Low-Complexity Parallel Cyclic Redundancy Check

Xinmiao Zhang and Yok Jye Tang
The Ohio State University

***Abstract*—Cyclic redundancy check (CRC) is adopted in many digital communication and storage systems to ensure data integrity. CRC en/decoding is carried out using linear feedback shift registers (LFSRs) and a parallel LFSR can be implemented by registers with a feedback matrix multiplication and an input pre-processing matrix multiplication. A large parallelism is needed to achieve the high throughput required by modern applications. In prior designs, the complexity of parallel LFSRs has been reduced by applying state transformation and/or modifying the input tap. In this paper, we first show that the input tap modification can be actually described by a category of state transformation. Using this type of transformation, the pre-processing matrix in a highly-parallel LFSR can be simplified without changing the feedback matrix. Additionally, we show that the post-processing matrix multiplication in state-transformed designs can be eliminated without affecting the error detection capability of the CRC. Utilizing these two techniques, the area requirement of highly-parallel CRC can be reduced by 7-16% without increasing the critical path for various parallelisms and most generator polynomials compared to the best previous design.**

***Index Terms*—Error correction code (ECC), cyclic redundancy check (CRC), linear feedback shift register (LFSR).**

I. Introduction

Cyclic redundancy check (CRC) is used to detect errors in many digital communication and storage systems, such as Flash memory and optical communications. Both the encoding and decoding of CRC are basically to compute the remainder of the data polynomial divided by the generator polynomial of CRC. Such computations can be implemented by linear feedback shift registers (LFSRs). To reach a throughput in the order of GigaBytes/s, highly parallel designs are needed.

A p-parallel LFSR can be derived by applying look-ahead computation to the register state of a serial LFSR for p iterations [1]. Such parallel LFSRs are typically smaller than those derived by unfolding [2]–[4] or parallel recursive filtering [5], [6]. A parallel LFSR based on state look-ahead has a matrix multiplication in the feedback loop and an input pre-processing matrix multiplication. The register state transformation proposed in [7] results in modified feedback and pre-processing matrices and adds a post-processing matrix multiplication at the end. The data path of the feedback matrix multiplication is reduced to one XOR gate by using companion-matrix-like transformation matrices [7], [8]. The design in [9] searches for transformation matrices in a triangular form to minimize the overall gate count. By searching for inverse transformation matrices in a triangular form that minimize the number of nonzero entries in the pre-processing matrix, the design in [10] achieves lower gate count and power consumption.

This material is based upon work supported by the National Science Foundation under Award No. 2011785

It was shown in [11] that, when p is larger than the generator polynomial degree, adding the data input to the least significant tap (LST) instead of the most significant tap (MST) of the LFSR as in conventional designs makes the pre-processing matrix consist of an identity matix instead of all dense columns. Moreover, when p is less than the degree of the generator polynomial and the same remainder as in MST-input design needs to be computed, inserting the data to the p-th tap leads to the simplest design [12].

This paper proposes two techniques to further reduce the complexity of CRCs for high-throughput digital communications and storage, in which case the parallelism is larger than the generator polynomial degree. When the input is added to a different tap of the LFSR, the register state update formula in [11] is derived by analyzing the effect of the input on each register. The formula derivation is not associated with any state transformation and the shifting of the input is limited to from the MST to the LST. The first discovery made in this paper is that the input tap modification can be actually described as a state transformation using a power of the companion matrix as the transformation matrix. This interpretation allows larger space to be exploited to find even simpler pre-processing matrix without changing the feedback matrix. Besides, an additional state transformation for reducing the gate count can be further applied. The second simplification utilizes the property that the remainder computed for CRC can be bijectively mapped to another vector as long as the same mapping is used in both the encoder and decoder. Accordingly, the post-processing matrix multiplication can be eliminated. Utilizing these two techniques, the gate counts of CRCs with various parallelisms for most generator polynomials are reduced by 7-16% compared to the best previous design [11] without sacrificing the critical path.

