Journal of Engg. Research Vol. 7 No. (3) September 2019 pp. 105-122

Designing programmable parallel LFSR using parallel prefix trees

Behrouz Zolfaghari, Mehdi Sedighi and Mehran S. Fallah

Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran
**Corresponding Author: msedighi@aut.ac.ir*

ABSTRACT

The throughput of an LFSR (Linear Feedback Shift Register) is affected by the sampling rate as well as the clock rate. On the other hand, the system-level characteristics of an LFSR such as its error detection capabilities are determined by the generating polynomial. Parallel LFSRs aim at improving the sampling rate in order to meet high throughput demands in parallel transmission or computation environments. Moreover, programmable LFSRs provide more system level flexibility by allowing different generating polynomials to be used. Thus, using programmable parallel LFSRs looks an attractive solution to improve both throughput and system-level parameters. Programmable parallel LFSRs can be useful in stream ciphers, microprocessors, and many other environments. But parallelism and programmability can reduce the clock rate by increasing the logical depth and increase power and area by increasing the number of gates. Thus, we will need an efficient solution to manage the tradeoffs. This paper proposes an approach based on Parallel Prefix Trees (PPTs) to design programmable parallel LFSRs. PPTs are a family of topologies previously used in the design of parallel arithmetic circuits in order to manage the tradeoff between different circuit-level parameters. Our approach allows designers to use different PPTs in order to improve different circuit level parameters. A sample PPT-based programmable parallel LFR is designed and evaluated. Empirical results show more than 23% improvement in throughput and more than 27% improvement in area compared to state-of- the-art programmable parallel LFSR architectures.

Keywords: Parallel Prefix Tree; Parallel LFSR; Programmable LFSR; Brent-Kung; Programmable Parallel LFSR.

INTRODUCTION

Linear Feedback Shift Registers (LFSRs) are widely used in real-world applications such as generating and checking error detection codes (Wu, 2015; Parhi, 2004; Zhang et al., 2005), sequence generation and Pseudo-Random Number Generation (PRNG) (Li et al., 2016; Rahimov et al., 2011), Automatic Test Pattern Generation (ATPG) (Acevedo et al., 2016; Pomeranz, 2016), Built-In Self-Test (BIST) (Ying et al., 2018; Xiang, el at., 2017; Yasodharan et al., 2014; Acevedo et al., 2015), coding and cryptography (Mashhady et al., 2015; Upadhyay et al., 2015; Matsui, 2014; Lee et al., 2014), and modular arithmetic computation (Morales-Sandoval et al., 2009). Therefore, these circuits are of much importance to the designers and the researchers (Li et al., 2017; Wang et al., 2016). LFSRs may occasionally be implemented in software (Delgado-Mohatar et al., 2011), but the common trend is to implement them in hardware by forming a shift register with a feedback loop and a number of $GF(2)$ addition elements (XOR gates) that are essentially used to accomplish $GF(2)$ polynomial division.

An LFSR is specified by its generating sequence that determines the locations of $GF(2)$ addition elements on the feedback loop. The binary generating sequence can be represented by a $GF(2)$ polynomial that is called the generating polynomial. The degree of an LFSR is defined as the number of flip-flops constructing the LFSR or equivalently the degree of its generating polynomial.

There are two types of LFSRs: Fibonacci-type LFSRs and Galois-type LFSRs (Wei eat al., 2015; Pomeranz, 2017). The XOR gates lay on the feedback loop in the former type and out of the feedback loop in the latter. The two types can be converted to each other by reversing the generating sequences and selecting proper initial values (Dubrova, 2009). Figure 1 shows two counterpart LFSRs, one of which is of Fibonacci type and the other is of Galois type. In this paper, the focus is on Galois-type LFSRs because they are more common in real-world applications due to their lower delay in the feedback loop, which allows higher clock frequencies. We also assume that LFRS are implemented using XOR gates although they have occasionally been implemented using XNOR gates (Ahmad et al., 2008).

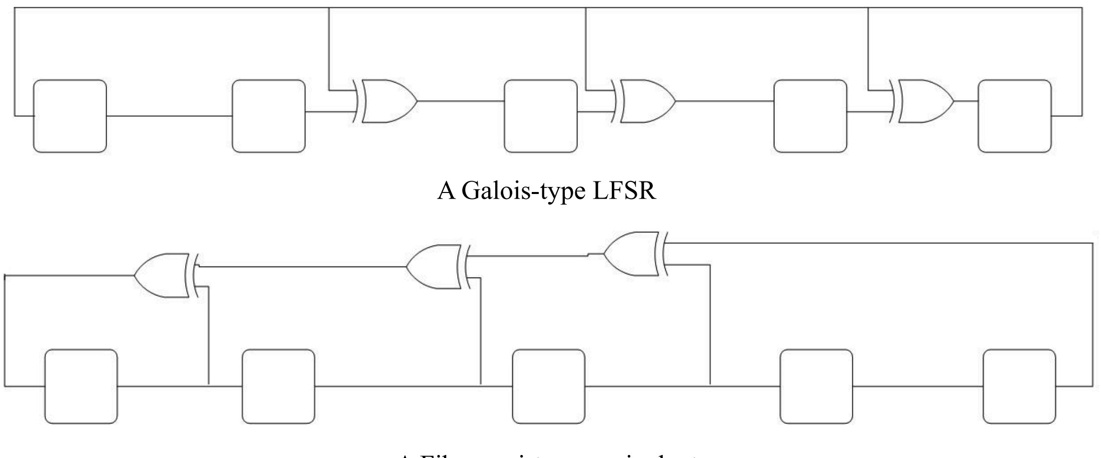


Fig. 1. Fibonacci and Galois type LFSRs.

A programmable LFSR is an LFSR that can operate on any generating sequence with a given length in contrast with a static LFSR, which operates only on a specific generating sequence.

Programmability provides system-level design flexibility by allowing the designers to select among different generating sequences. For instance, in LFSR-based error detection systems, programmability allows detecting different categories of errors by choosing different generating sequences. Programmable LFSRs have been of particular interest for researchers during the last decades (Ren et al., 2015; Gai et al., 1986; Toal et al., 2009; Grymel et al., 2011). For example, it has been proposed to use a single programmable LFSR in a microprocessor for many applications such as cryptography, BIST, and PRNG, each of which requires its own generating function (Gai et al., 1986). A programmable LFSR is shown in Figure 2.

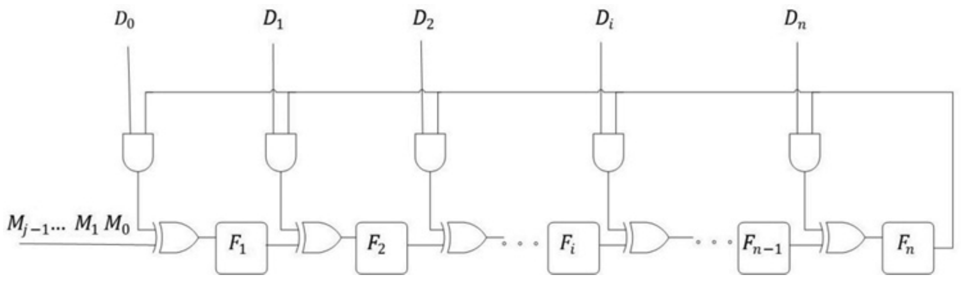


Fig. 2. A programmable LFSR of degree n.

In this figure, F_1 through F_n are the flip-flops that form the shift register and the XOR gates perform the $GF(2)$ additions if enabled by the corresponding AND gates according to the generating sequence $(D_n \dots D_i \dots D_1 D_0)$. The input sequence $(M_i \dots M_1 M_0)$ is fed into the LFSR one bit per clock. It should be noted that there may be LFSRs, into which no data enters except the initial values loaded in flip-flops. In such a case, it can be assumed that $M_i = \cdots = M_1 = M_0 = 0$.

