High-Speed Parallel LFSR Architectures Based on Improved State-Space Transformations

Guanghui Hu, Jin Sha, and Zhongfeng Wang

***Abstract*— Linear feedback shift register (LFSR) has been widely applied in BCH and CRC encoding. In order to increase the system throughput, the parallelization of LFSR is usually needed. Previously, a technique named state-space transformation was presented to reduce the complexity of parallel LFSR architectures. Exhaustive searches are performed to find good transformation matrix candidates. This brief proposes a new technique for construction of the transformation matrix together with a more efficient searching algorithm. The realization results indicate that the proposed architecture outperforms the prior arts, improving the hardware efficiency by around 35% and the corresponding searching algorithm finds the desirable transformation matrix much faster.**

***Index Terms*— BCH encoder, CRC encoder, linear feedback shift register (LFSR), parallel architecture, state-space transformation.**

I. Introduction

Linear feedback shift registers (LFSRs) are extensively applied in BCH and CRC encoders to compute the remainder polynomial. A conventional encoding scheme for BCH(n, k) code is shown in Fig. 1. Although such a serial LFSR architecture can operate at very high frequency, it suffers from the inherent serial-in and serial-out limitation. When the throughput of this serial architecture cannot catch up with the system data rate, parallel processing must be considered to meet the general requirements of high-speed communications and realize a higher throughput rate beyond gigabits per second such as in optical transmission.

Some kinds of parallel LFSR architectures have already been presented in the related literature for BCH and CRC encoders. In [1], a parallel CRC implementation has been designed through mathematical deduction. Tree-structured computation and subexpressions sharing are adopted to optimize the cascaded logic parts. Parhi [2] and Zhang and Parhi [3] proposed an improved high-speed BCH encoder designed to eliminate the fanout bottleneck. This parallel LFSR architecture is efficient in terms of speeding up the computation, but its hardware cost is high. A kind of state-space transformation is developed in [4] and [5] to reduce the complexity of the conventional parallel CRC circuits. By adopting linear matrix transformation, a full speedup factor can be achieved at the cost of an additional circuitry outside the feedback loop. Ayinala and Parhi [6] and Jung *et al.* [7] proposed another kind of parallel LFSR based on IIR filter topology and its advantage is that the pipeline technique can be applied to achieve some improvement in the hardware efficiency of LFSR encoders.

Manuscript received February 22, 2016; revised May 31, 2016 and July 22, 2016; accepted September 6, 2016. Date of publication September 26, 2016; date of current version February 22, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61370040, Grant 61006018, Grant 61376075, and Grant 61176024, in part by the Project on the Integration of Industry, Education and Research of Jiangsu Province under Grant BY2015069-05 and Grant BY2015069-08, and in part by the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions. *(Corresponding author: Jin Sha.)*

The authors are with Nanjing University, Nanjing 210046, China (e-mail: griffin.hu@gmail.com; shajin@nju.edu.cn).

Digital Object Identifier 10.1109/TVLSI.2016.2608921

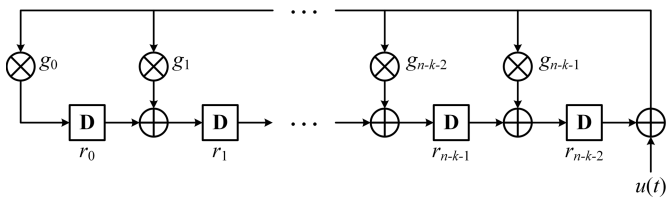

Fig. 1. Conventional serial LFSR architecture.

Among these parallel architectures, Ayinala and Parhi [6] and Jung *et al.* [7] achieve comparatively efficient hardware implementations with the loop update equations. In state-space transformation, the encoder is composed of some matrix multiplications, with the most effort devoted to searching of transformation matrices that may achieve the minimal-area circuits. In this brief, a new transformation matrix construction and an approximate searching method are proposed. Using this approximate algorithm, the desirable transformation matrix can be found in a much shorter time than exhaustive searches. A low-complexity and high-speed parallel LFSR architecture can be derived by applying the state-space transformation with the desirable solution. As a result, the hardware complexity can be effectively reduced without sacrificing performance or speed.