II. Parallel LFSRs and state transformation

A CRC that adds $n-k$ parity bits to a data string, $u(x)$, of length k can be specified by a generator polynomial $g(x) = x^{n-k} + g_{n-k-1}x^{n-k-1} + \cdots + g_1 x + g_0$. In CRC encoding, the remainder of dividing $u(x)x^{n-k}$ by $g(x)$ is computed and padded to $u(x)$. In decoding, the received data polynomial is multiplied by x^{n-k} and divided by $g(x)$. If the remainder matches that from encoding, then it is considered that there is no error. The LFSR in Fig. 1 divides $u(x)x^{n-k}$ by $g(x)$. The coefficients of $u(x)$ are input serially starting with the

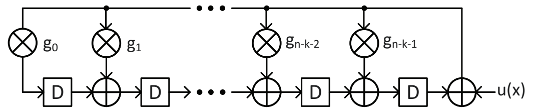

Fig. 1. Serial LFSR architecture with input added to the MST

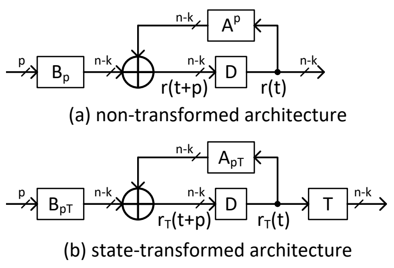

Fig. 2. p-parallel LFSR architectures

most significant one. After the last coefficient is processed, the remainder polynomial, $r(x)$, is available in the registers.

Denote the state of the register in clock cycle t by $\mathbf{r}(t) = [r_{n-k-1}(t), r_{n-k-2}(t), \cdots, r_0(t)]'$, where '$'$' means transpose. Let $u(t)$ be the input in clock cycle t. From Fig. 1,

$$\mathbf{r}(t+1) = \mathbf{A} \times \mathbf{r}(t) + \mathbf{b} \times u(t), \tag{1}$$

where

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

is a companion matrix and $\mathbf{b} = [g_{n-k-1}, \cdots, g_1, g_0]'$. Substituting (1) back to itself for p times, it can be derived that

$$\mathbf{r}(t+p) = \mathbf{A}^p \times \mathbf{r}(t) + \mathbf{B}_p \times \mathbf{u}_p(t), \tag{2}$$

and a p-parallel LFSR can be implemented as shown in Fig. 2(a). Here $\mathbf{B}_p = [\mathbf{A}^{p-1}\mathbf{b}, \cdots, \mathbf{A}\mathbf{b}, \mathbf{b}]$ and $\mathbf{u}_p(t) = [u(t), \cdots, u(t+p-2), u(t+p-1)]'$.

The state vector can be transformed to $\mathbf{r}(t) = \mathbf{T} \times \mathbf{r}_T(t)$ [7] by using any non-singular $\mathbf{T}$. Then (2) becomes

$$\mathbf{r}_T(t+p) = \mathbf{A}_{pT} \times \mathbf{r}_T(t) + \mathbf{B}_{pT} \times \mathbf{u}_p(t), \tag{3}$$

and a transformed p-parallel architecture is implemented as in Fig. 2(b). Here $\mathbf{A}_{pT} = \mathbf{T}^{-1}\mathbf{A}^p\mathbf{T}$ and $\mathbf{B}_{pT} = \mathbf{T}^{-1}\mathbf{B}_p$. It is impractical to exhaustively search for the optimal transformation matrix T. Hence, existing works [7]–[10] considered $\mathbf{T}$ of certain structures, such as companion-like and triangular, to achieve various optimization goals. $\mathbf{A}^p$ and $\mathbf{A}_{pT}$ are referred to as the feedback matrices in the respective designs, and $\mathbf{B}_p$ and $\mathbf{B}_{pT}$ are called pre-processing matrices.

