This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2456294, IEEE Transactions on Circuits and Systems II: Express Briefs

Efficient Parallel Architecture for Linear Feedback Shift Registers

Jaehwan Jung, *Student Member, IEEE*, Hoyoung Yoo, *Student Member, IEEE*, Youngjoo Lee, *Member, IEEE*, and In-Cheol Park, *Senior Member, IEEE*

***Abstract*— This brief presents a new parallel architecture for linear feedback shift registers (LFSRs), which can be used to achieve high-throughput BCH or CRC encoders for storage and communication systems. While previous parallel LFSR architectures have computed values by using the past input messages and the register outputs, the proposed parallel architecture based on the transposed serial LFSR calculates the output by using only the past feedback values. As a result, the proposed architecture reduces the area-time product by up to 59% compared to the recent architecture.**

***Index Terms*— BCH encoder, CRC encoder, critical path, linear feedback shift register (LFSR), parallel architecture.**

I. Introduction

LINEAR feedback shift registers (LFSRs) are conventionally used to compute a remainder polynomial by dividing a message polynomial by a generator polynomial, and widely employed in implementing BCH and CRC encoders [1], [2]. As the technology advances, the encoding throughput required in digital communication and storage systems increases accordingly. Since a serial LFSR dealing with a message bit at a time is not sufficient for such a high throughput, parallel architectures have been adopted in many BCH and CRC encoders.

The conventional method [3] to derive a parallel LFSR from a serial LFSR is shown in Fig. 1, where a simple LFSR corresponding to a generator polynomial $g(x) = 1 + x^2 + x^3$ is exemplified. To make a 3-parallel architecture, the combinational logic part of the serial LFSR is concatenated three times as shown in Fig. 1(c). The cascaded structure makes the combinational logic network long, and incurs significant performance degradation due to the long critical path resulting from the serial concatenation. Therefore, the conventional approach is limited to low-performance applications.

Manuscript received February 17, 2015; revised May 18, 2015; accepted June 26, 2015. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2010-0028680); by SK Hynix; by the IC Design Education Center (IDEC).

J. Jung, H. Yoo, and I.-C. Park are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, Republic of Korea (e-mail: jaehwan@kaist.ac.kr; hyyoo@ics.kaist.ac.kr; icpark@kaist.edu).

Y. Lee is with the Department of Electronic Engineering, College of Electronics and Information Engineering, Kwangwoon University, Seoul, 139-701, Republic of Korea (e-mail: yjlee@kw.ac.kr).

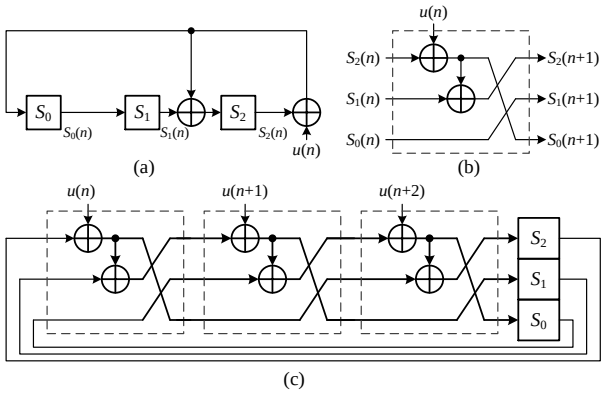

Fig. 1. (a) Serial LFSR for $g(x) = 1 + x^2 + x^3$. (b) The combinational logic part of the serial LFSR. (c) 3-parallel LFSR that concatenates (b) 3 times.

To alleviate the increase of critical path delay, many methods have been presented for LFSR parallelization [4]–[9]. The cascaded combinational logic part was optimized by adopting tree-structured computation [5] and by sharing sub-expressions [6]. Though the optimization is quite effective in reducing the hardware complexity, it is not sufficient for reducing the critical path delay. A partially pipelined architecture was proposed in [7] to overcome the limitation, which is to conduct a state-space transformation to the conventional parallel LFSR circuit. As a result, the calculation outside the feedback loop can be pipelined. However, this approach increases the overall hardware complexity, as additional hardware components are necessary for the transformation.

