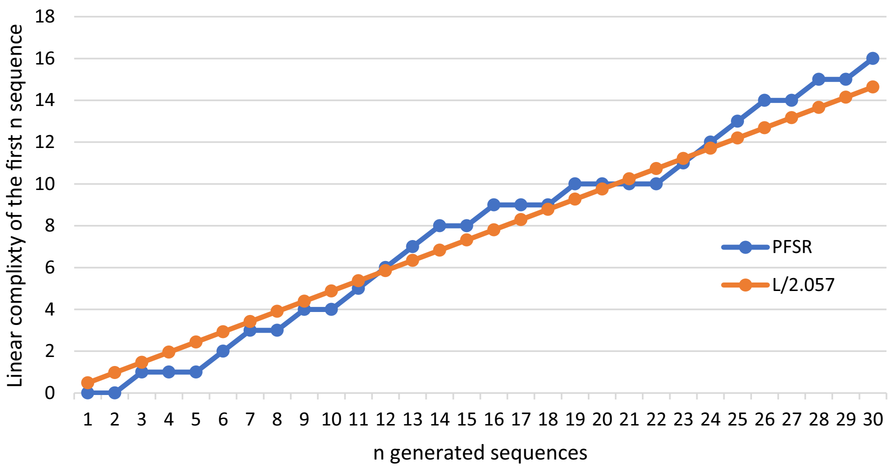


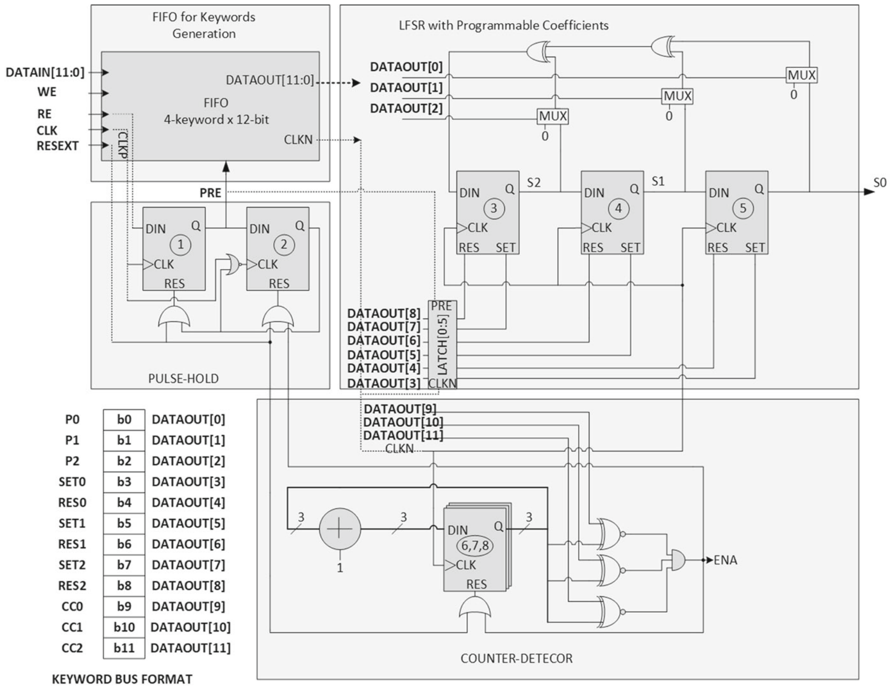
Fig. 4 Linear complexity profiles generated by PFSR of $m = 4$-bit and two keywords. The detailed characteristics of the PFSR are shown in Table 3

3 Circuit Design

The circuit objective of the PFSR is to provide new coefficients during the low phase of the clock while activating a new polynomial during the high phase of the clock. Thus, the two-phase clocking system (CLKP, CLKN) negates each other. The source clock (CLK) of a simple back-to-back NAND logic circuit or a more advanced circuit that is insensitive to PVT variations can generate the two-phase clocking system [10, 24]. Consequently, the FIFO, which provides the keywords, is triggered by CLKP, while the LFSR, which processes the keywords, is triggered by CLKN.

Figure 5 shows the proposed overall architectural circuit design of the PFSR with degree $m = 3$ as a choice for ease of explanation, where larger degrees also have the same structure but are wider in size. The size of the keyword is 12-bit based on Eq. (4), wherein the number of keywords is hypothetically chosen for four keywords. The keyword format recorded in Fig. 5 suggests that the first three bits hold the feedback coefficients (P0, P1, P2), the second six bits hold the Set-Reset for each DFF which are (SET0, RES0, SET1, RES1, SET2, RES2), and the last three bits (CC0, CC1, CC2) determine the number of count cycles. Additionally, the FIFO input and output data buses have the same size of keywords, which is 12-bit in Fig. 5, since the FIFO's purpose is to store and fetch all keywords in the same format.

The proposed PFSR in Fig. 5 constitutes three main blocks that are:



P0	b0	DATAOUT[0]
P1	b1	DATAOUT[1]
P2	b2	DATAOUT[2]
SET0	b3	DATAOUT[3]
RES0	b4	DATAOUT[4]
SET1	b5	DATAOUT[5]
RES1	b6	DATAOUT[6]
SET2	b7	DATAOUT[7]
RES2	b8	DATAOUT[8]
CC0	b9	DATAOUT[9]
CC1	b10	DATAOUT[10]
CC2	b11	DATAOUT[11]

Fig. 5 Architectural circuit design for the proposed PFSR with degree $m = 3$-bit and 4-keyword

1. The FIFO with size 4-keyword X 12-bit and the pulse-hold circuit,
2. the LFSR with degree $m = 3$-bit along with initial states and switches,
3. and a simple counter-detector circuit of size 3-bit.

The FIFO circuit is based on the synchronous design given in [24] with the addition of the pulse-hold circuit. The pulse-hold enables the FIFO to fetch one keyword by having a pulse of signal (PRE). Then, the pulse-hold holds the FIFO from doing any fetch operation until the counter-detector re-enables the pulse-hold, in which, the pulse-hold regenerates the pulse of PRE to fetch the next keyword. Holding the keyword by the FIFO is essential since the LFSR circuit needs to run several cycles (i.e., maximum period) before it requires another keyword from the FIFO.