When the input is added to the j-th tap, the serial LFSR in Fig. 1 computes $u(x)x^j/g(x)$. $n-k-j$ zeros can be padded to $u(x)$ to compute the remainder of $u(x)x^{n-k}/g(x)$. By analyzing the contributions of the input to the register state update, it was found in [11] that the $\mathbf{b}$ in (1) should be replaced by $\mathbf{b}^{(j)}$, which has a single nonzero entry in the j-th bit, if the input is added to the j-th ($j < n-k$) tap. Accordingly, the $\mathbf{B}_p$ in (2) is replaced by $\mathbf{B}_p^{(j)} = [\mathbf{A}^{p-1}\mathbf{b}^{(j)}, \cdots, \mathbf{A}\mathbf{b}^{(j)}, \mathbf{b}^{(j)}]$. It was also shown in [11] that $\mathbf{B}_p^{(j)}$ with smaller j has more columns with single nonzero entry. In particular, when $p > (n-k)$ and the input is added to the LST, $\mathbf{B}_p^{(0)}$ consists of an identity part. As a result, the LST-input design has smaller gate count than all prior architectures. Besides, state transformation may be further applied to achieve additional complexity reduction.

III. Reduced-complexity highly-parallel CRC

Two techniques are proposed in this section to further reduce the complexity of highly-parallel CRC. The first connects the input-tap modification to state transformation, through which larger space is exploited to further reduce the complexity of the pre-processing matrix. The second utilizes the fact that that error detection in CRC is made based on remainder comparisons to eliminate unnecessary computations.

$\mathbf{A}$ is a companion matrix whose leftmost column is specified by the generator polynomial $g(x)$. Let $\mathbf{a} = [a_{n-k-1}, \cdots, a_1, a_0]'$. It can be derived that

$$\mathbf{A}\mathbf{a} = ([a_{n-k-2}, \cdots, a_0, 0] \text{XOR} (a_{n-k-1} \cdot [g_{n-k-1}, \cdots, g_1, g_0]))'.$$

Computations in exactly the same format are done when an element $a \in GF(2^{n-k})$ in standard basis representation is multiplied by $\alpha \in GF(2^{n-k})$, which is a root of the irreducible polynomial used to construct $GF(2^{n-k})$. Hence, although $g(x)$ is not necessarily an irreducible polynomial, multiplying $\mathbf{A}$ to a vector $\mathbf{a}$ can be translated as multiplying a root of $g(x)$, denoted by α, to an element of $GF(2^{n-k})$ whose standard basis representation is $\mathbf{a}$. In other words, $\mathbf{A}$ can be considered as the binary matrix representing the multiplication by α.

As it was discovered in [11] through analyzing the contribution of the input to the register state, when the input is added to the j-th tap, the pre-processing matrix becomes $\mathbf{B}_p^{(j)} = [\mathbf{A}^{p-1}\mathbf{b}^{(j)}, \cdots, \mathbf{A}\mathbf{b}^{(j)}, \mathbf{b}^{(j)}]$ while the feedback matrix $\mathbf{A}^p$ is unchanged. $\mathbf{b}^{(j)}$ ($j < (n-k)$) is a vector that is only '1' in the j-th bit. Hence, it can be considered as the standard basis representation of α^j. Accordingly, $\mathbf{B}_p^{(j)}$ consists of columns that are standard basis representations of $\alpha^{p-1+j}, \cdots, \alpha^{1+j}, \alpha^j$. Similarly, the pre-processing matrix for the MST-input design, $\mathbf{B}_p$, consists of columns representing $\alpha^{p-1+n-k}, \cdots, \alpha^{1+n-k}, \alpha^{n-k}$ since $\mathbf{b} = [g_{n-k-1}, \cdots, g_1, g_0]'$ is the standard basis representation of α^{n-k}. Hence $\mathbf{B}_p^{(j)} = \mathbf{A}^{-(n-k-j)}\mathbf{B}_p$. $\mathbf{B}_p^{(j)}$ can be considered as $\mathbf{B}_p$ transformed by using $\mathbf{T}^{-1} = (\mathbf{A}^{n-k-j})^{-1}$. Let $\mathbf{D}$ be the binary matrix describing the multiplication by an element $\delta \in GF(2^q)$. Assume that the binary inverse of $\mathbf{D}$ is $\mathbf{D}^{-1}$. $\mathbf{D}^{-1}$ actually also equals the binary matrix representing the multiplication by $\delta^{-1} \in GF(2^q)$ [13]. As a result, $\mathbf{T}^{-1}\mathbf{A}^p\mathbf{T} = (\mathbf{A}^{n-k-j})^{-1}\mathbf{A}^p\mathbf{A}^{n-k-j}$ represents the multiplication by $\alpha^{-(n-k-j)+p+(n-k-j)} = \alpha^p$. Therefore, the feedback matrix is still $\mathbf{A}^p$ and it is unchanged by using transformation $\mathbf{T} = \mathbf{A}^{(n-k-j)}$. This matches the design in [11]. Nevertheless, it was not realized in [11] that the input tap modification can be described by state transformation.