Based on the IIR filter topology, other approaches [8], [9] have been presented to derive a new architecture equivalent to the serial LFSR. As the input part is not associated with the feedback loop in the new architecture, that part can be pipelined with ease, enabling a high-speed decoder without much increasing the hardware complexity. All the previous architectures resort to the conventional LFSR architecture calculating the redundant intermediate values and thus have a limitation in reducing the critical path delay and the hardware cost.

To eliminate redundant calculations and registers, this brief proposes a novel parallel LFSR architecture based on the transposed LFSR. The proposed architecture stores the feedback values only, whereas the previous architectures [7]–[9] need to store both the past inputs and outputs. Compared to the previous works [7]–[9], the proposed architecture reduces the number of registers by half if the pipelining registers are not counted, and requires a less number of XOR gates, leading to a shortened critical path as well.

Although the proposed technique is explained for the encoding of cyclic codes, it can be applied to similar

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2456294, IEEE Transactions on Circuits and Systems II: Express Briefs

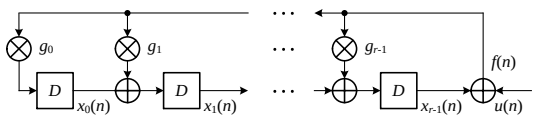

Fig. 2. General serial LFSR architecture.

applications such as LFSR-based pseudo-random number generators widely used for VLSI testing [10], [11] and stream cryptography [12].

The rest of this brief is organized as follows. Section II summarizes the previous parallel LFSR architectures and Section III describes the proposed architecture in detail. Experimental results are explained and analyzed in Section IV, and conclusions are made in Section V.

II. Previous Parallel Architectures

As illustrated in Fig. 1, the conventional parallel method replicates the combinational logic part p times and concatenates them in serial, where p called the parallel factor denotes the number of message bits to be processed at a time [3]. If p is not small, the cascaded structure makes the combinational logic network extremely long, and leads to a significant degradation of performance.

A. State-Space Transformed Architecture

To reduce the critical path delay of the conventional parallelization, a state-space transformed architecture was proposed in [7]. The general LFSR circuit that computes the remainder polynomial by dividing the message polynomial by the r-th order generator polynomial $g(x) = g_0 + g_1x + g_2x^2 + \cdots + g_{r-1}x^{r-1} + g_rx^r$ is depicted in Fig. 2, where $u(n)$ and $x_{r-1}(n)$ are the input message bit and the output of the LFSR at time n, respectively. In the state-space vector representation, the serial LFSR is denoted as

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{b}\mathbf{u}(n), \tag{1}$$

where $\mathbf{x}(n) = \begin{bmatrix} x_0(n) & x_1(n) & \cdots & x_{r-1}(n) \end{bmatrix}^T$ is the state value at time n, and $\mathbf{u}(n)$ is the vector representation of the message bit at time n. The $r \times r$ matrix $\mathbf{A}$ is

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -g_0 \\ 1 & 0 & 0 & \cdots & 0 & -g_1 \\ 0 & 1 & 0 & \cdots & 0 & -g_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -g_{r-1} \end{bmatrix} \tag{2}$$

and the $r \times 1$ matrix $\mathbf{b}$ is

$$\mathbf{b} = \begin{bmatrix} -g_0 & -g_1 & \cdots & -g_{r-1} \end{bmatrix}^T. \tag{3}$$

The state-space representation is depicted in Fig. 3(a). For a p-parallel system, (1) can be rewritten as

$$\mathbf{x}(mp+p) = \mathbf{A}^p\mathbf{x}(mp) + \mathbf{B}_p\mathbf{u}_p(mp), \tag{4}$$

where $m = 0, 1, \cdots, \lceil K/p \rceil - 1$ if the message length is K, and $\mathbf{B}_p$ is a $r \times p$ matrix specified as

$$\mathbf{B}_p = \begin{bmatrix} \mathbf{b} & \mathbf{A}\mathbf{b} & \mathbf{A}^2\mathbf{b} & \cdots & \mathbf{A}^{p-1}\mathbf{b} \end{bmatrix}. \tag{5}$$