The rest of this brief is organized as follows. In Section II, the basic serial and parallel LFSR designs are reviewed briefly. State-space transformation and IIR-based architectures are mentioned as two typical parallel schemes. Section III introduces the method to construct the modified transformation matrix, and the corresponding searching algorithms for the desirable transformation matrix are described in Section IV. Section V gives the experimental results and makes a comparison. Conclusions are drawn in Section VI.

II. Basic Serial and Parallel LFSR Architectures

A conventional serial LFSR architecture for BCH(n,k) code is illustrated in Fig. 1. The symbol $\oplus$ indicates an XOR operation, and the symbol $\otimes$ indicates a finite-field multiply operation. In binary codes, since the coefficient of the generator polynomial $g_i (0 \le i \le n-k)$ equals either 0 or 1, the corresponding multiply can be simplified to a connection or disconnection. Registers denoted as $r_0, r_1, \ldots, r_{n-k-1}$ represent the remainder polynomial. Two kinds of typical parallel LFSR architectures are introduced next.

A. Parallel Architectures Based on State-Space Model

Supposing an $n-k$ dimensional state vector $\mathbf{R}(t) = (r_{n-k-1}(t), \ldots, r_1(t), r_0(t))^T$, where $r_i(t)$ represents the state of the ith remainder registers at time t, and $u(t)$ represents the tth single-bit input, $\mathbf{R}(t+1)$ can be expressed as

$$\mathbf{R}(t+1) = \mathbf{A} \times \mathbf{R}(t) + \mathbf{b} \times u(t) \quad (1)$$

where matrix $\mathbf{A}$ is

$$\mathbf{A} = \begin{pmatrix} g_{n-k-1} & 1 & 0 & \cdots & 0 \\ g_{n-k-2} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ g_1 & 0 & 0 & 0 & 1 \\ g_0 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

and vector $\mathbf{b}$ is

$$\mathbf{b} = (g_{n-k-1}, \ldots, g_1, g_0)^T.$$

According to (1), $\mathbf{R}(t+p)$ can be derived from recursive deductions as

$$\mathbf{R}(t+p) = (\mathbf{A}^{p-1}\mathbf{b}, \ldots, \mathbf{Ab}, \mathbf{b}) \times \begin{pmatrix} u(0) \\ u(1) \\ \vdots \\ u(t+p-1) \end{pmatrix} + \mathbf{A}^p \times \mathbf{R}(t). \quad (2)$$

This means $\mathbf{R}(t+p)$ can be obtained from $\mathbf{R}(t)$ directly.

Assuming p message bits are input at the same time and p divides the message length k, the parallel input at time t is

$$\mathbf{U}_p(t) = (u(tp), u(tp+1), \ldots, u(tp+p-1))^T \left(t = 0, 1, \ldots, \frac{k}{p} - 1\right).$$

According to (2)

$$\mathbf{R}(t+1) = \mathbf{A}^p \times \mathbf{R}(t) + \mathbf{B}_p \times \mathbf{U}_p(t) \quad (3)$$

in the case of parallel input where matrix $\mathbf{B}_p$ is an $(n-k) \times p$ matrix

$$\mathbf{B}_p = (\mathbf{A}^{p-1}\mathbf{b}, \ldots, \mathbf{Ab}, \mathbf{b}).$$

To reduce the complexity of (3), a linear transformation of $\mathbf{R}(t)$ is applied through a nonsingular matrix $\mathbf{T}$

$$\mathbf{R}(t) = \mathbf{T} \times \mathbf{R}_T(t).$$

The state equation (3) can be rewritten as

$$\begin{aligned} \mathbf{R}_T(t+1) &= \mathbf{A}_{pT} \times \mathbf{R}_T(t) + \mathbf{B}_{pT} \times \mathbf{U}_p(t) \\ \mathbf{R}(t+1) &= \mathbf{T} \times \mathbf{R}_T(t+1) \end{aligned} \quad (4)$$

where

$$\begin{aligned} \mathbf{A}_{pT} &= \mathbf{T}^{-1} \times \mathbf{A}^p \times \mathbf{T} \\ \mathbf{B}_{pT} &= \mathbf{T}^{-1} \times \mathbf{B}_p. \end{aligned} \quad (5)$$

Fig. 2 shows a block diagram of the system described by (4).