Since LFSRs sample one single bit in each clock cycle, they may not be able to provide adequate throughput in applications where data stream arrives at a rate higher than one bit per clock or in applications in which data should be processed in words. This problem can be solved by using parallel LFSRs. An n-bit j-parallel LFSR ($j > 1$) is a circuit that performs the same function as an ordinary n-bit LFSR but samples j bits in each clock cycle. Therefore, parameter j can be viewed as the sampling rate of a parallel LFSR.

The two notions of programmability and parallelism in LFSRs can be combined together to form a programmable parallel LFSR, in which the generating sequence can be changed depending on the application while the overall structure remains parallel. Figure 3 shows a schematic representation of such a circuit. In this figure, F_i^k represents the value of flip-flop F_i after k iterations. The oval box represents a combinational circuit that takes F_1^0 through F_n^0 along with M_0 through M_{j-1} and D_0 through D_n as input and produces F_1^j through F_n^j as output.

If, in addition to the generating sequence, the operational mode of a parallel LFSR is also adjustable, then a reconfigurable parallel LFSR is formed. It has been shown that this level of reconfigurability has an adverse effect on the logical depth and performance of a parallel LFSR (Zibin et al., 2013; Savic et al., 2014). On the other hand, programmability of the generating sequence often provides sufficient flexibility without sacrificing performance (Toal et al., 2009; Grymel et al., 2011). Therefore, this paper focuses only on programmable parallel LFSRs.

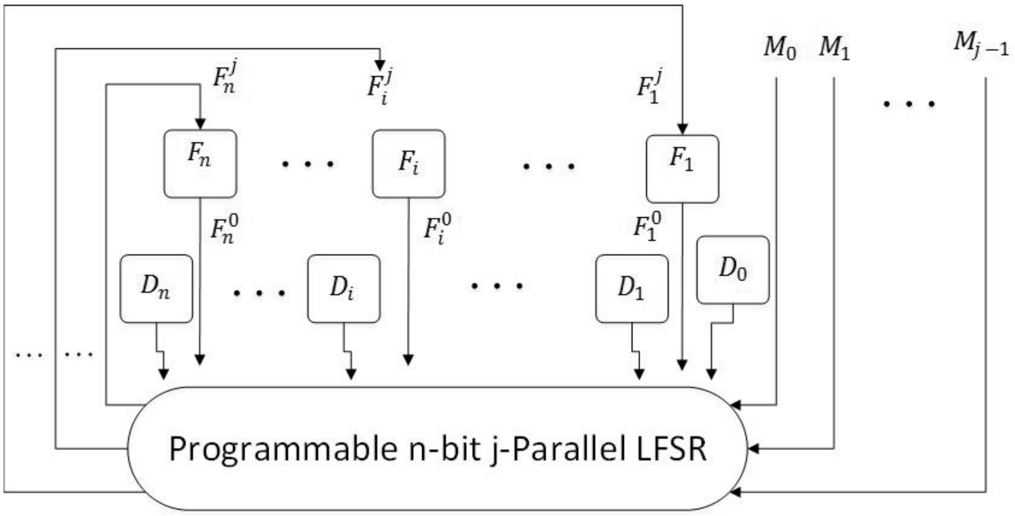


Fig. 3. A programmable n-bit j-parallel LFSR.

Programmable parallel LFSRs, as they are, provide system level design flexibility as well as high performance. What we are seeking in this paper is adding circuit level design flexibility. We are going to present a flexible architecture for designing programmable parallel LFSRs that allows designers to select among a variety of design options. This will make it possible to choose among various tradeoff points depending on design objectives and constraints on sampling rate, clock period, and area. Similar research works have previously been presented in the area of arithmetic circuits on the basis of PPTs (Zarandi et al., 2014). Among the PPTs used in arithmetic circuit design, we can refer to Brent-Kung, Kogge-Stone, Sklansky, Knowles, Han-Carlson, and Lander-Fischer PPTs (Harris, 2003). Taxonomies and characterizations presented in this regard show that selecting different kinds of PPTs can improve different circuit level parameters. This helps the designers maneuver in a space of design schemes to manage the tradeoffs according to design constraints and objectives (Harris, 2003; Hoe et al., 2011).

The rest of this paper is organized as follows. Section 2 studies relevant works. Section 3 establishes a relation between programmable parallel LFSRs and PPTs. Section 4 discusses the design details. Section 5 presents the results of evaluations. Section 6 concludes the paper and suggests further works.

RELATED WORKS

There are two categories of research works that can be considered relevant to this work. The first category consists of those related to the design of reconfigurable parallel LFSR and the second includes those related to PPT-based parallel logic circuits. Each of the two categories will be discussed in the next section.

Reconfigurable and Programmable Parallel LFSR

The design of parallel LFSR has been the focus of research for the last two decades (Hu et al., 2017; Kim et al., 2015). There are various approaches to design such circuits (Panda et al., 2014; Singh et al., 2013; Manikandan et al., 2013). For example, an approach based on the Chinese Remainder Theorem (CRT) has been proposed in Chen (2009). An approach based on transition and control matrices has also been proposed in Grymel et al. (2011). There are also approaches based on division on shorter generating polynomials with less feedback terms (Glaise, 1997). Moreover, there are special techniques for designing parallel LFSRs for specific applications. For example, in Condo et al. (2014) a variable-parallelism parallel LFSR has been proposed for 3GPP-LTE/LTE-Advanced applications. As another example, a parallel CRC computation circuit convenient for SoCs has been proposed in Toal et al. (2009). But the most relevant works are those proposing approaches based on unfolding, mathematical deduction, and recursive equations.

A common approach for the design of parallel LFSRs is unfolding, which essentially aims at revealing hidden concurrencies in DSP programs. Unfolding was proposed and developed later as a technique for mitigating the problem of high fan-outs in LFSRs that affects the clock frequency (Zhang et al., 2005). Since this method leads to long clock periods and this adversely affects the throughput, it needed some optimizations in order to be convenient in high throughput parallel applications. For this purpose, techniques such as pipelining and retiming were considered later (Cheng et al., 2006). It has been proven that increasing the unfolding factor more than a specific threshold decreases throughput because of notable increase in the clock period. The literature comes with techniques for alleviating this problem (Cheng et al., 2009; Ayinala et al., 2011). Among these techniques, we can refer to those based on Look-ahead transformations (Lin et al., 2013).

Mathematical deduction (Parhi, 2004) is another approach that has been used by researchers to design parallel LFSRs. The main idea behind mathematical deduction is focusing on the mathematical function of the LFSR and designing circuits, which can implement the combined function of multiple iterations. Mathematical deduction may be based on recursive equations (Parhi, 2004).

There are also research works that present approaches based on look-ahead for designing parallel LFSRs (Lin et al., 2013).

On the other hand, there are several applications in which it is very useful to choose among a number of generating polynomials. For instance, we can refer to microprocessors performing LFSR instructions, multiple-standard modems, stream ciphers, and pseudo-random number generators. Programmable and reconfigurable LFSRs have been introduced in response to this demand (Ouahab et al., 2017; Mishra et al., 2016; Lama et al., 2016). Combining the notions of parallel LFSR and reconfigurable/programmable LFSR seems a natural idea to achieve the advantages of both notions (Wei et al., 2015). However, the existing programmable parallel LFSRs in the literature suffer from the lack of circuit-level design flexibility. In other words, their designs do not provide enough degree of freedom to efficiently manage the tradeoffs among different circuit-level design parameters such as delay and area.

The most relevant research works to which we compare our proposed architecture are Toal et al. (2009) and Grymel et al. (2011). A 4.92Gbps field programmable parallel CRC (Cyclic Redundancy Check) calculator has been designed, synthesized, and mapped to 130-nm UMC library in Toal et al. (2009) based on matrix calculations. In the proposed architecture, called cell array architecture, the main component consists of a number of configurable cells. Each cell includes two multiplexers, a configuration register, and an XOR gate. A preprocessing stage consisting of a number of XOR gates along with a post-processing stage performing matrix multiplications is added to the main component. Another 15.38Gbps programmable parallel LFSR has been designed and mapped to the same library in Grymel et al. (2011). The latter programmable parallel LFSR uses XNOR gates and latches instead of multiplexers in the main component. We compare our proposed programmable parallel LFSR to these designs.