The p-parallel architecture shown in Fig. 3(b) induces a large hardware overhead caused by $\mathbf{A}^p$. Applying a proper linear transformation can alleviate the hardware overhead. The explicit output $\mathbf{y}$ is defined as $\mathbf{y}(mp) = \mathbf{C}_p\mathbf{x}(mp)$, where $\mathbf{C}_p$ is a $r \times r$ identity matrix, and the linear transformation is defined as

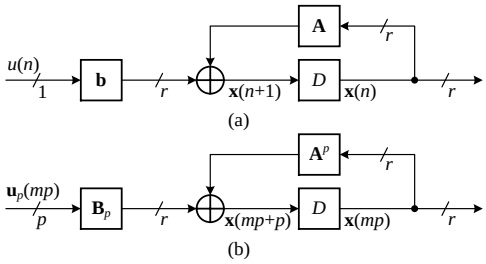

Fig. 3. (a) Serial LFSR architecture and (b) p-parallel LFSR architecture.

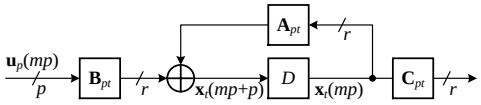

Fig. 4. State-space transformed p-parallel architecture.

$$\mathbf{x}(mp) = \mathbf{T}\mathbf{x}_t(mp), \tag{6}$$

where $\mathbf{T}$ is a non-singular constant matrix. By the transformation, (4) can be rewritten as

$$\begin{aligned} \mathbf{x}_t(mp+p) &= \mathbf{A}_{pt}\mathbf{x}_t(mp) + \mathbf{B}_{pt}\mathbf{u}_p(mp), \\ \mathbf{y}(mp) &= \mathbf{C}_{pt}\mathbf{x}_t(mp), \end{aligned} \tag{7}$$

where $\mathbf{A}_{pt} = \mathbf{T}^{-1}\mathbf{A}^p\mathbf{T}$, $\mathbf{B}_{pt} = \mathbf{T}^{-1}\mathbf{B}_p$, and $\mathbf{C}_{pt} = \mathbf{T}$.

The transformed parallel LFSR architecture is illustrated in Fig. 4. As the hardware complexity of the $\mathbf{A}_{pt}$ multiplication is similar to that of the $\mathbf{A}$ multiplication, the feedback loop of the p-parallel architecture has almost the same hardware cost as the conventional serial architecture. The matrix multiplications of $\mathbf{B}_{pt}$ and $\mathbf{C}_{pt}$, which are outside the feedback loop, can be pipelined to decrease the critical path delay. However, substantial matrix computations and corresponding pipelining registers require so many hardware elements that it is not easy to apply the approach to practical encoders.

B. Parallel LFSR Based on IIR Filter Topology

To mitigate the impractical hardware complexity of the parallel architecture [7], a new transformation based on the IIR filter topology [9] was applied. The past input and output values are combined to generate the output $y(n)$. The conventional serial encoder depicted in Fig. 5 can be represented as follows;

$$y(n) = g_{r-1} \cdot f(n-1) + g_{r-2} \cdot f(n-2) + \cdots + g_0 \cdot f(n-r) \tag{8}$$

$$f(n) = u(n) + y(n). \tag{9}$$

While the message sequence is entering, the codeword $c(n)$ is generated by selecting $u(n)$. After the message bits are completely entered, the multiplexer is switched to $y(n)$ to extract the internal values of the registers. As indicated in (8), $y(n)$ is calculated internally when the input message are bypassed. By substituting (9) into (8), $y(n)$ is rewritten as

$$\begin{aligned} y(n) &= g_{r-1} \cdot \left[y(n-1) + u(n-1) \right] + g_{r-2} \cdot \left[y(n-2) + u(n-2) \right] \\ &\quad + \cdots + g_0 \cdot \left[y(n-r) + u(n-r) \right] \\ &= \left[g_{r-1} \cdot y(n-1) + g_{r-2} \cdot y(n-2) + \cdots g_0 \cdot y(n-r) \right] \\ &\quad + \left[g_{r-1} \cdot u(n-1) + g_{r-2} \cdot u(n-2) + \cdots g_0 \cdot u(n-r) \right]. \end{aligned} \tag{10}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2456294, IEEE Transactions on Circuits and Systems II: Express Briefs

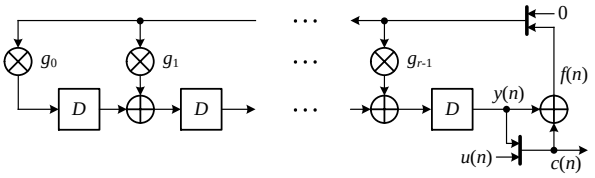

Fig. 5. The conventional serial encoder architecture.