Derby [4] proposed a kind of transformation matrix $\mathbf{T}$ constructed by an arbitrary $n-k$ dimensional vector $\mathbf{b}_1$

$$\mathbf{T} = (\mathbf{b}_1, \mathbf{A}^p\mathbf{b}_1, \mathbf{A}^{2p}\mathbf{b}_1, \ldots, \mathbf{A}^{(n-k-1)p}\mathbf{b}_1).$$

If $\mathbf{b}_1$ can be chosen such that $\mathbf{T}$ is reversible, $\mathbf{A}_{pT}$ will be a companion matrix that has at most two nonzero elements in one row. Thereby the complexity of the feedback loop can be limited. However, the hardware complexity of state-space architectures is not decided by the number of 1s in matrix $\mathbf{A}_{pT}$ only, but the total denseness in $\mathbf{T}$, $\mathbf{A}_{pT}$, and $\mathbf{B}_{pT}$. In addition, the exhaustive search method should be applied for the construction of low-complexity $\mathbf{b}_1$, which needs to run more for than one month for CRC-32 generator polynomial [5].

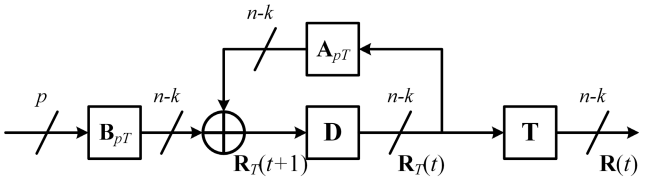

Fig. 2. Block diagram of the parallel LFSR architecture applying state-space transformation.

TABLE I
FREQUENTLY REFERENCED GENERATOR POLYNOMIALS

Name	Polynmials
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$
CRC-16	$x^{16}+x^{15}+x^2+1$
SDLC	$x^{16}+x^{12}+x^5+1$
CRC-16 REVERSE	$x^{16}+x^{14}+x+1$
SDLC REVERSE	$x^{16}+x^{11}+x^4+1$
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

B. Parallel Architectures Based on IIR Filter Model

In [7], a kind of parallel LFSR architecture based on IIR filter model is proposed. The serial LFSR encoder shown in Fig. 1 is treated as an IIR filter, and the loop update equations can be derived by adopting the look-ahead scheme [8]. A parallel BCH encoder can be implemented based on these equations. The detailed introduction can be found in [7].

III. PROPOSED METHOD TO CONSTRUCT TRANSFORMATION MATRIX

The transformation matrix used in [4] is chosen such that matrix $\mathbf{A}_{pT}$ is a companion matrix, which will simplify the feedback loop of the parallel architecture. Unfortunately, when $\mathbf{A}^p$ is transformed into a companion matrix $\mathbf{A}_{pT}$, matrix $\mathbf{T}$ and $\mathbf{B}_{pT}$ may become complicated and dense with many 1s. Even the feedback loop is fast and of low complexity as expected; the other parts of the encoder may have a longer critical path with high complexity. After applying pipelining and retiming techniques to reduce the critical path, the data-path timing is still not satisfying and brings extra hardware cost as well.

On the other hand, the $(n-k) \times (n-k)$ matrix $\mathbf{T}$ has $2^{(n-k)\times(n-k)}$ possibilities, but the vector $\mathbf{b}_1$ has only 2^{n-k} possibilities. This difference indicates that only a minority of possibilities of $\mathbf{T}$ are considered if using $\mathbf{b}_1$ to generate $\mathbf{T}$. The transformation matrix derived from optimal $\mathbf{b}_1$ may not be the optimal one among all the possibilities of $\mathbf{T}$.

Since there are other types of matrices that may transform the circuits into more efficient designs, some attempts can be devoted to improve the method for constructing matrix $\mathbf{T}$ and to further optimize the hardware implementations with state-space transformations. In order to find a better transformation matrix, some constraints need to be defined here.

First, the transformation matrix $\mathbf{T}$ must be reversible to make the state-space transformation workable.

Second, the total number of 1s in matrices $\mathbf{T}$, $\mathbf{A}_{pT}$, and $\mathbf{B}_{pT}$ should be as small as possible. The number of 1s in these matrices determines the number of XOR gates of the encoder directly.