PPT-based parallel logic circuits

Parallel prefix trees have been studied for the last few decades (Harris, 2003). PPTs can be used as part of a parallel solution to any recursive equation provided that the recursive equation is stated using an associative operation. There are various families of PPTs with the same functionality but different structures (Jaberipur et al., 2015; Abdel-Hafeez et al., 2013; Hobson, 2015; Kumar et al., 2015). Their structural differences create differing implementation complexity (Sergeev, 2013), depth (Lin et al., 2009), deficiency (Zhu et al., 2006), fan-out (Lin et al., 2009), problem-size-independability (Lin et al., 2009), capability of running on parallel machines (Sergeev, 2013), convenience for running on pipeline systems (Santos, 2002), application in different branches of science (Lin et al., 2013), and implementation technology (Lin et al., 2003). PPTs have been previously used in the design of various parallel logic circuits. PPT-based parallel adders have been well studied, developed, and evaluated (Lina et al., 2005). The literature also comes with parallel priority encoders (Huang et al., 2002), parallel comparators (Abdel-Hafeez et al., 2013), parallel round robin arbiters (Ugurdag et al., 2012), and parallel, reverse converters (Panda et al., 2014) designed using PPTs.

PPTs AND PROGRAMMABLE PARALLEL LFSRs

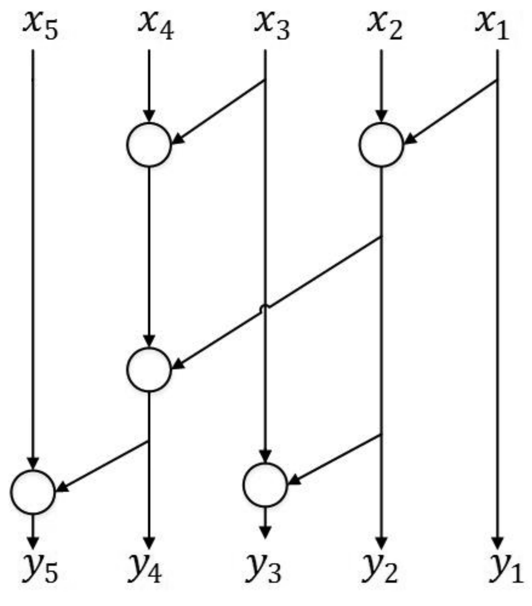
PPTs are topologies originally presented to be used in prefix processing, which is a kind of parallelizable recursive computation. The prefix processing problem is the problem of calculating $Y = [y_1, y_2, \dots, y_i, \dots, y_n]$ from $X = [x_1, x_2, \dots, x_i, \dots, x_n]$ where $\forall k \in [1{:}n],\ y_k = x_k \circ \dots \circ x_i \dots \circ x_2 \circ x_1$ and "∘" is an associative operation. In this paper, we define a LOO (Last Output Only) prefix processing problem as the problem of as the problem of calculating only $y_n = x_n{}^\circ \dots {}^\circ x_i{}^\circ \dots {}^\circ x_1$ from $X = [x_1, \dots, x_i, \dots {}^\circ x_n]$.

PPTs can also be modified to parallelize computations formalized as LOO prefix processing problems. To do this, we should simply remove the edges and nodes not contributing to the calculation of y_n. Figure 4-a shows the Brent-Kung PPT for $n = 5$. Moreover, Figure 4-b shows a LOO PPT designed on the basis of the Brent-Kung topology for $n = 5$.

An obvious serial algorithm to solve a classical prefix processing as well as a LOO prefix processing problem is as follows.

```
$y_1 \leftarrow x_1$
For $i = 2$ to $n$                Algorithm (1)
    $y_i \leftarrow y_{i-1} \circ x_i$
```

Let us assume that each "∘" operation takes a single clock cycle to be accomplished using a single processing unit. The above serial algorithm will obviously take 4 clock cycles to accomplish assuming $n = 5$. But the topologies in Figure 4 can solve both classical and LOO prefix processing problems in 3 cycles using two processing units. This improvement will get more significant for larger values of n.



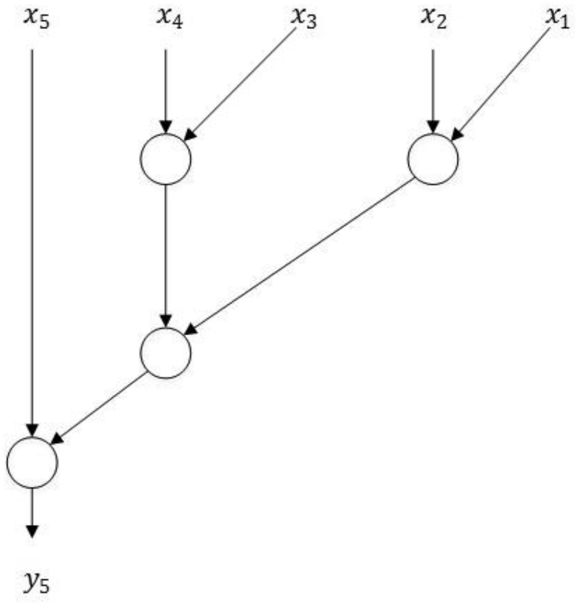


Fig. 4-a The Brent-Kung topology for n = 5 **Fig. 4-b** The Brent-Kung LOO PPT for n = 5

Fig. 4. Sample PPTs for solving classical and LOO prefix processing problems.

There are two general challenges that need to be handled in order to use PPTs for parallelizing every sequential circuit. The first challenge is to restate the function of the sequential circuit in the form of a proper recursive equation. The second is to define a proper associative logic operation to convert the recursive equation to a prefix processing problem. These challenges motivate several research works (Esposito et al., 2016; Gurusamy et al., 2016; Hepzibha et al., 2016).

Let us resolve the first general challenge in the design of PPT-based programmable parallel LFSRs by deriving a recursive description of the sequential programmable LFSR shown in Figure 2. To do this, we derive the set of equations that state the value of each individual flip-flop at the end of the kth clock cycle, in terms of the values of the flip-flops in the $(k-j)$th cycle. This set of equations is referred to as the jth State Transition Equation System (STES) in this paper. The logic function of the programmable LFSR of Figure 2 can be modeled by the following equation.

$$\begin{gathered} F_i^k = D_{i-1}F_n^{k-1} + F_{i-1}^{k-1} \\ k \in [1,j] \end{gathered} \tag{1}$$

In Equation 1, addition and multiplication operations are in $GF(2)$, F_i^k denotes the value of F_i in the kth iteration, and F_n^{k-1} is the value of F_n in the $(k-1)$th cycle or equivalently the value on the feedback loop in the kth cycle. The initial value of the ith flip-flop is shown by F_i^0. Moreover, D_i represents the ith least significant bit of the generating sequence. It is assumed that $D_n = 1$ and $D_0 = 1$ because the generating polynomial must be prime and of degree n and indivisible by the GF(2) polynomial "x".

During the recursive application of Equation 1, D_i should be replaced by 0 if $i < 0$. Moreover, F_i^0 should be replaced by M_{-i} for $i \le 0$ in recursive applications of the equation.

The recursive nature of Equation 1 shows that we have overcome the first general challenge.

Now let us handle the second general challenge by defining a proper associative logic operation through which Equation 1 can be converted to a prefix processing problem. It can be easily shown that the AX (AND-XOR) operation, defined as follows, can meet this requirement.

$$a \uplus^{D} b = a + D.b$$ Equation

The circuit shown in Figure 5 implements an AX operation. This circuit can be used as a building block in the proposed programmable parallel LFSR.

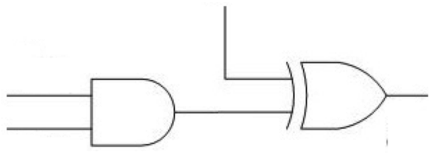


Fig. 5. Logic implementation of the AX operation.