The LFSR has three DFFs with a feedback path that combines pass-gate switches of three controlling coefficients (P0, P1, P2) which are directing the outputs of DFFs to XOR-sum logics. Moreover, each DFF has SET and RESET (SET-RESET) that control the initial state of the LFSR; thus, there are six SET-RESET (SET0, RES0, SET1, RES1, SET2, RES2). The six SET-RESET signals change the initial value of the LFSR for the first cycle of each keyword. Therefore, the six SET-RESET signals must be disabled for the consecutive cycles of the keyword. Subsequently, a small latch circuit with feedback reset is attached to the six SET-RESET signals as shown in Fig. 6.

The last component is the counter-detector circuit that orchestrates the FIFO fetches keywords operation by controlling the pulse-hold circuit, which triggers the pulse PRE signal. In other words, the counter-detector circuit controls the time of releasing the keywords from the FIFO to LFSR. Subsequently, each keyword's coefficients (CC0, CC1, CC2) determine the number of cycles that the LFSR requires to run before fetching a new keyword. Therefore, the counter-detector circuit is a simple counter with a 2-input XORs logic detector circuit that keeps counting until the value reaches the coefficients (CC0, CC1, CC2). Upon the counter-detector circuit detects the value of the count (CC0, CC1, CC2), it enables the pulse-hold circuit to activate a new pulse PRE signal to the FIFO, which releases a new keyword.

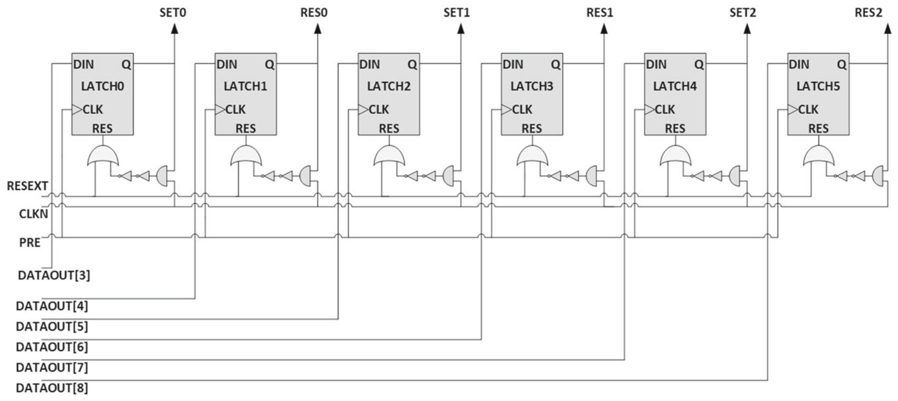


Fig. 6 Architectural circuit design of six latches with feedback reset

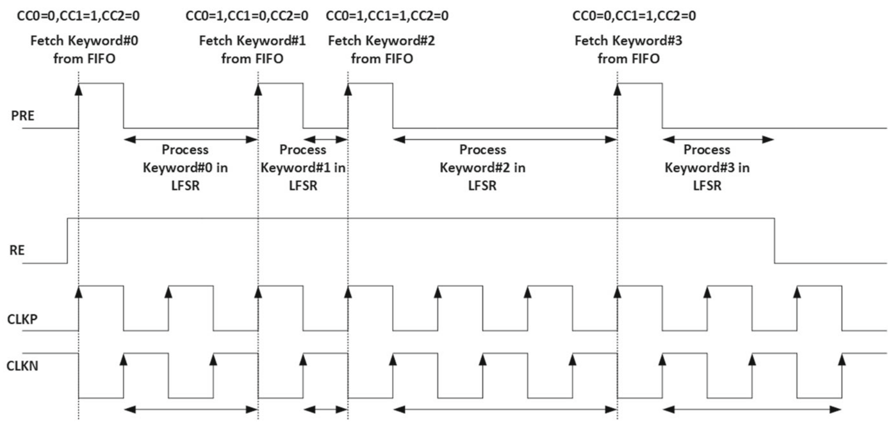

CC0=0,CC1=1,CC2=0
Fetch Keyword#0 from FIFO
CC0=1,CC1=0,CC2=0
Fetch Keyword#1 from FIFO
CC0=1,CC1=1,CC2=0
Fetch Keyword#2 from FIFO
CC0=0,CC1=1,CC2=0
Fetch Keyword#3 from FIFO
PRE
Process Keyword#0 in LFSR
Process Keyword#1 in LFSR
Process Keyword#2 in LFSR
Process Keyword#3 in LFSR
RE
CLKP
CLKN

Fig. 7 PFSR timing diagram design of fetching the keyword from FIFO

Figure 7 illustrates the timing diagram of the PFSR using the two-phase clocking system with four keywords. The pulse-hold signal (PRE) is initially activated due to the read enable (RE) signal request. First, the FIFO releases keyword#0 during the rising edge of the CLKP signal and holds keyword#0 for two cycles (i.e., two rising edges of CLKN) based on the coefficients data ($CC2 = 0$, $CC1 = 1$, $CC0 = 0$) of keyword#0. Second, once the counter-detector circuit detects the two cycles, it enables again the pulse-hold to release the PRE signal, which enables the FIFO to fetch keyword#1 at CLKP rising edge. The LFSR component processes Keyword#1 for one cycle due to coefficients ($CC2 = 0$, $CC1 = 0$, $CC0 = 1$) of keyword#1. Third, after one cycle, the counter-detector circuit requests the pulse-hold again to release the PRE signal, which enables the FIFO to fetch keyword#2 at CLKP rising edge. Finally, the LFSR runs sequences for three cycles before a new keyword is released since its count cycle is ($CC2 = 0$, $CC1 = 1$, $CC0 = 1$). Then, the next keyword is fetched from the FIFO, and the operation continues until the RE signal is de-asserted indicating the end of the text.