TABLE II
COMPARISON BETWEEN THE STATE-SPACE REPRESENTATIONS USING PROPOSED TRANSFORMATION MATRIX AND [5]

Name	Algorithm	b_1^*	$\mathbf{A}_{pT}$			$\mathbf{B}_{pT}$			$\mathbf{T}$			Total		
		b_2^*	1	XOR	CPD	1	XOR	CPD	1	XOR	CPD	**TN**	XOR	CPD
CRC-12	[5]	$[0x814]^T$	20	10	1	54	31	5	46	13	3	120	66	5
	Proposed	[0xA01]	29	19	3	25	17	4	23	5	2	77	53	4
CRC-16	[5]	$[0xC00D]^T$	18	2	1	80	45	5	90	35	4	188	98	5
	Proposed	[0xC001]	35	23	2	33	15	4	32	6	2	100	60	5
SDLC	[5]	$[0x908C]^T$	18	2	1	106	59	5	102	37	4	226	114	5
	Proposed	[0x8408]	88	46	3	45	25	3	31	10	2	164	97	3
CRC-16 REVERSE	[5]	$[0x7401]^T$	18	2	1	80	46	4	92	33	4	190	97	4
	Proposed	[0xC002]	154	32	4	73	19	4	33	15	2	260	82	4
SDLC REVERSE	[5]	$[0xAC1F]^T$	18	2	1	106	59	4	102	39	4	226	116	4
	Proposed	[0x8810]	84	41	4	38	20	3	33	13	2	155	90	4
CRC-32	[5]	$[0xD8405018]^T$	45	16	2	447	234	5	436	236	5	928	518	5
	Proposed	[0x80000212]	414	221	5	425	198	5	49	10	2	888	461	5

Third, the method to find the transformation matrix needs to be efficient. The whole searching space is too huge for exhaustive search [5].

Based on these three constraints, a new method to construct the transformation matrix is proposed as follows.

First, a binary vector $\mathbf{b}_2$ with length $n-k$ is defined here. The first element of $\mathbf{b}_2$ is fixed as 1. The other elements of $\mathbf{b}_2$ can be arbitrary, i.e., $\mathbf{b}_2 = (1, b_1, b_2, \ldots, b_{n-k-1})$. Then the transformation matrix $\mathbf{T}$ is constructed as

$$\mathbf{T} = \begin{pmatrix} 1 & b_1 & b_2 & \cdots & b_{n-k-2} & b_{n-k-1} \\ 0 & 1 & b_1 & \cdots & b_{n-k-3} & b_{n-k-2} \\ 0 & 0 & 1 & \cdots & b_{n-k-4} & b_{n-k-3} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & b_1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Since the determinant of matrix $\mathbf{T}$ is fixed to 1, $\mathbf{T}$ is guaranteed to be a reversible matrix. In addition, $\mathbf{T}$ and its inverse are both upper triangular matrices, which makes all the transformed matrices sparse as well. Under a different choice of vector $\mathbf{b}_2$, the whole matrix set of $\mathbf{T}$, $\mathbf{A}_{pT}$, and $\mathbf{B}_{pT}$ will be different. Therefore, the hardware complexity will be decided by the choice of $\mathbf{b}_2$. In order to design the most efficient LFSR, exhaustive search of $\mathbf{b}_2$ may be applied. The specific description of the searching algorithm is given in the following section.

IV. METHODOLOGY

A. Basic Searching Algorithm

Six CRC or BCH codes are referenced here as examples. Their generator polynomials are listed in Table I. In order to find the desirable transformation matrix $\mathbf{T}$, an exhaustive search is performed in the vector space of $\mathbf{b}_2$. The parallel level p or the input size is chosen as the degree of generator polynomials for a clear comparison with previous architectures. The proposed method can be applied at any parallel level.

Since the matrices $\mathbf{T}$, $\mathbf{A}_{pT}$, and $\mathbf{B}_{pT}$ actually imply the connection of the coupling circuits, it follows that the hardware complexity is related to the total number of 1s in these three matrices, which is denoted as **TN** in the following. Consequently, we are interested in finding a $\mathbf{b}_2$ that minimizes **TN**.