There is still an extra challenge to be handled, which is specific to programmable parallel LFSRs. The problem here is that the recurrence in Equation 1 should be resolved for two indices (i and k), while classical PPTs have been originally proposed to solve single-index recursive computing problems. We solve this problem by decomposing Equation 1 into two LOO prefix processing problems resolved using two cascaded stages of PPTs. The first PPT stage calculates F_n^k (the value on the feedback loop in the kth cycle) for $k \in [1,\ j-1]$. The second stage calculates F_i^j for $i \in [1, n]$. The jth STES of a programmable parallel LFSR consists of the two mentioned equations. Figure 6 shows the architecture of a PPT-based programmable parallel LFSR designed based on this approach.

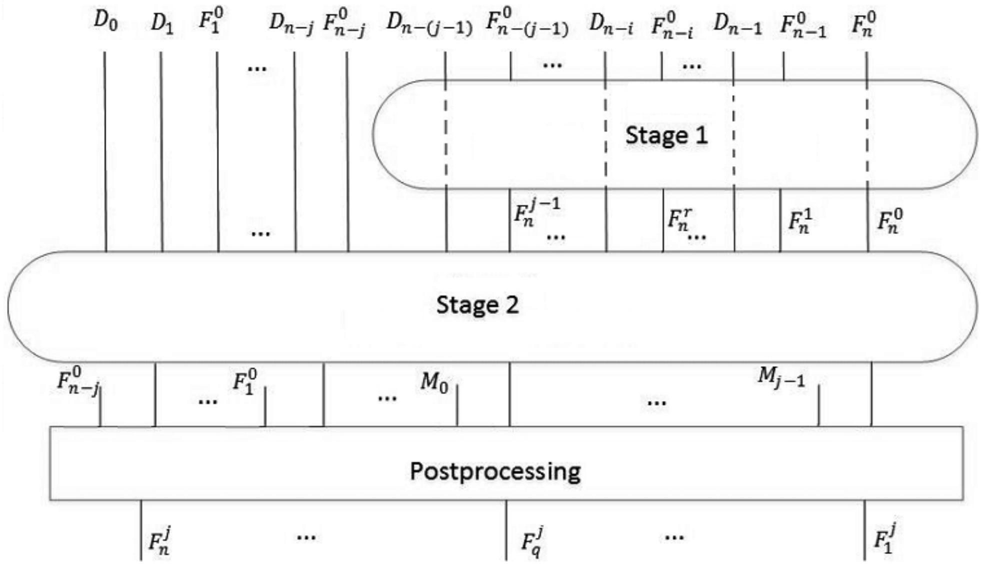

D_0 D_1 F_1^0 … D_{n-j} F_{n-j}^0 $D_{n-(j-1)}$ $F_{n-(j-1)}^0$ D_{n-i} F_{n-i}^0 D_{n-1} F_{n-1}^0 F_n^0
Stage 1
F_n^{j-1} … F_n^r … F_n^1 F_n^0
Stage 2
F_{n-j}^0 … F_1^0 … M_0 … M_{j-1}
Postprocessing
F_n^j … F_q^j … F_1^j

Fig. 6. The architecture of a PPT-based n-bit j-parallel programmable LFSR.

Now let us start decomposing Equation 1. This equation can be expanded as follows.

$$F_i^j = D_{i-1}F_n^{j-1} + F_{i-1}^{j-1} =$$

$$D_{i-1}F_n^{j-1} + D_{i-2}F_n^{j-2} + F_{i-2}^{j-2} =$$

$$D_{i-1}F_n^{j-1} + D_{i-2}F_n^{j-2} + D_{i-3}F_n^{j-3} + F_{i-3}^{j-3} =$$

$$\vdots$$

$$= \sum_{t=1}^{j} D_{i-t}F_n^{j-t} + F_{i-j}^{0}$$

$$i \in [1, n] \qquad (2)$$

Since D_is and F_i^0s are given, the sum in Equation 2 can be computed using a PPT including associative AX operations if the F_n^ks are known. The second PPT stage in Figure 6 is designed on the basis of this equation.

We can also calculate F_n^ks using Equation 1 as follows.

$$F_n^k = F_{n-\mathrm{k}}^0 + \sum_{t=1}^{\mathrm{k}} D_{n-t}F_n^{k-t}$$

$$k \in [1, j] \qquad (3)$$

The sum in Equation 3 can also be computed using another PPT, which constructs stage 1 of the architecture shown in Figure 6. Equations 2 and 3 form the jth STES of the programmable parallel LFSR of Figure 2. Equation 2 actually represents n equations for n different values of i. We refer to these equations as the jth stage 2 STES equations in this paper. Also Equation 3 represents j equations for j different values of k. The latter are referred to as the jth stage 1 STES equations in this paper.

THE DESIGN METHOD

In this section, we design a programmable 8-bit 5-parallel LFSR using the proposed method to illustrate how the method works.

The Design of the First Stage

According to Equations 3, the 5th stage 1 STES equations of a programmable parallel LFSR of degree 8 are as follows.

$$F_8^1 = F_7^0 + D_7F_8^0$$

$$F_8^2 = F_6^0 + D_7F_8^1 + D_6F_8^0$$

$$F_8^3 = F_5^0 + D_7F_8^2 + D_6F_8^1 + D_5F_8^0$$

$$F_8^4 = F_4^0 + D_7F_8^3 + D_6F_8^2 + D_5F_8^1 + D_4F_8^0$$

$$F_8^5 = F_3^0 + D_7F_8^4 + D_6F_8^3 + D_5F_8^2 + D_4F_8^1 + D_3F_8^0 \qquad (3)$$

Each of the above equations represents a LOO prefix processing problem, which can be parallelized using a PPT. Figure 7 shows the first stage of a programmable 8-bit 5-parallel LFSR designed without the use of any PPT. Figure 8 shows the same circuit in which every LOO prefix processing problem has been solved using a Brent-Kung PPT. This circuit has been designed as a sample by feeding the AX operation as a building block into the Brent-Kung topology.

The Design of the Second Stage

According to Equations 2, the 5th stage 2 STES equations of a programmable parallel LFSR of degree 8 are as follows.

$$F_1^5 = D_0F_8^4 + M_4$$

$$F_2^5 = D_1F_8^4 + D_0F_8^3 + M_3$$

$$F_3^5 = D_2F_8^4 + D_1F_8^3 + D_0F_8^2 + M_2$$

$$F_4^5 = D_3F_8^4 + D_2F_8^3 + D_1F_8^2 + D_0F_8^1 + M_1$$

$$F_5^5 = D_4F_8^4 + D_3F_8^3 + D_2F_8^2 + D_1F_8^1 + D_0F_8^0 + M_0$$

$$F_6^5 = D_5F_8^4 + D_4F_8^3 + D_3F_8^2 + D_2F_8^1 + D_1F_8^0 + F_1^0$$

$$F_7^5 = D_6F_8^4 + D_5F_8^3 + D_4F_8^2 + D_3F_8^1 + D_2F_8^0 + F_2^0$$

$$F_8^5 = D_7F_8^4 + D_6F_8^3 + D_5F_8^2 + D_4F_8^1 + D_3F_8^0 + F_2^0 \qquad (2)$$

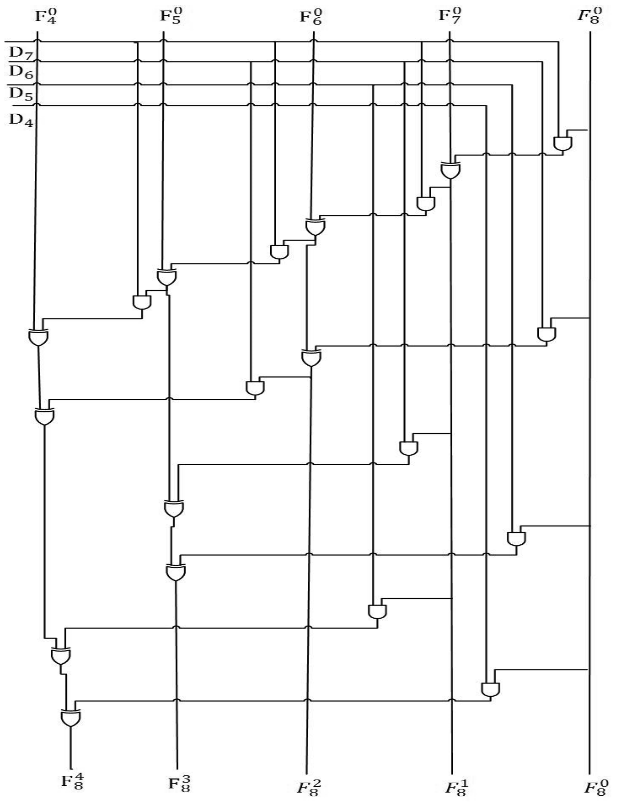


Fig. 7. The first stage of the programmable 8-bit 5-parallel programmable LFSR.