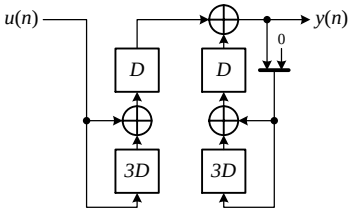

Fig. 6. LFSR architecture based on IIR filter topology for $g(x) = 1 + x^3 + x^4$.

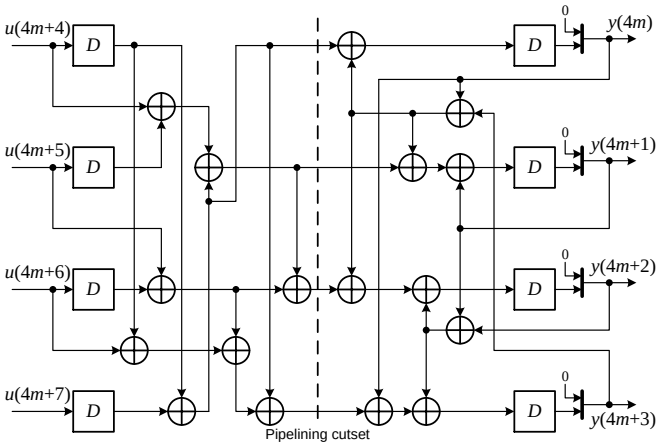

Fig. 7. 4-parallel architecture for $g(x) = 1 + x^3 + x^4$ [9].

As a result, the LFSR output can be composed of two parts, a part depending on the inputs and the other part depending on the past outputs. Since the transformed structure has no feedback loop in the input-dependent part, the pipelining technique can be adopted between the input-dependent part and the output-dependent part. In addition, the look-ahead technique [13] can be applied to achieve a parallel structure. For the sake of understanding, a transformed LFSR and its 4-parallel architecture corresponding to $g(x) = 1 + x^3 + x^4$ is exemplified in Fig. 6 and Fig. 7, respectively. The feedback loop in the input-dependent part is dismantled by using additional registers and calculations. This approach is effective in reducing the critical path delay, but still necessitates the higher hardware complexity than the conventional parallel method [3]–[6].

III. Proposed Architecture

While the previous architectures directly parallelize the conventional serial encoder shown in Fig. 5, a new parallel architecture based on the unaccustomed serial LFSR is proposed in the section to effectively reduce both the hardware complexity and the critical path delay. According to (8), the output of the systematic encoder, $y(n)$, is composed of the past feedback values, $f(n)$. Therefore, it is inefficient to store non-feedback values in the parallel architecture. Since the registers of the conventional serial LFSR are used to store incomplete outputs instead of the feedback values, the straightforward parallelization based on the conventional serial LFSR may lead to redundant elements and calculations. For instance,

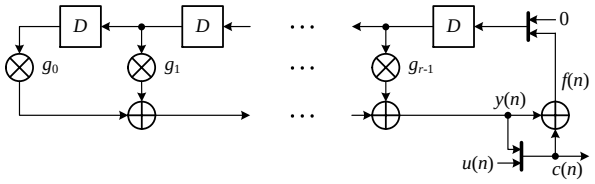

Fig. 8. Transposed serial encoder for $g(x)$.

TABLE I
Outputs of the Conventional and Transposed LFSRs Corresponding to $g(x) = 1 + x^3 + x^4$

Clock (n)	$u(n)$	$y(n)$	
		Conventional	Proposed
0	1	0	0
1	0	1	1
2	0	1	1
3	1	1	1
4	0	1	1
5	1	0	0
6	1	0	0
7	0	1	1
8	0	0	0
9	0	1	1
10	0	1	1
11	0	1	1

the parallel architecture in [9] generates the outputs by using the past inputs and outputs stored in registers.