The basic searching algorithm we used is as follows. First, the coupling matrices $\mathbf{A}$ and $\mathbf{B}_p$ can be computed for each of the generator polynomials in Table I with input size $p = n-k$. Then we traverse all the possible values of $\mathbf{b}_2$ to construct transformation matrix $\mathbf{T}$, and subject it to (5) to obtain transformed coupling matrices $\mathbf{A}_{pT}$ and $\mathbf{B}_{pT}$. Afterward, the **TN** is counted. The least one here indicates the lowest complexity of hardware, and the corresponding vector $\mathbf{b}_2$ is regarded as the best solution denoted as $\mathbf{b}_2^*$.

When $n-k = 12$ or $n-k = 16$, our searching algorithm is well qualified to find $\mathbf{b}_2^*$ in a reasonable time. The results are listed in Table II in terms of the set of the vectors ($\mathbf{b}_2^*$) (represented in hexadecimal), the number of 1s, the number of the XOR gates used in hardware circuits (XOR), and the critical path delay (CPD) (in terms of XOR gates). By contrast, Table II also lists the vectors ($\mathbf{b}_1^*$) used to construct the transformation matrices by the method described in [4]. Their detailed searching algorithm can be found in [5]. It should be noted that **TN** is regarded as the number of XOR gates directly in [5]. However, in our experiment, the actual number of XOR gates is less than **TN** by adopting the subexpressions sharing technique. For example, if there are four 1s forming a rectangle in matrix $\mathbf{T}$ ($\mathbf{A}_{pT}$ or $\mathbf{B}_{pT}$), one XOR is needed for the matrix computation instead of 2. Table II replaces the original number of XOR gates listed in [5] with the results we computed. We can see that the total number of XOR gates using the proposed transformation matrix is smaller than the results in [5], while the CPD basically remains the same.

However, for $n-k = 32$ or more, the vector space of $\mathbf{b}_2$ is too enormous and this exhaustive searching algorithm requires more than one month to obtain the results. This restricts the application of this searching algorithm for high-order generator polynomials. An alternative searching algorithm is proposed next to obtain the desired transformation matrix more quickly.

B. Approximate Searching Algorithm

For each $\mathbf{b}_2$, its corresponding $\mathbf{T}$, $\mathbf{A}_{pT}$, and $\mathbf{B}_{pT}$ can be uniquely decided and the **TN** can be counted. Using $n-k = 4$ as an example, all the possible $(n-k)$-dimensional $\mathbf{b}_2$ vectors can be divided into $n-k$ layers in a tree structure as shown in Fig. 3. The root node (the only node in layer 1) is set as $[100\ldots0]$, and each node in layer i is obtained by adding one "1" in the node of layer $i-1$. We denote this pair of nodes in adjacent layers as the "father node"

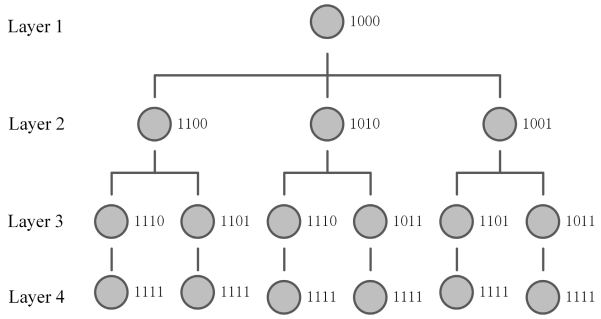

Fig. 3. All the possible values of a 4-D vector $\mathbf{b}_2$.

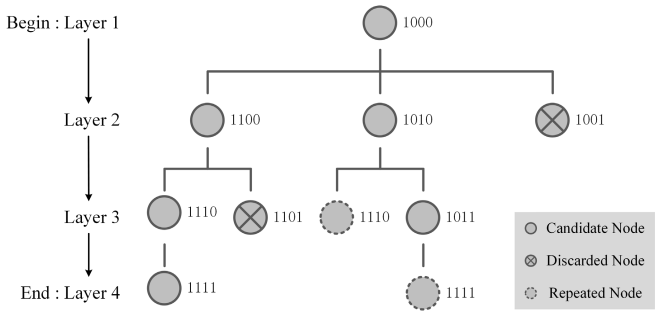

Fig. 4. Description of the approximate searching algorithm with $L = 2$.