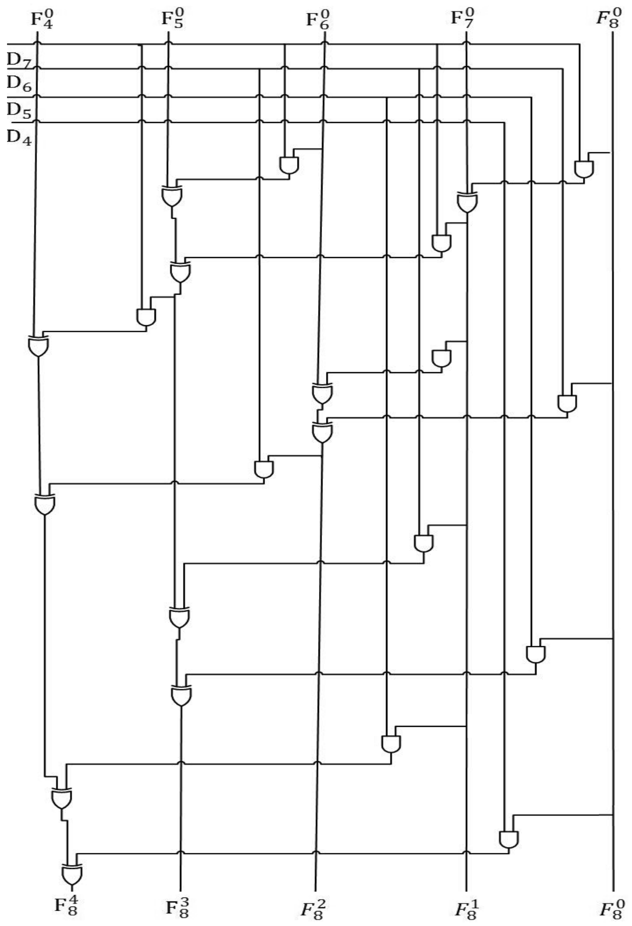


Fig. 8. The first stage of the programmable 8-bit 5-parallel programmable LFSR using Brent-Kung PPT.

Again, each of the above equations represents a LOO prefix processing problem, which can be parallelized using a PPT. Figure 9 shows a sample implementation for the second stage of the PPT-Based programmable 8-bit 5-parallel LFSR. Again, it has been constructed through applying the AX operation into the Brent-Kung topology.

It should be noted here that an independent PPT should be selected for parallelizing every individual LOO prefix processing problem in the STES. The selected PPTs can be mutually different or the same according to the design objectives. Thus, this approach gives the designer at least n + j options in the circuit-level design space. This is the main contribution of this paper. In Figure 8, we have selected the Bren-Kung PPT only to illustrate the design process.

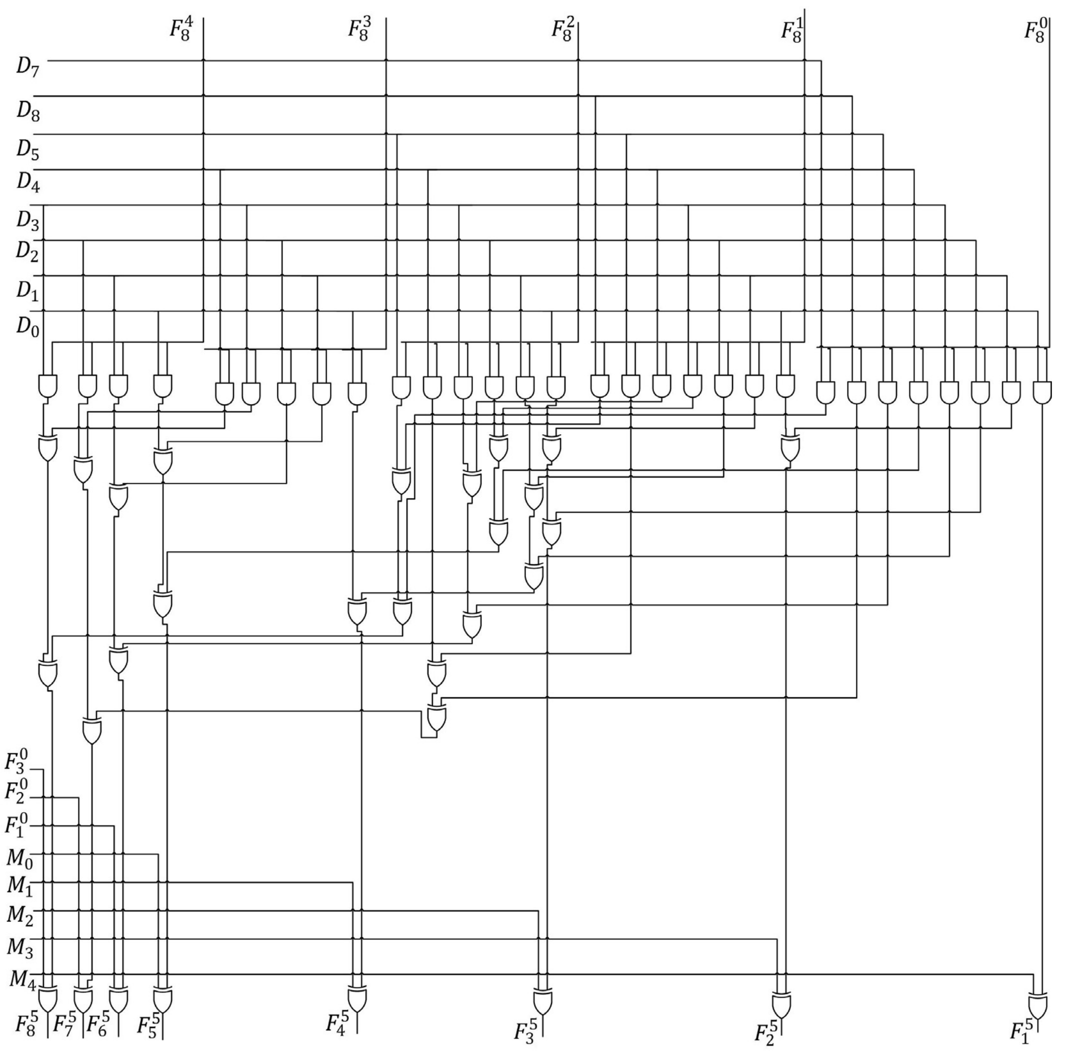


Fig. 9. The second stage for a programmable 8-bit 5-parallel LFSR.

The Design of the Post Processing Module

The post processing stage should convert $\sum_{t=1}^{k} D_{i-t}F_n^{q-t}$ to $F_n^k = F_{n-k}^0 + \sum_{t=1}^{k} D_{n-t}F_n^{k-t}$ for every $q \in [0, n-1]$. This stage does not depend on the generating sequence. In this stage, D_{i-t} should be replaced by 0 if $i - t$<0 and $F_{n-\min(k,j)}^0$ should be replaced by M_{k-n} if $n - k \leq 0$. The following figure shows the third stage for the 8-bit 5-parallel LFSR.

Pipelined Design

A PPT-based n-bit j-parallel programmable LFSR can be designed in a pipelined form. Let us assume that we can write j as the product of e and f ($j = e.f$). We can construct a PPT-based n-bit j-parallel programmable LFSR of e cascaded n-bit f-parallel programmable LFSRs with latches between them. A sample pipelined implementation is shown in the next section. In this case, our approach will give the designers $e.(n + f) = e.n + e.f$ options, which will obviously be more than $n + j = n + e.f$ assuming $e > 1$.