A. Encoding Architecture Based on the Transposed LFSR

To eliminate unnecessary registers and XOR gates in parallelization, the serial LFSR architecture which stores only the past feedback values is employed. The proposed serial systematic encoder is described in Fig. 8, which is in fact a transposed architecture of the conventional systematic encoder depicted in Fig. 5. The two multiplexers are controlled to select $u(n)$ and $y(n)$ when the message bits are entering into the LFSR. After the message sequence is completely entered, the multiplexers are switched to $y(n)$ and zero to extract the systematic parities generated by the feedback values stored in the state registers. Since the proposed encoder is a transposed architecture, (8) and (9) are still valid for the proposed serial LFSR. By substituting (8) into (9), we have

$$f(n) = g_{r-1} \cdot f(n-1) + g_{r-2} \cdot f(n-2) + \cdots + g_0 \cdot f(n-r) + u(n). \tag{11}$$

According to (11), we can calculate $f(n)$ from the values of past $f(n)$ and the exact $y(n)$ can be calculated by substituting the values obtained from (11) into (8).

Prior to the parallelization of the serial LFSR, the operation of the proposed serial encoder is verified. An LFSR associated with $g(x) = 1 + x^3 + x^4$ is depict in Fig. 9. The output $y(n)$ of the conventional LFSR and that of the proposed one should be equal to each other. Table I shows how the values of $y(n)$ change according to the progress of the cycle, where it is apparent that the output of the proposed architecture is exactly equal to that of the conventional one.

B. Proposed Parallel Architecture

Consider the parallel architecture based on the transposed serial LFSR. For the sake of clear explanation, a 4-parallel architecture with $g(x) = 1 + x^3 + x^4$ is obtained by applying the look-ahead scheme [13] to (11). For the LFSR of Fig. 9,

$$y(n) = f(n-1) + f(n-4) \tag{12}$$

$$f(n) = f(n-1) + f(n-4) + u(n). \tag{13}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2456294, IEEE Transactions on Circuits and Systems II: Express Briefs

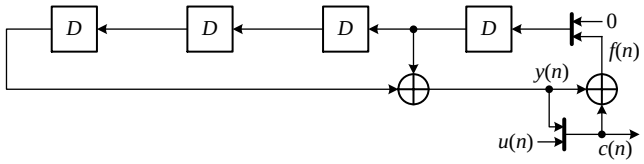

Fig. 9. The proposed encoder for $g(x) = 1 + x^3 + x^4$.

Adopting the look-ahead technique, $f(n–1)$ can be calculated. Replacing n with $n–1$ in (13), we have

$$f(n-1) = f(n-2) + f(n-5) + u(n-1). \quad (14)$$

Substitute it into (13),

$$f(n) = f(n-2) + f(n-4) + f(n-5) + u(n) + u(n-1). \quad (15)$$

By repeating the process, $f(n)$ can be expressed as

$$f(n) = f(n-3) + f(n-4) + f(n-5) + f(n-6) + u(n) + u(n-1) + u(n-2) \quad (16)$$

$$f(n) = f(n-5) + f(n-6) + f(n-7) + u(n) + u(n-1) + u(n-2) + u(n-3). \quad (17)$$

By substituting $4m$+4, $4m$+5, $4m$+6, and $4m$+7, where m is an integer, into n in (13), (15), (16), and (17), we obtain the following relations,

$$f(4m+4) = f(4m+3) + f(4m) + u(4m+4) \quad (18)$$

$$f(4m+5) = f(4m+3) + f(4m+1) + f(4m) + u(4m+5) + u(4m+4) \quad (19)$$

$$f(4m+6) = f(4m+3) + f(4m+2) + f(4m+1) + f(4m) + u(4m+6) + u(4m+5) + u(4m+4) \quad (20)$$

$$f(4m+7) = f(4m+2) + f(4m+1) + f(4m) + u(4m+7) + u(4m+6) + u(4m+5) + u(4m+4). \quad (21)$$