TABLE III
SIZE OF THE SOLUTION SPACE FOR CRC-16 GENERATOR POLYNOMIAL

Algorithm	Size of solution space	Smallest **TN** result
$L = \infty$	32767	100
$L = 200$	8561	100
$L = 100$	5229	100
$L = 50$	3038	100

and "child node," and use a line to connect them. Constructed in this way, layer i actually contains all the $\mathbf{b}_2$ vectors with i 1s.

By extensive simulation, we found that when a father node has a large **TN**, its child nodes tend to have large **TN** values as well, and vice versa. Therefore, an approximate algorithm is proposed as follows.

1) We define $\mathbf{S}_1$ as a set of candidate nodes, i.e., nodes tending to have child nodes with small **TN**. At first, set $\mathbf{S}_1$ is initialized as vector $[100\ldots0]$, i.e., the only node in layer 1.
2) We define another set $\mathbf{S}_2$ as a set of child nodes, which contains all the child nodes of each node in set $\mathbf{S}_1$. At first, set $\mathbf{S}_2$ is actually equivalent to layer 2.
3) The **TN** of each node in set $\mathbf{S}_2$ is calculated, and the node with the smallest **TN** is chosen as the "optimal node" of the equivalent layer.
4) We select the L nodes with L smallest **TN** in $\mathbf{S}_2$ as the new candidate nodes and discard the others. Set $\mathbf{S}_1$ is updated with these candidate nodes. Operations in steps 2)–4) are repeated until all the layers are traversed.
5) By comparing all the "optimal nodes" in each layer, the node with the smallest **TN** is chosen as the final result.

Fig. 4 describes the operations of the approximate searching algorithm. The search begins at the root node and goes down layer by layer. Each searching path covers several candidates. If $C_{n-k-1}^{i-1} \leq L$, all the nodes in layer i are kept as candidates. Else, only the L nodes with the minimum **TN** in layer i are kept and the other nodes are discarded. Therefore, for the approximate searching algorithm, the size of the solution space is about $(1/2) \times L \times (n-k)(n-k-1)$, while for the basic searching algorithm, the size is $2^{n-k-1}-1$. The difference between them indicates the shortening of the runtime since the searching time is proportional to the size of the searching space.

Table III lists the size of the solution space for CRC-16. It takes about one minute for the exhaustive search to find the smallest **TN**. By restricting L, the runtime can be further reduced without changing the searching results. Table IV shows the size of the solution space for CRC-32. The result of the exhaustive search is not available as more than one month will be needed for searching throughout the solution space. By restricting L, we can obtain the results in a reasonable time. It can be seen that $L = 10\,000$ is not enough to get the smallest **TN**.

TABLE IV
SIZE OF THE SOLUTION SPACE FOR CRC-32 GENERATOR POLYNOMIAL

Algorithm	Size of solution space	Smallest **TN** result
$L = \infty$	2147483647	Not available
$L = 200000$	37622348	879
$L = 100000$	26014234	879
$L = 50000$	13142546	879
$L = 10000$	2771714	882

TABLE V
COMPARISON BETWEEN SEVERAL ARCHITECTURES

Name (p)	Algorithm	XOR	DE	CPD	AT
CRC-12 (12)	[4]	113	35	5	827.5 (2.32)
	[6]	109	36	5	815 (2.28)
	[7]	103	24	4	556 (1.56)
	Proposed	53	24	4	356 (1.00)
CRC-16 (16)	[4]	187	48	5	1295 (2.39)
	[6]	113	50	5	940 (1.74)
	[7]	94	32	5	710 (1.31)
	Proposed	60	32	5	540 (1.00)
SDLC (16)	[4]	207	48	5	1395 (3.21)
	[6]	139	50	5	1070 (2.45)
	[7]	97	32	3	435 (1.00)
	Proposed	97	32	3	435 (1.00)
CRC-16 REVERSE (16)	[4]	219	46	5	1440 (2.76)
	[6]	126	50	5	1005 (1.93)
	[7]	82	32	4	520 (1.00)
	Proposed	82	32	4	520 (1.00)
SDLC REVERSE (16)	[4]	217	48	5	1445 (2.61)
	[6]	127	50	5	1010 (1.82)
	[7]	90	32	4	552 (1.00)
	Proposed	90	32	4	552 (1.00)
CRC-32 (32)	[4]	968	96	5	5560 (1.99)
	[6]	794	96	5	4690 (1.68)
	[7]	675	64	5	3855 (1.38)
	Proposed	461	64	5	2785 (1.00)
BCH (225,223,4) (32)	[4]	903	96	5	5235 (1.75)
	[6]	863	96	5	5035 (1.68)
	[7]	720	64	5	4080 (1.36)
	Proposed	502	64	5	2990 (1.00)

When L is more than 50 000, a satisfying result is achieved and we use this vector in our encoder design.