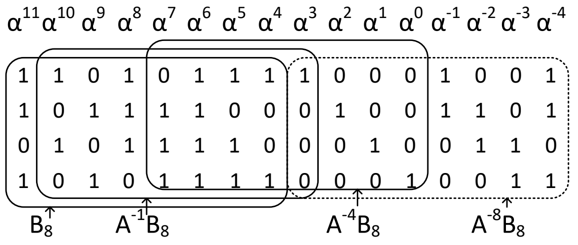

Fig. 3. Example array of columns for pre-processing matrices when $g(x) = x^4 + x^3 + 1$

The design of [11] is limited to adding the input to one of the $n-k+1$ taps of the LFSR. In terms of state transformation, this means that $\mathbf{T}$ can only be $\mathbf{A}^{n-k-j}$ with $0 \le j \le n-k$. Relaxing this constraint, any $\mathbf{A}^j$ with $0 \le j < 2^{n-k} - 2$ can be used as the transformation matrix, although employing $j > n-k$ does not correspond to inserting the input to any physical taps. This relaxation allows larger design space to be exploited to further simplify the pre-processing matrix.

The vector representations of $\alpha^0, \alpha^1, \alpha^2, \cdots$ can be listed as the columns in an array. Fig. 3 shows a toy example for the case that $n-k=4$ and $g(x) = x^4 + x^3 + 1$. Over $GF(2^4)$, $\alpha^{11} = \alpha^{2^4-1-4} = \alpha^{-4}$. For the purpose of showing different pre-processing matrices, the column of α^{11} wraps around and is shown as a duplicated column of α^{-4} in Fig. 3. Using a power of $\mathbf{A}$ as the transformation matrix, the transformed pre-processing matrix consists of a window of p consecutive columns in the array. The window for $\mathbf{B}_p$, which corresponds to the case that $\mathbf{T} = \mathbf{A}^0 = \mathbf{I}$, starts from the column for α^{n-k} and goes to the left. The window for $\mathbf{A}^{-j}\mathbf{B}_p$ starts from the column for α^{n-k-j}. The windows shown in Fig. 3 are for the case of $p = 8$. By exhaustively gliding the window to cover different p consecutive columns, the simplest pre-processing matrix and its corresponding transformation matrix can be found. When $n-k$ is larger, such as 32, this exhaustive search takes very long time. Instead, the search can be centered around the columns for $\alpha^{n-k-1}, \cdots, \alpha^1, \alpha^0$ since they are the most sparse columns in the array. The input-tap modification in [11] only allows the right border of the window to shift from the column of α^{n-k} to the column of α^0. On the contrary, the proposed relaxation allows the window to pass beyond the α^0 column. Depending on the pattern of $g(x)$, the columns to the right of the α^0 column may be less dense than those to the left of the α^{n-k} column, as shown by the example in Fig. 3. As a result, pre-processing matrices of even lower complexity may be found. Besides, the transformation in [10] with lower-triangular matrix can be additionally applied to further reduce the complexity.

When state transformation is applied, $\mathbf{T}$ needs to be multiplied at the end as shown in Fig. 2(b) to get the remainder of $u(x)x^{n-k}/g(x)$. In the modified input tap design of [11], adding the input to the j-th tap $(0 \le j \le n-k)$, which is equivalent to using $\mathbf{T} = \mathbf{A}^{n-k-j}$, makes the register state equal the remainder of $u(x)x^j/g(x)$ after the last data bit is sent in. $u(x)x^j$ is just $u(x)x^{n-k}$ with the least significant zeros eliminated. If the same number of zeros are deleted in both the CRC encoder and decoder, the remainders should match.