Since the two phases of the clock hold different operations, the critical path timing measures the worst timing of the two phases that determine the maximum clock running frequency. During the high phase of the clock, FIFO receives the PRE signal, which was enabled by the pulse-hold circuit, and thus the FIFO releases the keyword. Therefore, the critical path during high phase is:

$$\text{High} - \text{phase delay} = T_{\text{Pulse - Hold}} + T_{\text{FIFO}} \tag{6}$$

such that the $T_{\text{Pulse-hold}}$ is equal to the DFF access time, and the T_{FIFO} is the FIFO memory read access time, which was measured in [24] to be less than 1 ns (ns) for 90 nm CMOS technology of size 64-word × 64-bit. Therefore, Eq. (6) gives the high-phase delay time of about 1 ns since the DFF delay time is less than 0.05 ns.

Furthermore, the critical path due to the low phase of the clock comprises two paths running in parallel, which are the LFSR circuit and the counter-detector circuit.

Subsequently, the delay time for the LFSR circuit is:

$$\text{LFSR delay} = T_{\text{DFF}} + T_{\text{Mux}} + T_{\text{XOR}} \tag{7}$$

In parallel, the delay time of the counter–detector circuit is:

$$\text{Counter} - \text{detector delay} = T_{\text{Counter}} + T_{\text{Detector}} \tag{8}$$

Equation (8) has a delay of about 0.5 ns since the counter delay (T_{counter}) has a similar delay to a DFF in regards to the counter size as derived in [2]. Consequently, the low-phase delay is about 0.5 ns, as a conservative measure by taking into account the parasitic loading delay. On the other hand, the high-phase delay by Eq. (6) is depicted as 1 ns due to the FIFO read access time. Therefore, the maximum delay of the clock cycle is 2 ns ($F_{\text{clock}} = 500$ MHz), considering the worst-case duty cycle between the high and low phases of the clock. In conclusion, the critical path of m-bit PFSR is dictated by Eq. (6) during the high phase of the clock due to the FIFO operation.

Table 4 summarizes some of the characteristics of the PFSR for different degree (m) values. For example, the PFSR with degree $m = 16$-bit represents the LFSR component with sixteen DFFs, where the size of the keyword using Eq. (4) is 64-bit. Subsequently, The FIFO memory is chosen to store 64 keywords since it is recommended to have a regular layout structure for FIFO memory [1]. Thus, the size of the FIFO is 64-word X 64-bit. Furthermore, the maximum number of sequences (i.e., assuming maximum length) is given by Eq. (2), which is (2^{64}–1) *64 = 4,194,240. Finally, the clock frequency and the total number of DFFs are recorded as 500 MHz and 64 DFFs, respectively.

Table 5 summarizes the key features between PFSR and the conventional LFSR for a degree of $m = 16$. The PFSR enjoys much higher security features and wider generating

Table 4 Characteristics of the PFSR for different degrees and keywords

Degree m	Number of keywords	Size of FIFO	Maximum period	Clock frequency (MHz)	Number of DFFs
3	6	6-word X 12-bit	42	1000	12-DFF
6	16	16-word X 24-bit	1,040	750	16-DFF
10	40	40-word X 40-bit	4,092	500	40-DFF
16	64	64-word X 64-bit	4,194,240	500	64-DFF
20	80	80-word X 80-bit	83,886,080	400	80-DFF
32	128	128-word X 128-bit	549,755,813,760	300	128-DFF

Table 5 Key features of the PFSR versus LFSR for a degree $m = 16$

Type	Sequence	Maximum number of sequences	Memory keywords	Clock frequency (GHz)	Number of DFFs
LFSR	Linear	65,535	–	1	16-DFF
PFSR	Nonlinear	4,194,240	64-word X 64-bit	0.5	64-DFF

sequences than the LFSR for the trade of extra hardware such as memory and DFFs. Nowadays, technology scaling that reaches up to 3 nm produces this extra hardware at a very low cost. Additionally, most cryptography and scan-for-test components with the feature of wide nonlinear sequences are not simple and require complex designs with large hardware structures.

4 Simulations and Results

The PFSR proposed in Fig. 5 with the degree $m = 3$-bit and 4-keyword is designed and tested using HDL Verilog. The HDL implementation provides functional verification and faster reconfiguration for varieties of PFSRs. The synthesis is achieved through Xilinx ISE 14.7 and Virtex-5 XC5VLX50T FPGA development board for a 90-nm standard cell library at 1 V as per Virtex-5. Furthermore, the pulse-hold and the counter-detector circuits are simulated using HSPICE with 90 nm CMOS technology [22] to verify circuit characteristics, wherein the FIFO and LFSR have been approved widely in the literature.

Starting with Verilog simulations, Fig. 8 shows the functional verification of the proposed PFSR design based on the HDL implementation hardware structure shown in Fig. 5. For a clear graphical representation, Fig. 8 shows only CLKP since CLKN is the opposite signal. The second row shows the external-reset signal of the complete design (RESEXT) for initialization purposes, while the third row shows the read-request-enable (REN) signal for processing the text message, such that the REN signal is requesting the PFSR to start functioning and delivering sequences. Subsequently, the PFSR turns off when the REN signal is de-asserted. The moment the REN signal is asserted and CLKP rising edge occurs, the pulse-hold signal (PRE) activates the FIFO to release the first keyword (keyword#0).

The first keyword is shown on the output bus (DOUT [11:0]), that is "111011001001," indicating $P_0 = 1$, $P_1 = 0$, $P_2 = 0$, RES0 = 1, SET0 = 0, RES1 = 0, SET1 = 1, RES2 = 1, SET2 = 0, CC0 = 1, CC1 = 1, CC2 = 1. Subsequently, the first nine bits of the output bus (DOUT [8:0]) dictate the operation of the LFSR delivering the output sequence S0 during the cycles CLKN (i.e., the falling edge of CLKP). The S0 has a different pattern for each keyword; therefore, the S0 has unpredictable and nonlinear sequences. The last three bits on the output bus (DOUT [11:9]) give the number of cycles for each keyword that the LFSR should be running, that