V. Comparison and Analysis

For different generator polynomials, the proposed architecture is compared with previous ones and the result is shown in Table V. The parallelism level p is chosen equal to the degree of each generator polynomial as well. The area-time product (AT) is regarded as a reasonable parameter to indicate the efficiency

$$\mathrm{AT} = (1.5 \cdot \mathrm{DE} + \mathrm{XOR}) \cdot \mathrm{CPD}$$

where DE denotes the number of delay elements. According to the results, our parallel architectures can achieve hardware designs with the minimal XOR gates and the same CPD. In order to make the data more intuitive, the relative AT values are listed in the rightmost column marked with parentheses, normalized by the proposed method.

One thing attracts our attention in the course of the experiment. If we use the coefficients of the generator polynomials ($g_0, g_1, \ldots, g_{n-k-1}$) as the vector $\mathbf{b}_2$ directly, the transformed circuits are the same as those proposed in [7] (introduced in Section II-B). This means that the architecture proposed in [7] is equivalent to a special case of the proposed parallel architectures. In addition, for SDLC, CRC-16 reverse, and SDLC reverse, the $\mathbf{b}_2^*$ vectors we find are just the coefficients of their corresponding generator polynomials. Thus for these three ones, our architectures have the same results as [7], and as given in Table V. For other generator polynomials, our design can always achieve the best AT. Therefore, the proposed method has an outstanding effect to reduce the area and achieve high-speed hardware designs with low complexity.

VI. Conclusion

This brief has proposed a new method to construct the transformation matrix used in the state-space transformation. A time-saving searching algorithm has been presented for searching of a good transformation matrix as well. Based on this improved state-space transformation, reduced-complexity parallel architectures can be obtained. Experimental results have shown that the proposed architecture outperforms the previous designs for high-speed implementation of BCH or CRC encoders.

References

[1] 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.

[2] K. K. Parhi, "Eliminating the fanout bottleneck in parallel long BCH encoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 3, pp. 512–516, Mar. 2004.

[3] X. Zhang and K. K. Parhi, "High-speed architectures for parallel long BCH encoders," in *Proc. ACM Great Lakes Symp. VLSI*, Boston, MA, USA, Apr. 2004, pp. 1–6.

[4] J. H. Derby, "High-speed CRC computation using state-space transformations," in *Proc. IEEE GLOBECOM*, Nov. 2001, pp. 166–170.

[5] C. Kennedy and A. Reyhani-Masoleh, "High-speed CRC computations using improved state-space transformations," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, Jun. 2009, pp. 9–14.

[6] 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.

[7] J. Jung, H. Yoo, Y. Lee, and I.-C. Park, "Efficient parallel architecture for linear feedback shift registers," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 62, no. 11, pp. 1068–1072, Nov. 2015.

[8] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters. II. Pipelined incremental block filtering," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 7, pp. 1118–1134, Jul. 1989.

[9] C. Kennedy and A. Reyhani-Masoleh, "High-speed parallel CRC circuits," in *Proc. 42nd Annu. Asilomar Conf. Signals, Syst., Comput.*, Oct. 2008, pp. 1823–1829.

[10] C. Cheng and K. K. Parhi, "High-speed parallel CRC implementation based on unfolding, pipelining, and retiming," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 53, no. 10, pp. 1017–1021, Oct. 2006.

[11] M. Y. Hsiao and K. Y. Sih, "Serial-to-parallel transformation of linear-feedback shift-register circuits," *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 6, pp. 738–740, Dec. 1964.

[12] M. Ayinala and K. K. Parhi, "Efficient parallel VLSI architecture for linear feedback shift registers," in *Proc. IEEE Workshop Signal Process. Syst.*, Oct. 2010, pp. 52–57.