Let $u(4m+4)$, $u(4m+5)$, $u(4m+6)$, and $u(4m+7)$ be inputs at the current time, the relations in (18), (19), (20), and (21) include four past feedback values but no past inputs. Therefore, the proposed parallel architecture needs to store only past $f(n)$ values. Similarly, by substituting $4m$+4, $4m$+5, $4m$+6, and $4m$+7 into n in (12), we have

$$y(4m+4) = f(4m+3) + f(4m) \quad (22)$$

$$y(4m+5) = f(4m+4) + f(4m+1) \quad (23)$$

$$y(4m+6) = f(4m+5) + f(4m+2) \quad (24)$$

$$y(4m+7) = f(4m+6) + f(4m+3). \quad (25)$$

The feedback values after $f(4m+3)$ are zero if the user message ends at $u(4m+3)$. Hence, (22), (23), (24), and (25) can be shortened to

$$y(4m+4) = f(4m+3) + f(4m) \quad (26)$$

$$y(4m+5) = f(4m+1) \quad (27)$$

$$y(4m+6) = f(4m+2) \quad (28)$$

$$y(4m+7) = f(4m+3). \quad (29)$$

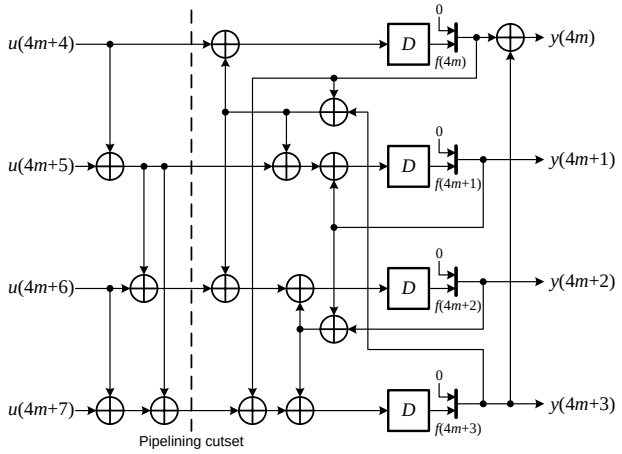

Fig. 10. The proposed 4-parallel architecture for $g(x) = 1 + x^3 + x^4$.

The systematic encoder only needs the remainder as parity bits, so we choose (26)–(29) instead of (22)–(25) as the output calculation logic. A 4-parallel architecture implementing (18)–(21), and (26)–(29) is shown in Fig. 10, where the feedback-dependent part realizes (18)–(21), and the output calculation part corresponds to (26)–(29). In the 4-parallel architecture, each register can be regarded as a block delay of four cycles [9]. Therefore, the corresponding past feedback values can be generated by updating look-ahead registers and are used for output computations. To generate the systematic parities correctly, the multiplexers and the inputs should be switched to zero in the parallel architecture. Table II shows the output values of the 4-parallel architecture resulting from the same input bits that are used in Table I. The table shows that the remainder of the proposed 4-parallel encoder is exactly equal to that of the serial encoder.

C. Analysis on the Proposed Parallel Architecture

Let N be the code length and p be the parallel factor. The proposed parallel architecture takes N/p cycles in the case. Compared to the architecture of [9] in Fig. 7, the proposed architecture completely eliminates the registers used to store the past input values and their related logic circuitry at the cost of a small logic computing the parity outputs. In the proposed architecture, the registers holding the past inputs are eliminated and the substantial calculations depending on the inputs are effectively distributed to two small combinational logics.

The critical path delay in Fig. 10 is $4 \cdot \tau_{XOR}$, where τ_{XOR} represents the delay of a 2-input XOR gate. Since the proposed architecture has no feedback loop associated with the input values, it can be pipelined like in [9]. The circuit shown in Fig. 10 is pipelined along the dotted line, which reduces the critical path to $3 \cdot \tau_{XOR}$. In addition, the circuit can be optimized by applying the tree-structured computation [5] and subexpression sharing [6].