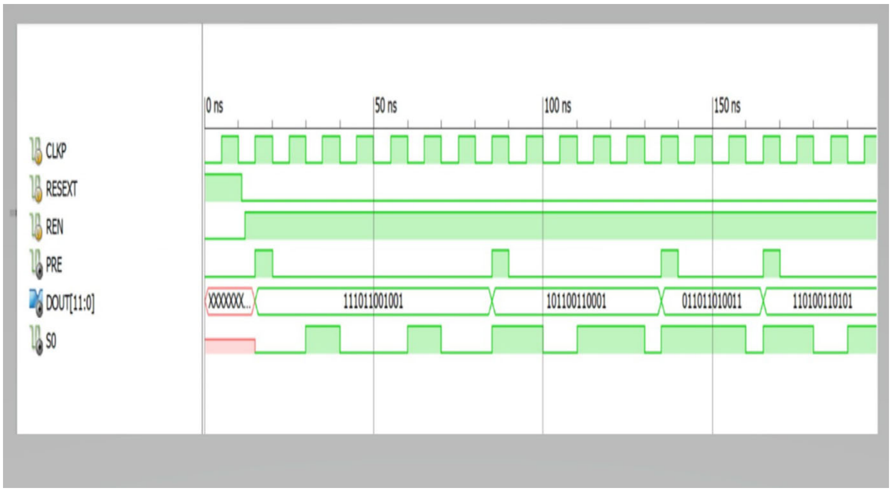


Fig. 8 PFSR Verilog simulations, input signals versus output signals

is seven cycles for the first keyword, in which, by the end of the seven cycles, the pulse of the PRE signal occurs (i.e., at the time of 85 ns) indicating the release of the second keyword (keyword#1), where the DOUT [11:0], at time 85 ns, appears as "101100110001." The operation continues until the REN signal is de-asserted, and thus, the PRE signal is stemming from releasing pulses and is turned off.

Table 6 shows the synthesis and simulation of three PFSRs with different degrees and keyword geometry. However, because of resource impediments of the

Table 6 Simulation and synthesis results for the PFSR with degree $m = 3, 10, 16$ bits

Performance	3-bit/6-keyword	10-bit/40-keyword	16-bit/64-keyword
Total number of sequences	42	40,960	4,194,240
Clock period	5 ns	13 ns	18 ns
Shift register	03	10	16
XOR gate	05	19	31
No. of slices	04	09	18
No. of slices flip-flops	03	10	16
No. of slices latches	06	20	32
No. of 4 i/p LUT	01	03	12
SRAM CELLs	72	1,600	4,096
SRAM IOs drivers	12	40	64
GCLK	01	01	01
(Gate + Net) delay	2.1 ns	2.1 ns	2.1 ns
Total pin	17	45	69

XC5VLX50T platform, we couldn't test beyond the degree of 16 bits. Table 6 lists the memory usage and simulation time of different PFSRs, which appears much worse than what we expect in the estimated frequency time recorded in Table 4. This is because the timing of the Xilinx XC5VLX50T platform includes input–output bus delay; besides, the memory cell circuit has 18 transistors instead of using only 8 transistors (8 T-Cell) as in custom design [23]. Additionally, the interconnecting, due to place-and-route between several lookup table logics, degrades the frequency and increases power consumption. Therefore, we recommend the use of FPGA for verification purposes and the realization of maximum polynomials. Subsequently, Table 6 records as well the total number of sequences for each PFSR, which surpasses most of the counterpart LFSR designs.

Another usage of FPGA is to help in collecting data for statistical measurement; the PFSR of $m = 16$-bit with 8-keyword results in the overall period; that is, $L = (2^{16}-1) \times 8 = 524{,}288$ of generating sequences is subjected to the linear complexity test and the NIST test for randomness measure [32]. The only constraint is that each keyword should represent a polynomial with a maximum length ($L = 2^{16}$–1) of generating sequences. Table 7 gives the detail of the keywords associated with polynomials that each generates a maximum length of $L = 2^{16}$–1 sequences.

Figure 9 shows the linear complexity profile of the sequence obtained by the PFSR with degree $m = 16$-bit and eight keywords. Thus, the overall period is $L = (2^{16}-1) \times 8 = 524{,}288$ of generating sequences. The resulting simulation shows that L/2 has an average height of 2 and an average length of 4 for a total $L/2 \approx$ of 262,144, which results in acceptable pseudo-random generating sequences.

Further random characteristic tests such as the National Institute of Standards and Technology (NIST) are conducted, as shown in Table 8, to verify randomness criteria [27, 32], such that the fifteen statistical benchmark tests are performed on 50 different collected sequences of length 524,288-bit generated by the PFSR presented in Table 7 by changing the initial conditions for every overall period until the 50 collected sequences are completed. Table 8 depends on the alpha (α) value for a selected accuracy of 0.05 and collected P values (UT) 0.0001 for passing the test with respect to chi-square distribution [32]. Thus, if the probability value of a pseudorandom bit sequence is greater than the threshold value α in all fifteen tests, then the sequence is assumed to be random [27]. The tests in Table 8 are: (1) frequency test, (2) frequency test within a block, (3) run test, (4) longest run of ones in a block, (5) binary matrix rank test, (6) discrete Fourier transform test, (7) non-overlapping template matching test, (8) overlapping template matching test, (9) Maurer's "Universal Statistical" test, (10) linear complexity test, (11) serial test, (12) approximate entropy test, (13) cumulative sums test, (14) random excursions test, and (15) random excursions variant test. Results show that the collected 50 sequences of size 524,288-bit have passed these tests. In conclusion, the proposed PFSR has overall good statistical properties that are close to LFSR with the advantages of a nonlinear set of equations (i.e., unpredicted period) and a wide range of generating overall period.