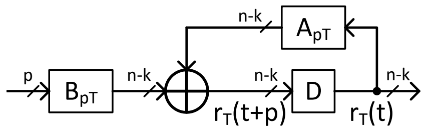

Fig. 4. Proposed p-parallel CRC architecture without post-processing matrix multiplication

TABLE I
GENERATOR POLYNOMIALS OF CRCs

	Generator polynomial
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	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}$ $+x^8+x^7+x^5+x^4+x^2+x+1$

Hence, the design in [11] does not have the $\mathbf{T}$ matrix multiplication at the end. Actually, the remainder of $u(x)x^{n-k}/g(x)$ can be (inversely) mapped to another vector without affecting the error detection capability of CRC as long as the same bijective mapping is used in both the encoder and decoder. Since a valid transformation matrix, which is not limited to the format of $\mathbf{T} = \mathbf{A}^{n-k-j}$, is always invertible, the matching test can be carried out on the register state directly and the multiplication with $\mathbf{T}$ at the end can be always eliminated. The proposed CRC architecture without the post-processing $\mathbf{T}$ matrix multiplication is shown in Fig. 4.

IV. Hardware complexity comparisons

This section compares the hardware complexity of the proposed parallel CRC architecture with those of previous designs for various generator polynomials listed in Table I. The same generator polynomials have been considered in prior works. Since the proposed design is targeting at highly parallel CRCs, $p = 2(n-k)$ and $4(n-k)$ are considered in the comparisons. Among prior designs, the one in [11] that adds the input to the LST, or equivalently using $\mathbf{T} = \mathbf{A}^{n-k}$ as the transformation matrix, achieves the lowest complexity for $p > n-k$ and it is utilized for comparison.

For both the proposed and LST-input design, the transformation using lower-triangular matrices from [10] can be additionally applied to further reduce the complexity. For $p = 2(n-k)$, the complexities of these designs are listed in Table II. The designs with and without the additional transformation are denoted by 'trans' and 'n-trans', respectively. After the number of '1's in the matrices is minimized by searching for optimal transformations, the substructure sharing method that can set the critical path delay (CPD) constraint in [10] is employed to derive the numbers of XOR gates for all designs. These numbers are normalized with respect to that of the non-transformed LST-input design from [11] in Table II. For most CRCs, the proposed design achieves 7-16% total gate count reduction without increasing the critical path. Unlike the pre-processing and feedback matrices, the post-processing matrix only needs to be multiplied in the last clock cycle. To compare the power consumption, the number of XOR gates that are