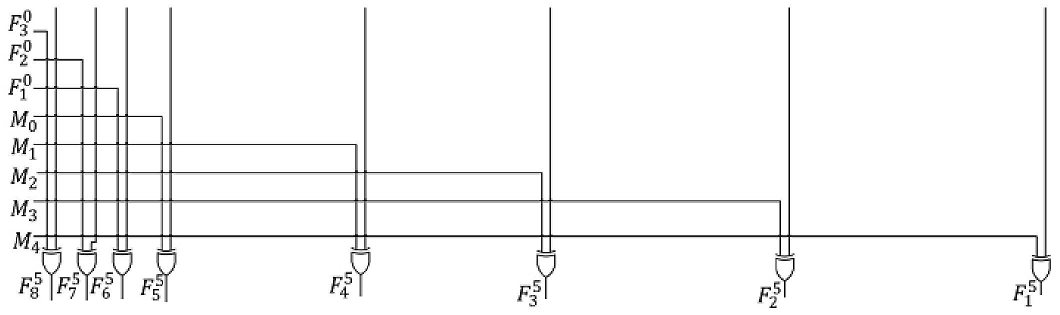


Fig. 10. The third stage for the 8-bit 5-parallel (programmable) LFSR.

EVALUATION

As mentioned in previous sections, PPT-based programmable parallel LFSR aims at allowing designers to maneuver in a large space of tradeoff points order to meet circuit level design objectives. Let us assume here that the design objective is to maximize the throughput. The reason for making such an assumption is that the state-of-the-art programmable parallel LFSRs have considered throughput as the main objective (Toal et al., 2009; Grymel et al., 2011). With such an objective, the main tradeoff in current research works is the tradeoff between the sampling rate and the clock frequency. The reason is that programmability and parallelism both increase the logical depth and consequently tend to reduce the clock frequency. It can be shown that the Brent-Kung topology minimizes the logical depth (Harris, 2003). Thus, we will use the Brent-Kung PPT to solve all LOO prefix processing problems in the design of our programmable parallel LFSR in this section. We will compare Brent-Kung programmable parallel LFSR with the art programmable parallel LFSRs presented in Toal et al. (2009) and Grymel et al. (2011) in terms of throughput, area, and power to highlight the tradeoffs.

The degree of the generating polynomial and the sampling rate are both equal to 32 similar to Toal et al. (2009) and Grymel et al. (2011). Pipelining has been applied to the design. The PPT-based 32-parallel Programmable LFSR has been designed by cascading 8 instances of a 4-parallel programmable LFSR. Figure 11 shows the pipelined architecture.

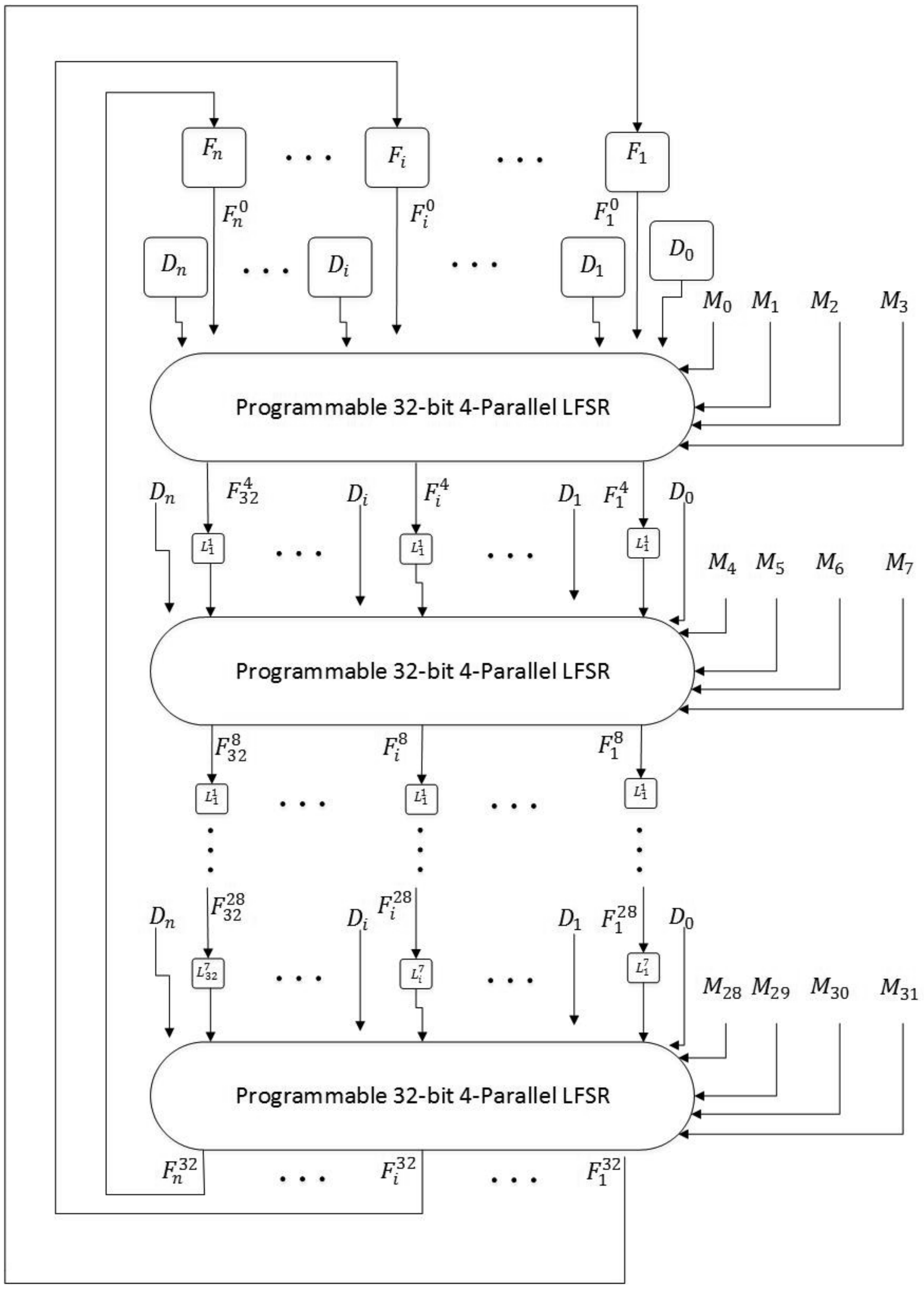

F_n
F_i
F_1
F_n^0
F_i^0
F_1^0
D_n
D_i
D_1
D_0
M_0 M_1 M_2 M_3
Programmable 32-bit 4-Parallel LFSR
F_{32}^4 F_i^4 F_1^4
L_1^1
M_4 M_5 M_6 M_7
F_{32}^8 F_i^8 F_1^8
F_{32}^{28} F_i^{28} F_1^{28}
L_{32}^7 L_i^7 L_1^7
M_{28} M_{29} M_{30} M_{31}
F_n^{32} F_i^{32} F_1^{32}

Fig. 11. Pipelined programmable 32-bit 32-parallel LFSR.

VHDL code has been used with ModelSim in the design phase and 130-nm TSMS technology has been used with Synopsys Design Compiler in the synthesis phase with the objective of overall optimization. In Figure 11, L_i^k is the ith latch temporarily storing $F_i^{4.k}$.

Table 1. Comparison.

	TR	GR	Proposed	Improvement TR	Improvement GR
Clock Frequency	154 MHZ	481 MHZ	592 MHz	284%	23%
Throughput	4.92 Gbps	15.38 Gbps	18.94Gbps	284%	23%
Total Power	12.21 mw	14.74 mw	18.42 mw	-51%	-25%
Total Area	0.150 mm^2	0.033 mm^2	0.024mm^2	72%	27%

In the above table, TR represents the programmable parallel LFSR designed in Toal et al. (2009) and GR represents the one designed in Toal et al. (2009). The above table shows a tangible improvement in the throughput (284% against TR and 23% against GR) and area (72% against TR and 27% against GR). But as shown in the table, this improvement is gained at the cost of increased power. This clarifies the reason why we have not used FPGAs in our design. FPGAs are widely used for rapid prototyping as well as a standalone product. However, they suffer from high power consumption due to their internal structure. As such, using an FPGA to measure the power consumption of a circuit is not a viable option and, therefore, is not common in the literature.