On the other hand, the HSPICE simulation for the PFSR, which is proposed in Fig. 5, is demonstrated in Fig. 10 by a two-phase clocking system of $F_{\text{clock}} = 500$ MHz at a slew rate of 1v/0.5 ns. The timing diagram shows the PRE signal releases pulses based on the number of cycles for each fetching keyword as was theoretically predicted in

Table 7 Characteristic behavior of PFSR with $m = 16$-bit and eight keywords

Keywords	Polynomial (primitive)	Initial state S15…….S0	Counting cycle (length)
1111111111111111–10–01–10–10–01–10–10–01–01–01–10–10–10–10–10–01–00–00–00–00–00–10–10–11	$P(X) = X^{16} + X^5 + X^3 + X + 1$	0100100111000001	$2^{16}–1$
1111111111111111–01–01–10–10–01–01–10–10–10–01–10–10–10–10–10–01–00–00–00–00–00–00–01–01	$P(X) = X^{16} + X^2 + 1$	1100110001000001	$2^{16}–1$
1111111111111111–10–01–10–10–10–01–10–10–10–10–10–10–10–10–01–10–00–11–01–00–00–00–00–01	$P(X) = X^{16} + X^{14} + X^{13} + X^{11} + 1$	0100100100000010	$2^{16}–1$
1111111111111111–10–01–10–10–01–10–10–01–10–01–10–01–10–01–10–10–10–10–00–00–00–01–00–01	$P(X) = X^{16} + X^{15} + X^{13} + X^4 + 1$	0100100101010100	$2^{16}–1$
1111111111111111–01–10–10–10–10–10–10–01–10–10–10–01–01–10–10–01–10–11–01–00–00–00–00–01	$P(X) = X^{16} + X^{15} + X^{13} + X^{11} + 1$	1001100100011001	$2^{16}–1$
1111111111111111–01–10–10–10–01–10–10–01–10–10–01–10–10–01–10–01–01–00–00–00–00–00–00–01	$P(X) = X^{16} + X^{14} + 1$	1000100100100101	$2^{16}–1$
1111111111111111–10–01–10–01–01–10–10–01–01–01–10–10–10–10–10–10–00–00–00–00–00–10–11–01	$P(X) = X^{16} + X^5 + X^3 + X^2 + 1$	0101000111000000	$2^{16}–1$
1111111111111111–10–10–10–01–01–01–10–10–01–01–01–10–01–10–10–10–00–01–00–00–00–00–10–11	$P(X) = X^{16} + X^{12} + X^3 + X + 1$	0001100111001000	$2^{16}–1$

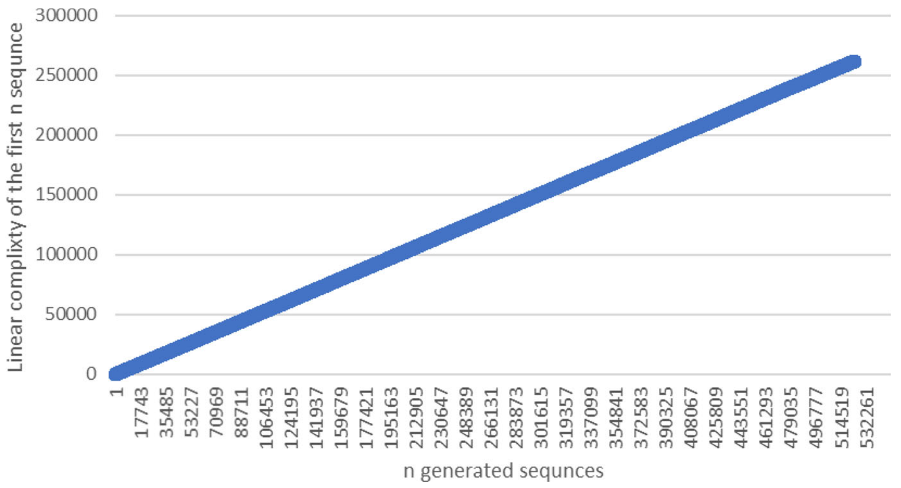


Fig. 9 Linear complexity profiles generated by PFSR of $m = 16$-bit and eight keywords. The detailed characteristics of the PFSR are shown in Table 7

Table 8 Statistical results for PFSR versus LFSR

NIST tests	PFSR *Prop. value (%)	LFSR *Prop. value (%)
1. Frequency test	95	97
2. Frequency test within block	99	97
3. Run test	96	94
4. Longest run of ones in a block test	98	96
5. Binary matrix rank test	97	98
6. Discrete Fourier transform test	96	95
7. Non-overlapping template matching test	94	97
8. Overlapping template matching test	99	97
9. Maurer's universal statistical test	98	93
10. Linear complexity test	97	96
11. Serial test	95	95
12. Approximate entropy test	97	96
13. Cumulative sums test	95	95
14. Random excursions test	96	95
15. Random excursions variant test	96	98

*Prop—the proportion is the ratio between the successes and the trials; that is, the accepted value is Prop. Value $\geq 95\%$, where the worst case is reported for all tests

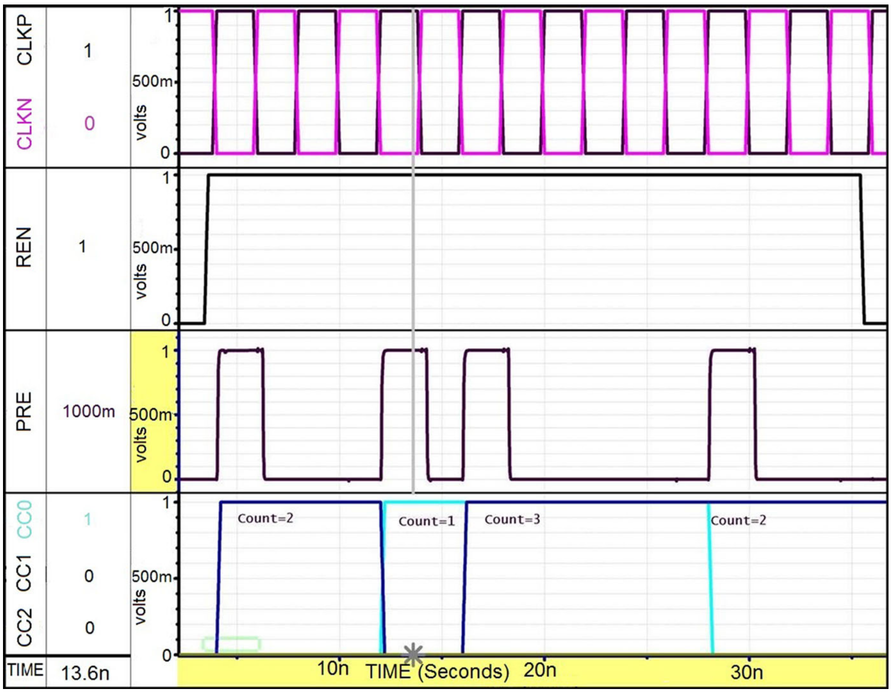


Fig. 10 HSPICE timing simulation for the proposed PFSR with $m = 3$, detail of PRE signal