TABLE II
COMPARISONS OF $p = 2(n-k)$-PARALLEL CRC

	design	$\mathbf{A}_{pT}(\mathbf{A}^p)$			$\mathbf{B}_{pT}^{(j)}(\mathbf{B}_p)$			$\mathbf{T}$			Total		XORs active every
		'1'	XOR	CPD	'1'	XOR	CPD	'1'	XOR	CPD	XOR (normalized)	CPD	clock (normalized)
CRC-12	[11] (n-trans.)	50	25	4	64	33	5	-	-	-	70 (1)	6	70 (1)
	[11] (trans.)	35	20	3	50	32	4	17	5	1	69 (0.99)	5	64(0.91)
	proposed (n-trans.)	50	25	4	64	33	5	-	-	-	70 (1)	6	70 (1)
	proposed (trans.)	35	20	3	50	32	4	-	-	-	64 (0.91)	5	64 (0.91)
CRC-16	[11] (n-trans.)	76	30	5	88	43	5	-	-	-	89 (1)	6	89 (1)
	[11] (trans.)	50	25	4	66	42	5	18	2	1	85 (0.96)	6	83 (0.93)
	proposed (n-trans.)	76	30	5	88	43	5	-	-	-	89 (1)	6	89 (1)
	proposed (trans.)	50	25	4	66	42	5	-	-	-	83(0.93)	6	83 (0.93)
SDLC	[11] (n-trans.)	130	63	4	104	56	4	-	-	-	135 (1)	5	135 (1)
	[11] (trans.)	113	57	4	82	52	3	31	12	3	137 (1.01)	5	125 (0.93)
	proposed (n-trans.)	130	63	4	78	48	3	-	-	-	127 (0.94)	5	127 (0.94)
	proposed (trans.)	125	57	4	78	49	3	-	-	-	122 (0.90)	5	122 (0.90)
CRC-16 Reverse	[11] (n-trans.)	154	48	4	170	60	5	-	-	-	124 (1)	6	124 (1)
	[11] (trans.)	129	42	4	65	46	4	134	34	4	138 (1.11)	5	104 (0.84)
	proposed (n-trans.)	154	48	4	88	45	4	-	-	-	109 (0.88)	5	109 (0.88)
	proposed (trans.)	150	46	4	66	42	4	-	-	-	104 (0.84)	5	104 (0.84)
SDLC Reverse	[11] (n-trans.)	108	55	4	100	53	4	-	-	-	124 (1)	5	124 (1)
	[11] (trans.)	103	55	3	79	49	3	32	14	3	134 (1.08)	4	120 (0.97)
	proposed (n-trans.)	108	55	4	78	48	3	-	-	-	119 (0.96)	5	119 (0.96)
	proposed (trans.)	106	57	4	78	49	3	-	-	-	122 (0.98)	5	122 (0.98)
CRC-32	[11] (n-trans.)	485	221	5	484	250	5	-	-	-	503 (1)	6	503 (1)
	[11] (trans.)	507	226	5	362	229	4	191	94	4	581 (1.16)	6	487 (0.97)
	proposed (n-trans.)	485	221	5	479	250	5	-	-	-	503 (1)	6	503 (1)
	proposed (trans.)	463	211	5	368	226	4	-	-	-	469 (0.93)	6	469 (0.93)

TABLE III
COMPARISONS OF $p = 4(n-k)$-PARALLEL CRC

	design	$\mathbf{A}_{pT}$ $(\mathbf{A}^p)$	$\mathbf{B}_{pT}^{(j)}$ $(\mathbf{B}_p)$	$\mathbf{T}$	Total		XORs active
					XOR	CPD	every clk
CRC-12	[11] (n-trans.)	26	103	-	141 (1)	6	1
	[11] (trans.)	25	95	2	134 (0.95)	6	0.94
	prop. (n-trans.)	26	98	-	136 (0.96)	6	0.96
	prop. (trans.)	24	91	-	127 (0.90)	6	0.90
CRC-16	[11] (n-trans.)	34	131	-	181 (1)	7	1
	[11] (trans.)	29	124	2	171 (0.94)	7	0.93
	prop. (n-trans.)	34	126	-	176 (0.97)	7	0.97
	prop. (trans.)	30	123	-	169 (0.93)	7	0.93
SDLC	[11] (n-trans.)	67	176	-	259 (1)	6	1
	[11] (trans.)	61	163	19	259 (1)	6	0.93
	prop. (n-trans.)	67	160	-	243 (0.94)	6	0.94
	prop. (trans.)	53	167	-	236 (0.91)	6	0.91
CRC-16 Reverse	[11] (n-trans.)	62	170	-	248 (1)	7	1
	[11] (trans.)	45	134	34	229 (0.92)	6	0.79
	prop. (n-trans.)	62	130	-	208 (0.84)	7	0.84
	prop. (trans.)	60	124	-	200 (0.81)	7	0.81
SDLC Reverse	[11] (n-trans.)	59	172	-	247 (1)	6	1
	[11] (trans.)	52	165	31	264 (1.07)	6	0.94
	prop. (n-trans.)	59	153	-	228 (0.92)	6	0.92
	prop. (trans.)	56	169	-	241 (0.98)	6	0.98
CRC-32	[11] (n-trans.)	232	675	-	939 (1)	7	1
	[11] (trans.)	225	616	115	988 (1.05)	7	0.93
	prop. (n-trans.)	232	675	-	939 (1)	7	1
	prop. (trans.)	225	616	-	873 (0.93)	7	0.93

active in every clock cycle is listed in the last column of Table II.