In general, the critical path delay of the proposed architecture is bounded by $\left(\lceil \log_2 p \rceil + \left\lceil \log_2 \left(p \cdot \sum_{i=0}^{r-1} g_i + 1 \right) \right\rceil\right) \cdot \tau_{XOR}$, if the pipelining technique is not applied. Since the number of XOR

TABLE II

INPUTS AND OUTPUTS OF THE PROPOSED 4-PARALLEL ARCHITECTURE FOR $g(x) = 1 + x^3 + x^4$

Clock (m)	$u(4m)$	$u(4m+1)$	$u(4m+2)$	$u(4m+3)$	$y(4m)$	$y(4m+1)$	$y(4m+2)$	$y(4m+3)$
0	1	0	0	1	0	0	0	0
1	0	1	1	0	1	1	1	0
2	0	0	0	0	0	1	1	1

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2015.2456294, IEEE Transactions on Circuits and Systems II: Express Briefs

operations between input values cannot exceed the parallel factor p, the critical path delay of the input network is bounded by $\lceil \log_2 p \rceil \cdot \tau_{XOR}$ with the tree-structured optimization [5]. The maximum number of computational logic gates of the feedback-dependent part is $p \cdot \sum_{i=0}^{r-1} g_i + 1$, since the LFSR has $\sum_{i=0}^{r-1} g_i$ feedback updates at maximum and the look-ahead scheme is applied to the p-parallel structure. To connect the input- and feedback-dependent part, an additional gate is added. The critical path delay of the $y(n)$ calculation part is bounded by $\left\lceil \log_2 \left(\sum_{i=0}^{r-1} g_i \right) \right\rceil \cdot \tau_{XOR}$, which is much smaller than the $f(n)$ calculation logic. The critical path delay can be reduced to $\left\lceil \log_2 \left(p \cdot \sum_{i=0}^{r-1} g_i + 1 \right) \right\rceil \cdot \tau_{XOR}$ by inserting pipeline registers in front of the feedback-dependent part.

In addition, the proposed method is also applicable to non-binary LFSRs, since operations over binary-extension fields are easily obtained from the straightforward extension of binary operations [14].

IV. Experimental Results

For various CRC and BCH codes quoted from [9], the proposed architecture is compared to previous parallel architectures. The same codes are used for fair comparison, and the results are summarized in Table III. The parallel factor p is also specified in the table, and the whole message bits are fed at a time. The area-time product (AT) [9] is used as a figure of merit, which is defined as

$$AT = (1.5 \cdot nDE + nXOR) \cdot CPD, \quad (30)$$

where nDE, $nXOR$ and CPD denote the number of delay elements, the number of XOR gates, and the critical path delay, respectively. The ratio of a XOR gate to a delay element comes from the fact that the complexity of a D flip-flop corresponds to six 2-input NAND gates and that of a XOR gate to four NAND gates. The lower figure of merit implies the better performance, and the architecture of [7] is configured to have the minimum area-time product. The numbers in parentheses are relative values normalized by the AT resulting from the proposed method.

Experimental results show that the proposed parallel architecture considerably reduces the hardware cost compared to [9]. The critical path delay is also reduced by eliminating the look-ahead calculation of input values being used in [9]. As a result, the proposed architecture reduces the area-time product by up to 59% compared to the recently reported architecture. Therefore, the proposed method is effective in reducing the critical path delay as well as the hardware complexity.

V. Conclusion

This brief has presented a new parallel LFSR architecture that eliminates the redundant calculations from previous architectures. The proposed architecture is based on the transposed serial LFSR which stores only the feedback value. Therefore, the proposed architecture generates outputs by using the stored feedback values, while the recently reported architecture [9] stores both the past inputs and outputs. As a result, the proposed architecture is effective in achieving a high-speed, low-complexity block for parallel BCH or CRC encoding. Experimental results show that the proposed architecture improves the area-time product by up to 59% compared to the recent architecture.