Fig. 7. The pulse of the PRE signal is crucial since it indicates the starting time of fetching the keyword from the FIFO at CLKP. Subsequently, the PFSR starts releasing the sequences at the next cycle (i.e., the rising edge of CLKN). The detailed simulation in Fig. 10 shows the first row is the two-phase clock signals (CLKP, CLKN) with a slew rate of 1v/0.5 ns and $F_{\text{clock}} = 500$ MHz, and the second row shows the REN signal that enables the start of releasing pulses of the PRE signal. The third row shows the pulses of the PRE signal based on the count cycle (CC0, CC1, CC2) recorded in the fourth row. The PRE signal is nonlinear during all keywords since it is variant with different keywords. Therefore, the generated sequence is nonlinear and unpredictable. The counter-detector and pulse-hold circuits give exactly the theoretical timing diagram explained in Fig. 7. Besides, verifying the functionality and timing speed. As a result, the ASIC implementation of the PFSR utilizes most of the design characteristics' key factors. The design speed and power consumption can further be improved using recent FinFET CMOS technology. However, we aim to use basic standard CMOS library cells with cost-effective digital CMOS technology and improve the architectural circuit and design methodology.

The design setup/hold time between the internal feedback signals and the clock at the gated components gives a margin of half a clock cycle due to the use of the two-phase clocking system. Additionally, the design has a few input signals CLK, RESEXT, REN, WRE, DIN[11:0], and one output signal S0 that produces the nonlinear pseudorandom outcome. All keywords can be stored in the FIFO during the writing phase by asserting

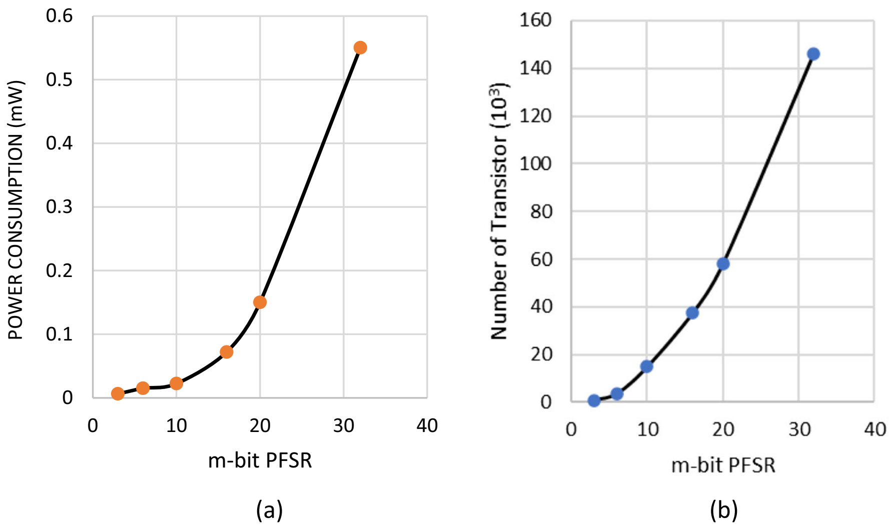


Fig. 11 Circuit characteristics of the PFSR with variable degree m. **a** Power consumption; **b** transistor count

the write enable (WRE) signal along with the clock source. Subsequently, the private key, which comprises keywords, can be dynamically changed for the next message resulting in a more secure and unpredicted nonlinear sequence.

Moreover, the power consumption records are slightly higher than standard LFSR designs due to memory components. However, we use the 8T-Cell, which is known for its low-power and high-speed advantages. The power consumption is about linear with $\log_2$ of memory size added to the number of DFFs, in which, every DFF takes about 0.047 μW. Consequently, Fig. 11a, b shows the total power consumption and transistor count for the PFSRs, which are characterized in Table 4.

5 Comparison Analysis

The proposed PFSR is compared against several configurations of nonlinear feedback shift registers [8, 11, 19, 33] that have the same aim of generating nonlinear pseudorandom sequences with a large period. Although our comparable designs are chosen to the best of our knowledge, the comparisons are not limited to the above references and can be generalized for a large class of nonlinear pseudorandom feedback shift register generators. Table 9 summarizes the differences concerning the maximum periods, type of polynomial seeds, nonlinear and random characteristics, hardware design complexity, and hardware attack with the type of decryption function. Furthermore, all values are recorded as they appeared in the references without any modification from our side. The degree is considered a common parameter for all recorded references.

The first column in Table 9 illustrates the maximum period of the comparable works, which shows that our proposed work considers the largest period due to multiple keywords. The second column shows the structure of the feedback polynomial function

Table 9 Comparison between prior works and our proposed PFSR

	Maximum periods ≤	Polynomial seeds	Nonlinear and randomness	Hardware complexity	Reverse attack/decryption
[19]	$(2^m - 1) + 0.4(2^m - 1)$	Eigenfunction + subsequence function	Poor nonlinear Poor randomness	Low complexity Fixed feedback High speed Low power	Vulnerable/large cost
[33]	$(2^{m1} - 1)*(2^{m2} - 1)$	Zech's or Jacobi's logarithms	High nonlinear Poor randomness	Low Complexity Fixed feedback High speed Low power	Vulnerable/large cost
[11]	$(2^m - 1) + 4000$	Polynomial Modulator	Poor nonlinear High randomness	Low Complexity Fixed feedback High speed Low power	Vulnerable/large cost
[8]	$2^m - S$ S = pre-period sequence	Pre-period sequence function	High nonlinear Poor randomness	Low Complexity Fixed feedback High speed Low power	Vulnerable/large cost
Our work	$(2^m - 1)$*number of keywords	A programmable feedback shift register	High nonlinear High randomness	FIFO Memory Configurable High-speed Moderate power	Difficult/same as encryption function

causing nonlinearity for each listed work. The cause of nonlinearity is due to the changing of seeds' taps by mathematical algorithms, tending to reduce randomness and lower the periods for the cost of having a nonlinear set of equations. On the contrary, our work obviates the shortcoming of polynomial functions by changing the seeds' taps and initial states by injecting a new keyword by taking advantage that there are many primitive polynomials for every length m. Column three lists the NIST tests measures of randomness, and we classify these recorded measures into "Poor" or "High" concerning their closeness to 100%.