Since the LST-input design is one case in the proposed search, the proposed design does not ever have higher complexity. Besides, in previous transformed designs, the transformation matrix needs to be multiplied at the end although the complexities of the pre-processing and feedback matrix multiplications may be reduced. Overall, the total gate count may increase because of the transformation as shown in the cases of SDLC, CRC-16 Reverse, SDLC Reverse, and CRC-32 for LST-input designs [11] in Table II. On the other hand, our proposed design eliminates the post-processing matrix multiplication. Applying additional transformation with lower-triangular $\mathbf{T}$ always leads to further saving. The only exception is the SDLC Reverse CRC. This is because that the optimal transformation search is done based on the number of '1's to reduce the search time. However, the gate count after the substructure sharing depends on the patterns of '1's.

The complexities of CRCs with even higher parallelism, $p = 4(n-k)$, are presented in Table III. For this parallelism, the proposed design achieves 7-19% total gate count reduction for most CRCs. The achievable improvement is slightly bigger because there are more columns in the pre-processing matrix and the chance of finding matrices with fewer '1's is higher. For other parallelisms, we expect that the saving achievable by the proposed design is similar.

V. CONCLUSIONS

This paper first translates the idea of the LFSR input tap modification into a state transformation for parallel CRC design. By utilizing this interpretation, much wider design space is exploited to reduce the complexity of the pre-processing matrix without changing the feedback matrix in parallel CRC. Additionally, by utilizing the property that bijective mapping can be applied to the remainders used for matching test in CRC, the transformation matrix multiplication at the end is eliminated. Overall, the proposed design achieves significant complexity reduction without increasing the critical path. Future work will address more effective transformations for simplifying the multiplications of the pre-processing and feedback matrices.

References

[1] T.-B. Pei, and C. Zukowski, "High-speed parallel CRC circuits in VLSI," *IEEE Trans. on 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. on Circuits and Syst.-I*, vol. 51, no. 3, pp. 512 - 516, Mar. 2004.

[3] X. Zhang and K. K. Parhi, "High-speed architectures for parallel long BCH encoders," *IEEE Trans. on VLSI Syst.*, vol. 13, no. 7, pp. 872-877, Jul. 2005.

[4] Y. J. Tang and X. Zhang, "Low-complexity architectures for parallel long BCH encoders," *Proc. of IEEE Workshop on Signal Processing Syst.,* Coimbra, Portugal, Oct. 2019.

[5] M. Ayinala and K. K. Parhi, "High-speed parallel architectures for linear feedback shift registers," *IEEE Trans. on Signal Process.*, vol. 59, no. 9, pp. 4459-4469, Sep. 2011.

[6] J. Jung, *et. al.*, "Efficient parallel architecture for linear feedback shift registers," *IEEE Trans. on Circuits and Syst.-II*, vol. 62, no. 11, pp. 1068-1072, Nov. 2015.

[7] J. H. Derby, "High-speed CRC computation using state-space transformations," *Proc. IEEE Global Commun. Conf.*, pp. 166-170, Nov. 2001.

[8] C. Kennedy and A. Reyhani-Masoleh, "High-speed CRC computations using improved state-space transformation," *Proc. IEEE Intl. Conf. Electro/Info. Tech.*, pp. 9-14, 2009.

[9] G. Hu, J. Sha, and Z. Wang, "High-speed parallel LFSR architectures based on improved state-space transformations," *IEEE Trans. on VLSI Syst.* vol. 25, no. 3, pp. 1159-1163, Mar. 2017.

[10] X. Zhang, "A low-power parallel architecture for linear feedback shift registers," *IEEE Trans. on Circuits and Syst.-II*, 2019.

[11] X. Zhang, and Y. J. Tang, "Reducing parallel linear feedback shift register complexity through input tap modification," *Proc. of IEEE Intl. Symp. on Circuits and Systems*, Sapporo, Japan, May 2019.

[12] X. Zhang, "High-speed and low-complexity parallel long BCH encoder," *Proc. of IEEE Intl. Symp. on Circuits and Systems*, Seville, Spain, May 2020.

[13] J. Bloember, *et. al.*, "An XOR-based erasure-resilient coding scheme," *Technical Report*, 1995.