TABLE III
Hardware Complexity and Critical Path Delay

Code (p)	Algorithm	$nXOR$	nDE	CPD	AT
CRC-12 (12)	[4]	52	12	10	700 (1.01)
	[7]	113	35	5	828 (1.19)
	[9]	109	36	5	815 (1.17)
	Proposed	103	24	4	695 (1.00)
CRC-16 (16)	[4]	72	16	15	1440 (2.03)
	[7]	187	48	5	1295 (1.82)
	[9]	113	50	5	940 (1.32)
	Proposed	94	32	5	710 (1.00)
SDLC (16)	[4]	88	16	8	896 (2.06)
	[7]	207	48	5	1395 (3.21)
	[9]	139	50	5	1070 (2.46)
	Proposed	97	32	3	435 (1.00)
CRC-16 Reverse (16)	[4]	154	16	15	2670 (5.13)
	[7]	219	46	5	1440 (2.77)
	[9]	126	50	5	1005 (1.93)
	Proposed	82	32	4	520 (1.00)
SDLC Reverse (16)	[4]	84	16	8	864 (1.57)
	[7]	217	48	5	1445 (2.62)
	[9]	127	50	5	1010 (1.83)
	Proposed	90	32	4	552 (1.00)
CRC-32 (32)	[4]	452	32	17	8500 (2.20)
	[7]	968	96	5	5560 (1.44)
	[9]	794	96	5	4690 (1.22)
	Proposed	675	64	5	3855 (1.00)
BCH (255, 223, 4) (32)	[4]	538	32	24	14064 (3.45)
	[7]	903	96	5	5235 (1.28)
	[9]	863	96	5	5025 (1.23)
	Proposed	720	64	5	4080 (1.00)

References

[1] S. Lin and D. J. Costello, *Error control coding: Fundamentals and Applications*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall Inc., 2004.

[2] H. Yoo, J. Jung, J. Jo, and I.-C. Park, "Area-Efficient Multimode Encoding Architecture for Long BCH Codes," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 12, pp. 872–876, Dec. 2013.

[3] M. Sprachmann, "Automatic Generation of Parallel CRC Circuits," *IEEE Des. Test. Comput.*, vol. 18, no.3, pp 108–114, May 2001.

[4] T.-B. Pei. and C. Zukowski, "High-Speed Parallel CRC Circuits in VLSI," *IEEE Trans. Commun.*, vol. 40, no. 4, pp. 653–657, Apr. 1992,.

[5] J. Zhang, Z. Wang, Q, Hu, and J. Xiao, "Optimized Design for High-Speed Parallel BCH Encoder," in *Proc. IEEE Int. Workshop VLSI Des. & Video Technol. (IWVDVT)*, 2005, pp. 179–182.

[6] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 2, pp. 151–165, Feb. 1996.

[7] J. H. Derby, "High-Speed CRC Computation Using State-Space Transformations," in *Proc. IEEE Global Telecomm. Conf. (GLOBECOM)*, Nov. 2001, pp. 166–170.

[8] M. Ayinala and K. K. Parhi, "Efficient Parallel VLSI Architecture for Linear Feedback Shift Registers," in *Proc. IEEE Workshop on SiPS*, Oct. 2010, pp. 52–57.

[9] M. Ayinala and K. K. Parhi, "High-Speed Parallel Architectures for Linear Feedback Shift Registers," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4459–4469, Sep. 2011.

[10] I. Pomeranz, S. M. Reddy, "On Methods to Match a Test Pattern Generator to a Circuit-Under-Test," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 3, pp. 432–444, Sep. 1998.

[11] V. Tenentes, X. Kavousianos, and E. Kalligeros, "Single and Variable-State-Skip LFSRs: Bridging the Gap Between Test Data Compression and Test Set Embedding for IP Cores," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 10, pp. 1640–1644, Oct. 2010.

[12] J. Mathew, S. Banerjee, H. Rahaman, D. K. Pradhan, S. P. Mohanty, and A. M. Jabir, "On the Synthesis of Attack Tolerant Cryptographic Hardware," in *Proc. IEEE/IFIP Int. Conf. VLSI Syst. on Chip Conf. (VLSI-SoC)*, Sep. 2010, pp. 286–291.

[13] K. K. Parhi and D. G. Messerschmitt, "Pipeline Interleaving and Parallelism in Recursive Digital Filters–Part II: Pipelined Incremental Block Filtering," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 7, pp. 1118–1135, Jul. 1989

[14] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Englewood Cliffs, NJ: Prentice Hall, 1995.