The hardware configurations listed in column 4 use standard library LFSR units without modifications for low-power or high-speed circuitry. Our work has a higher cost design due to FIFO memory for the penalty of a large unpredicted overall period and a nonlinearity set of equations that represent the sequences. Although the FIFO memory might consider a high-cost design with comparable works, the use of the 8T-Cell memory array reduces the cost substantially. Furthermore, we consider that the listed works have the same comparable speed and can run at a clock speed of 500 MHz for $m \leq 16$ since they use the same LFSR structure with some feedback modulator mathematical functions. In our design, we split the critical path by using a two-phase clocking system to improve the performance of speed and power consumption.

The last column demonstrates the hardware attack difficulties among the listed works in a method called DE-cap silicon chip [27]. Since the comparable designs have a fixed structure of polynomial algorithms with pre-determined wiring connectivity between the polynomial algorithms and the LFSR states, they might be considered vulnerable to DE-cap hardware attacks. On the other hand, our work holds the polynomial algorithms in private keywords that dynamically changed; thus, the PFSR is difficult to intrude by a DE-cap attack. Furthermore, the last column elaborates on the cost of decryption functions, wherein our work shows the same encryption and decryption function along with the private key. On the contrary, other works require their encryptions to be of inverse residue mathematical functions with several mathematical constraints to be able to decrypt the same message.

As a result, our proposed design is considered comparable to analogous works and surpasses some of their essential key features. Such as wide period, unpredictability, and nonlinear pseudorandom number with high statistical properties. Besides, it is, to the best of our knowledge, a new method of generating a nonlinear set of equations by introducing a private key (i.e., collection of keywords) that are dynamically feeding the taps of LFSR with degree m. Finally, the proposed PFSR is recommended as an alternative replacement to the LFSR whenever the need for efficient generation of nonlinear and wide sequences, rather than as a comparable to cipher algorithms.

6 Conclusions and Future Work

In this work, we proposed the programmable feedback shift register (PFSR) with a degree m-bit and a private key (i.e., collection of keywords) as an alternative replacement for an LFSR for efficient scan-for-test generator and cryptography applications. The PFSR integrates first-in-first-out (FIFO) memory with linear feedback shift register (LFSR) through a two-phase clocking system. The FIFO releases the keyword to feed the LFSR; thus, generating a new polynomial with distinct sequence numbers per each new keyword. The private key can be a multiple of keywords, where each keyword contains coefficients, initial states, and a number of the counting sequence. The 16-bit PSFR with 64-keyword records an overall period of nonlinear binary pseudorandom numbers up to 4,194,240, surpassing most of the counterpart designs that have a degree of $m = 16$-bit. Therefore, each keyword presents a primitive polynomial that is modulated by the coefficients, initial states, and degree of the LFSR. This is due to the fact that there are many primitive polynomials for the given degree m with maximum length $(2^m - 1)$. Each polynomial generates binary sequences based on the number of count cycles for each keyword. The collection of all binary sequences is considered pseudorandom with unpredictable patterns and good statistical measures. Subsequently, the binary sequences cannot be modulated with the linear set of equations to reconstruct the structure of the PFSR.

The hardware of PFSR has similar components to LFSR with the addition of the FIFO memory array. The FIFO operates at one phase of the clock, while the LFSR operates at the other phase of the clock. Thus, the two-phase clocking system obviates the cumbersome design margin of setup/hold timing criteria, while reduces the critical path delay. The PFSR of degree $m = 16$-bit and 64 keywords operates at 500 MHz with a power consumption of 73 μW and total transistor counts of 37,593. The design is based on a 90 nm standard CMOS library with a 1 V power supply that is achieved in both ASIC and FPGA configurations.

Furthermore, the decryption PFSR is the same as the encryption PFSR with the same characteristics despite generating a nonlinear set of unpredictable patterns of binary sequences. Moreover, the PFSR design considers robust against physical DE-cap attacks (i.e., reverse hardware) since the private key is variant with messages and can be dynamically changed within the message itself. Future works are suggested to investigate the number and the structure of primitive polynomials having the same degree (m) that produces the maximum number of sequences ($2^m - 1$) since every polynomial is represented by a keyword. In addition, it is recommended to examine various security attacks along with substantial Brute-force bench tests are suggested to target the design in order to validate its security.

Data Availability Statement The datasets generated during and/or analyzed during the current study are not publicly available due to preserving the originality of the author's work but are available from the corresponding author upon reasonable request.

References

1. S. Abdel-Hafeez, A. Gordon-Ross, Reconfigurable FIFO memory circuit for synchronous and asynchronous communication. Int. J. Circuit Theory Appl. **49**(4), 938–952 (2021)
2. S. Abdel-Hafeez, A. Gordon-Ross, A digital CMOS parallel counter architecture based on state look-ahead logic. J IEEE Trans Very Large Scale Integr (VLSI) Syst **19**(6), 1023–1034 (2011)
3. P.M. Ajane Furth, E.E. Johnson, R.L. Subramanyam, Comparison of binary and LFSR counters and efficient LFSR decoding algorithm, in *IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, (Seoul, 2011), pp. 1–4
4. A. Alamgir, A.K. Bin A'ain, N. Paraman, U.U. Sheikh, I. Grout, A comparative analysis of LFSR cascading for hardware efficiency and high fault coverage in BIST applications, in *IEEE 29th Asian Test Symposium (ATS)* (Penang, Malaysia, 2020), pp. 1–5
5. W. Aloisi, R. Mita, Gated-clock design of linear-feedback shift registers. IEEE Trans. Circuits Syst. II **55**(6), 546–550 (2008)
6. Z. Chang, G. Gong, Q. Wang, Cycle structures of a class of cascaded FSRs. IEEE Trans. Inf. Theory **66**(6), 3766–3774 (2020)
7. P. Chen, An all-digital clock generator with modified dynamic frequency counting loop and LFSR dithering, in *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)* (Taipei, 2019), pp. 1–2
8. I.V. Chugunkov, B.V. Kliuchnikova, V.S. Tsyganov, M.A. Ivanov, V.I. Chugunkov, Pseudorandom number generators with predetermined period and pre-period, in *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, (St. Petersburg and Moscow, 2020), pp. 268–270
9. D. Datta, B. Datta, H.S. Dutta, *Design and implementation of multibit LFSR on FPGA to generate pseudorandom sequence number*, in *Devices for Integrated Circuit (DevIC)* (Kalyani, 2017), pp. 346–349
10. S. Furber, *ARM: System-On-Chip Architecture*, 2nd edn. (Addison-Wesley, New York, 2000)
11. M. Han, Y. Kim, Unpredictable 16 bits LFSR-based true random number generator, in *International SoC Design Conference (ISOCC)* (Seoul, 2017), pp. 284–285
12. G. Hu, J. Sha, Z. Wang, High-speed parallel LFSR architectures based on improved state-space transformations. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **25**(3), 1159–1163 (2017)