There is another point to consider regarding the tradeoff between area and power; reducing the area does not necessarily reduce the power. This is due to the fact that eliminating some circuit elements might increase the activity of others. The extra activity imposed on other resources might increase their dynamic power consumption. This is indeed the case when one moves from an ordinary serial LFSR to a parallel one.

CONCLUSIONS AND FURTHER WORKS

In this paper, a PPT-based architecture was presented for designing programmable parallel LFSRs using simple building blocks. This architecture allows designers to select among a variety of PPT topologies to maneuver in a large space of tradeoff points with respect to circuit level design objectives and constraints. We illustrated how it is possible to improve throughput and area using this approach at the cost of increased power. Presenting taxonomy of prefix-based programmable parallel LFSRs designed on the basis of different PPT topologies is suggested as further work. The work of this paper can also be continued by developing novel PPT topologies for improving parameters such as throughput and power in PPT-based programmable parallel LFSRs.

REFERENCES

Abdel-Hafeez, S., Gordon-Ross, A. & Parhami, B. 2013. Scalable digital CMOS comparator using a parallel prefix tree. IEEE Trans. Very Large Scale Integr. Syst. 21.

Acevedo, O. & Kagaris, D. 2016. On The Computation of LFSR Characteristic Polynomials for Built-In Deterministic Test Pattern Generation. IEEE Trans. Comput. 65(2).

Acevedo, O. & Kagaris, D. 2015. On the computation of LFSR characteristic Polynomials for built-in deterministic test pattern generation. IEEE Trans. Comput. 64.

Ahmad, A. & Al-Maashri, A. 2008. Investigating some special sequence lengths generated in an external exclusive-NOR type LFSR. J. Comput. Electr. Eng. 34.

Ayinala, M. & Parhi, K.K. 2011. High-speed parallel architectures for linear feedback shift registers. IEEE Trans. Signal Process. 59.

Chen, H. 2009. CRT-based high-speed parallel architecture for long BCH encoding. IEEE Trans. Circuits Syst. II Express Briefs. 56.

Cheng, C. & Parhi, K.K. 2006. High-speed parallel CRC implementation based on unfolding, pipelining, and retiming. IEEE Trans. Circuits Syst. II Express Briefs. 53.

Cheng, C. & Parhi, K.K. 2009. High speed VLSI architecture for general linear feedback shift register (LFSR) structures. Proc. 43rd Asilomar Conf. Signals, Syst. Comput.

Condo, C., Martina, C., Piccinini, M. & Masera, G. 2014. Variable parallelism cyclic redundancy check circuit for 3gpp-Lte/Lte-advanced. IEEE Signal Process. Lett. 21.

Delgado-Mohatar, O., Fúster-Sabater A. & Sierra J.M. 2011. Performance evaluation of highly efficient techniques for software implementation of LFSR. J. Comput. Electr. Eng. 37.

Dubrova, E. 2009. A transformation from the fibonacci to the galois nlfsrs. IEEE Trans. Inf. Theory 55(11).

Esposito, D., Caro, D.D. & Strollo, A.G.M. 2016. Variable Latency Speculative Parallel Prefix Adders for Unsigned and Signed Operands. IEEE Trans. Circuits Syst. I 63(8).

Gai, S., Lioy, A. & Neri, F. 1986. VLSI implementation of linear feedback shift registers for microprocessor applications. Microprocess. Microprogramming. 18.

Glaise, R.J. 1997. A two-step computation of cyclic redundancy code CRC-32 for ATM networks. IBM J. Res. Dev. 41.

Grymel, M. & Furber S.B. 2011. A novel programmable parallel CRC circuit. IEEE Trans. Very Large Scale Integr. Syst. 19.

Gurusamy, L., Kashif, M. & Julai, N. 2016. Design and Implementation of an Efficient Hybrid Parallel-Prefix Ling Adder Using 0.18micron CMOS Technology in Standard Cell Library. App. Mec. Mat. 833.

Harris, D. 2003. A taxonomy of parallel prefix networks. Proc. Thirty-Seventh Asilomar Conf. Signals, Syst. Comput.

Hepzibha, K.G. & Subha, C.P. 2016. A novel implementation of high speed modified brent kung carry select adder. 10th Int. Conf. Intell. Syst. Control.

Hobson, R.F. 2015. A framework for high-speed parallel-prefix adder performance evaluation and comparison. Int. J. Circuit Theory Appl. 43.

Hu, G., Sha, J. & Wang, Z. 2017. High-Speed Parallel LFSR Architectures Based on Improved State-Space Transformations. IEEE Trans. on Very Large Scale Integr. Syst. 25(3).

Huang, C.-H., Wang, J.-S. & Huang, Y.-C. 2002. Design of high-performance CMOS priority encoders and incrementer/decrementers using multilevel lookahead and multilevel folding techniques. IEEE J. Solid-State Circuits. 37.

Jaberipur, G. & Langroudi, S.H.F. 2015. (4+2logn) Δg parallel prefix modulo-(2^n-3) adder via double representation of residues in [0,2]. IEEE Trans. Circuits Syst. II Express Briefs. 62.

Kim, H., Lee, Y. & Kim, J-H. 2015. Low-complexity CRC-aided early stopping unit for parallel turbo decoder. Electron. Lett. 51(21).

Kumar, P.S., Sivakumar, N. & Rao, D.S. 2015. Design and implementation of hybrid parallel prefix adder. Int. J. Emerg. Eng. Res. Technol. 3.

Lama Shaer, L., Sakakini, T., Kanj, R. & Chehab, A. & Kayssi, A. 2016. A low power reconfigurable LFSR. 18th Mediterr. Electro. Conf.

Lee, M.T. & Su, C.M. 2014. Iterative image encryption based on the dyadic displacement and linear feedback shift register in discrete wavelet transform. Proc. Int. Symp. Comput. Consum. Control.

Li, C., Zeng, X., Li, C., Helleseth, T. & Li, M. 2016. Construction of de Bruijn Sequences From LFSRs With Reducible Characteristic Polynomials. IEEE Trans. Inf. Theory 62(1).

Li, M. & Lin, D. 2017. The Adjacency Graphs of LFSRs with Primitive-Like Characteristic Polynomials. IEEE Trans. Inf. Theory 63(2).

Lin, J.C., Chen, S.J. & Hu, Y.H. 2013. Cycle-efficient LFSR implementation on word-based microarchitecture. IEEE Trans. Comput. 62.

Lin, Y.-C. & Hung, L.-L. 2009. Straightforward construction of depth-size optimal, parallel prefix circuits with Fan-out 2. ACM

Trans. Des. Autom. Electron. Syst. 14.

Lin, Y.-C. & Hung, L.-L. 2009. Fast problem-size-independent parallel prefix circuits. J. Parallel Distrib. Comput. 69.

Lin, R., Nakano, K., Olariu, S. & Zomaya, Y. 2003. An efficient parallel prefix sums architecture with domino logic. IEEE Trans. Circuits Syst. 14.

Lina, Y.-C. & Sub, C.-Y. 2005. Faster optimal parallel prefix circuits: new algorithmic construction. J. Parallel Distrib. Comput 65.

Manikandan, S.K., Sharmitha, E.K., Angeline, M.N. & Palanisamy, C. 2013. High throughput LFSR design for BCH encoder using sample period reduction technique for MLC NAND based flash memories. Int. J. Comput. Appl. 66.

Mashhadi, S. & Dehkordi, M.H. 2015. Two verifiable multi secret sharing schemes based on nonhomogeneous linear recursion and LFSR public-key cryptosystem. J. Inf. Sci. 294.

Matsui, H. 2014. Lemma for linear feedback shift registers and DFTs applied to affine variety codes. EEE Trans. Inf. Theory 60.

Mishra, S., Tripathi, R.R. & Tripathi, D.Kr. 2016. Implementation of configurable linear feedback shift register in VHDL. Int. Conf. on Emerg. Trends in Elec. Electron. Sustain. Energy Syst.

Morales-Sandoval, M., Feregrino-Uribe C., Kitsos P. & Cumplido R. 2009. Area/performance trade-Off analysis of an FPGA digit-serial GF(2^M) Montgomery multiplier based on LFSR. J. Comput. Electr. Eng. 35.

Ouahab, Y., Rashidzadeh, R. & Muscedere, R. 2017. A secure scan chain using a phase locking system and a reconfigurable LFSR. IEEE 30th Can. Conf. Elec. and Comput. Eng.

Panda, A.S. & Moganti, G.L.K. 2014. A high-speed pipelined design for CRC-8 ATM-HEC circuit and a comparative study with its parallel and serial counterparts. Int. J. Adv. Res. Comput. Commun. Eng. 3.

Parhi, K. K. 2004. Eliminating the fanout bottleneck in parallel long BCH encoders. IEEE Trans. Circuits Syst. 51.

Pomeranz, I. 2016. Computing Seeds for LFSR-Based Test Generation from Nontest Cubes. IEEE Trans. Very Large Scale Integr. Syst. 24(6).

Pomeranz, I. 2017. LFSR-Based Generation of Multicycle Tests. IEEE Trans. on Comput.-Aided Design Integr. Circuits Syst. 36(3).

Rahimov, H. M. Babaei, M. & Farhadi, M. 2011. Cryptographic PRNG based on combination of LFSR and chaotic logistic map. J. Appl. Math. 2.

Ren., H. & Xiong, Z. 2015. A Multi-polynomial LFSR Based BIST Pattern Generator for Pseudorandom Testing. 2nd Int. Conf. Inf. Sci. Control Eng.

Santos, E.E. 2002. Optimal and efficient algorithms for summing and prefix summing on parallel machines. J. Parallel Distrib. Comput. 62.

Savic, N., Stojcev, M., Nicolic, T., Petrovic, V. & Jovanovic, G. 2014. Reconfigurable low power architecture for fault tolerant pseudo-random number generation. J. Circuits, Syst. Comput. 23.

Sergeev, I.S. 2013. On the complexity of parallel refix circuits. Proc. Electron. Colloq. Comput. Complex.

Singh, S., Sujana, S., Babu, I. & Latha, K. 2013. VLSI implementation of parallel CRC using pipelining, unfolding and retiming. IOSR J. VLSI Signal Process. 2.

Toal, C., McLaughlin, K., Sezer, S. & Yang, X. 2009. Design and implementation of a field programmable CRC circuit architecture. IEEE Trans. Very Large Scale Integr. Syst. 17.

Toal, C., McLaughlin, K., Sezer, S. & Yang, X. 2009. Design and implementation of a field programmable CRC circuit architecture. IEEE Trans. Very Large Scale Integr. Syst. 17: 1142-1147.

Ugurdag, H.F. & Baskirt, O. 2012. Fast parallel prefix logic circuits for n2n round-robin arbitration. Microelectronics J. 43.

Upadhyay, D., Shah, T. & Sharma, P. 2015. Cryptanalysis of hardware based stream ciphers and implementation of GSM stream cipher to propose a novel approach for designing n-bit LFSR stream cipher. 19th Int. Symp. VLSI Design. Test.

Wang, Q. & Tan, C.H. 2016. Proof of a conjecture and a bound on the imbalance properties of LFSR subsequences. Discrete

Appl. Math. 211.

Wei, L. & Yang, X. 2015. A Parallel and Reconfigurable United Architecture for Fibonacci and Galois LFSR. 7th Int. Conf. Intelligent Human-Machine Systems and Cybernetics.

Wu, Y. 2015. New Scalable Decoder Architectures for Reed–Solomon Codes. IEEE Trans. Commun. 63(8).

Xiang, D., Wen, X. & Wang, L-T. 2017. Low-Power Scan-Based Built-In Self-Test Based on Weighted Pseudorandom Test Pattern Generation and Reseeding. IEEE Trans. Very Large Scale Integr. Syst. 25 (3).

Yasodharan, S. & Swamynathan, S.M. 2014. Low power test pattern generation in BIST schemes. Int. J. Eng. Res. Gen. Sci. 2.

Ying, J-C., Tseng, W-D. & Tsai, W-J. 2018. Asymmetry dual-LFSR reseeding for low power BIST. Integration, the VLSI Journal 60.

Zarandi, A.A.E., Molahosseini, A.S., Hosseinzadeh, M., Sorouri, S., Antão, A. & Sousa, L. 2014. Reverse converter design via parallel-prefix adders: novel components, methodology and Implementations. IEEE Trans. Very Large Scale Integr. Syst. 23.

Zhang, X. & Parhi, K. K. 2005. High-speed architectures for parallel long BCH encoders. IEEE Trans. Very Large Scale Integr. Syst. 13.

Zhu, H., Cheng, C.-k. & Graham, R. 2006. Constructing zero-deficiency parallel prefix circuits of minimum depth. ACM Trans. Des. Autom. Electron. Syst. 11.

Zibin, D., Longmei, N., Xuan Y. & Xiaonan L. 2013. Design and implementation of configurable LFSR instructions targeted at stream ciphers. J. Circuits, Syst. Comput. 22.

Submitted: 29/08/2017
Revised: 08/01/2018
Accepted: 11/04/2018

تصميم LFSR متوازية قابلة للبرمجة باستخدام PPTs

بهروز زولفاغاري، مهدي سيديغي ومهران فلاح
قسم هندسة الكمبيوتر وتكنولوجيا المعلومات، جامعة أميركابير للتكنولوجيا، طهران، إيران

الخلاصة

تتأثر انتاجية سجلات التغذية الخطية المرتجعة (LFSR) بمعدل اختيار العينات بالإضافة إلى معدل التزامن. من ناحية أخرى، يتم تحديد الخصائص على مستوى النظام من LFSR، مثل: القدرة على اكتشاف الأخطاء؛ بواسطة توليد متعددة الحدود. تهدف LFSRs المتوازية إلى تحسين معدل اختيار العينات من أجل تلبية متطلبات الإنتاجية العالية في بيئات الإرسال أو الحوسبة المتوازية. علاوة على ذلك، توفر LFSRs القابلة للبرمجة المزيد من المرونة على مستوى النظام من خلال السماح باستخدام متعددات الحدود المختلفة. وبالتالي، فإن استخدام LFSRs المتوازية القابلة للبرمجة يبدو حلاً جذاباً لتحسين كل من معلمات الإنتاجية ومستوى النظام. وقد تكون LFSRs المتوازية القابلة للبرمجة مفيدة في التشفير الانسيابي، والمعالجات الدقيقة والعديد من البيئات الأخرى. ولكن التوازي وقابلية البرمجة يمكن أن يعملا على التقليل من معدل التزامن عن طريق زيادة العمق المنطقي وزيادة الطاقة والمساحة عن طريق زيادة عدد البوابات. وبالتالي، سنحتاج إلى حل فعال لإدارة المقارنات. يقترح هذا البحث نهج يرتكز على أشجار بادئة متوازية (PPTs) لتصميم LFSRs المتوازية القابلة للبرمجة. PPTs هي مجموعة من الطبولوجيا المُستخدمة سابقاً في تصميم الدوائر الحسابية المتوازية من أجل إدارة المقارنات بين المعلمات المختلفة لمستوى الدائرة. وهذا النهج يسمح للمصممين باستخدام أنواع PPTs مختلفة لتحسين مختلف المعلمات لمستوى الدائرة. تم تصميم وتقييم نموذج LFR متوازي قابل للبرمجة يرتكز على PPT. وأظهرت النتائج التجريبية تحسن في الإنتاجية بنسبة تزيد عن 23 ٪ وكذلك تحسن بنسبة تزيد عن 27 ٪ في المساحة مقارنةً بهندسة LFSR المتوازية المبرمجة الحالية.