13. J.L. Imaña, LFSR-based bit-serial $GF(2^m)$ GF(2m) multipliers using irreducible trinomials. IEEE Trans. Comput. **70**(1), 156–162 (2021)
14. H. Jiang, C. Li, J. Fan, Research on pseudo-random characteristics of new random components, in *International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)* (Dublin, 2019), pp. 163–167
15. J. Jung, H. Yoo, Y. Lee, I. Park, Efficient parallel architecture for linear feedback shift registers. IEEE Trans. Circuits Syst. II **62**(11), 1068–1072 (2015)
16. M. Jurecek, J. Bucek, R. Lórencz, Side-channel attack on the A5/1 stream cipher, in *22nd Euromicro Conference on Digital System Design (DSD)* (Kallithea, 2019), pp. 633–638
17. S. Krishnaswamy, H.K. Pillai, On the number of linear feedback shift registers with registercial structure. IEEE Trans. Inf. Theory **58**(3), 1783–1790 (2012)
18. W. Li, X. Yang, A parallel and reconfigurable United Architecture for Fibonacci and Galois LFSR, in *7th International Conference on Intelligent Human-Machine Systems and Cybernetics*, (vol. 1, Hangzhou, China, 2015), pp. 203–206
19. H. Lv, J. Fang, J. Xie, P. Qi, Generating of a nonlinear pseudorandom sequence using linear feedback shift register, in *International Conference on ICT Convergence (ICTC)*, (Jeju, 2012), pp. 432–435
20. I.Z. Moghadam, A.S. Rostami, M.R. Tanhatalab, Designing a random number generator with novel parallel LFSR substructure for key stream ciphers, in *International Conference on Computer Design and Applications (ICCDA 2010)* (vol. 5, Qinhuangdao, 2010), pp. V5-598–V5-601
21. D. Morrison, D. Delic, M.R. Yuce, J. Redoute, Multistage linear feedback shift register counters with reduced decoding logic in 130-nm CMOS for large-scale array applications. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **27**(1), 103–115 (2019)
22. 90 μm CMOS ASIC Process Digests, Taiwan Semiconductor Manufacturing Corporation (Hsinchu, 2005)
23. K. Nii, Y. Masuda, M. Yabuuchi, Y. Tsukamoto, S. Ohbayashi, S. Imaoka, M. Igarashi, K. Tomita, N. Tsuboi, H. Makino, K. Ishibashi, H. Shinohara, A 65 nm ultra-high-density dual-port SRAM with 0.71um/sup ~/ 8T-Cell for SoC, in *Symposium on VLSI Circuits, Digest of Technical Papers* (Honolulu, 2006)
24. B. Nowacki, N. Paulino, J. Goes, A simple 1 GHz non-overlapping two-phase clock generators for SC circuits, in *20th International Conference on Mixed Design of Integrated Circuits and Systems (MIXDES)* (Gdynia, 2013), pp. 174–178
25. C. Paar, J. Pelzl, *Understanding Cryptography* (Springer, Berlin, 2010)
26. S. Palnitkar, *Verilog HDL: A Guide to Digital Design and Synthesis*, 2nd edn. (SunSoft Press, Mountain View, 2003)
27. A.K. Panda, K.C. Ray, A coupled variable input LCG method and its VLSI architecture for pseudo-random bit generation. IEEE Trans. Instrum. Meas. **69**(4), 1011–1019 (2019)
28. A.K. Panda, P. Rajput, B. Shukla, FPGA implementation of 8, 16, and 32 Bit LFSR with maximum length feedback polynomial using VHDL, in *International Conference on Communication Systems and Network Technologies* (Rajkot, 2012), pp. 769–773
29. I. Pomeranz, Direct computation of LFSR-based stored tests for broadside and skewed-load tests. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **39**(12), 5238–5246 (2020)
30. R.A. Rueppel, *Analysis and Design of Stream Ciphers* (Springer, Berlin, 1986)
31. S.A. Salehi, Low-cost stochastic number generators for stochastic computing. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **28**(4), 992–1001 (2020)
32. A statistical test suit for random and pseudorandom number generators for cryptographic applications, NIST special publications 800–22. My (2001)
33. J. Szmidt, Nonlinear feedback shift registers and Zech's logarithms, in *International Conference on Military Communications and Information Systems (ICMCIS)* (Budva, Montenegro, 2019), pp. 1–4.
34. T. Tuncer, E. Avaroğlu, Random number generation with LFSR based stream cipher algorithms, in *40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (Opatija, 2017), pp. 171–175
35. X. Zhang, A low-power parallel architecture for linear feedback shift registers. IEEE Trans. Circuits Syst. II **66**(3), 412–416 (2019)
36. P. Zode, P. Zode, R. Deshmukh, FPGA based novel true random number generator using LFSR with dynamic seed, in *IEEE 16th India Council International Conference (INDICON)*, (Rajkot, India, 2019), pp. 1–3

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.