High-Speed and Low-Complexity Parallel Long BCH Encoder

Xinmiao Zhang
The Ohio State University, Columbus, OH 43210, U.S.

***Abstract*—Long BCH codes are broadly used in communications and storage. BCH encoders can be implemented by linear feedback shift registers (LFSRs), and modern systems demand parallel encoders to achieve high speed. LFSRs are also used for implementing cyclic redundancy check (CRC). It was shown previously that adding the input to the least significant tap of the CRC LFSR leads to the lowest complexity in the corresponding parallel architecture derived using state transition. However, the analyses for CRC LFSRs do not hold for long BCH encoders, since the parallelism needed is much smaller than the length of the LFSR for long BCH encoders. Besides, additional clock cycles are required to pad zeros to the input in order to derive the same final state for the parities. This paper first proposes a new parallel design that allows the input to be added to any tap without sacrificing the throughput. Then the matrices involved in parallel long BCH encoders are analyzed to identify the input tap leading to minimum complexity. Besides, hardware unit sharing for matrix multiplications is enabled through analyzing the timing of the encoder. For a 32-parallel (8191, 7684) BCH encoder, our design achieves 57% higher efficiency in terms of throughput/area ratio compared to the best prior design.**

I. Introduction

BCH codes are broadly used for error correction in digital communications and storage, such as optical communications and Flash memories. To achieve the target reliability with low redundancy, long BCH codes are needed. For example, the BCH codes used in Flash memories are at least 1K-byte long. BCH encoding can be implemented by linear feedback shift registers (LFSRs), whose length is $n-k$ for an (n,k) code. To achieve GigaBytes/s throughput, a parallelism in the order of tens is necessary.

LFSRs are also used to implement cyclic redundancy check (CRC). By applying look-ahead computations on the state transition equation of serial LFSR, [1] developed a parallel LFSR that consists of two matrix multiplications: one in the feedback loop and one at the input. Although the design in [2], which achieves improvements over [3], interprets LFSRs as infinite impulse response filters, its result is equivalent to parallel state transition on a re-timed version of the filter. The state transition in [1] is transformed in [4] to allow trading off the complexity of feedback matrix multiplication by adding a transformation matrix multiplication at the end. Transformation matrices in various formats have been proposed to achieve different goals, such as minimizing the critical path in the feedback loop [5], reducing the overall gate count [6], and lowering the power consumption and gate count [7]. It was found in [8] that adding the input to the least significant tap (LST) instead of the most significant tap (MST) of the LFSR leads to the simplest design when the parallelism p is larger than or equal to $n-k$. Although the final state is different from those with MST input, CRCs are not affected if the same design is used for both encoding and decoding.

This work is supported in part by Kioxia Corporation

The aforementioned designs are mainly for CRCs, whose LFSRs are usually short, such as 32 or smaller. The designs in [5]–[7] rely on exhaustive search to find the optimal transformation matrix, whose dimension is $(n-k)\times(n-k)$. The search complexity becomes prohibitive for long BCH codes. Besides, the feedback matrix has fewer dense columns when $p<(n-k)$ and hence transformation may actually increase the complexity. In addition, to adopt the LST-input design [8] for long BCH encoding, $n-k$ '0's need to be padded to the input to restore the same final state for the parities. This reduces the throughput by a factor of $(n-k)/n$.

Parallel long BCH encoders are derived by applying the unfolding technique in [9], [10]. The generator polynomial specifying the taps of the LFSR is modified to eliminate the large fanout issue in these architectures. Nevertheless, the iteration bound, which decides the critical path, increases linearly with the parallelism.

This paper investigates the design of parallel long BCH encoders based on state transition and explores alternative taps to add the input. A post-processing scheme is proposed to handle the zero padding and eliminate the throughput penalty. Taking into account that the parallelism is much smaller than the length of the LFSR in these encoders, the formats of the involved matrices are re-analyzed. The best tap for input insertion that minimizes the matrix complexities is identified. Moreover, by analyzing the timing of the encoder, it is discovered that hardware units can be shared for matrix multiplications and the logic complexity is further reduced to almost a half. For a 32-parallel (8191,7684) BCH encoder, our proposed design achieves 57% improvement on throughput/area ratio compared to the best existing design.

This paper is organized as follows. Section II introduces prior state-transition parallel LFSR designs. Section III presents our proposed long BCH encoder. Complexity analyses are done in Section IV and conclusions follow in Section V.

II. State-transition parallel LFSR architectures

Let the generator polynomial of BCH code or CRC be $g(x)=x^{n-k}+g_{n-k-1}x^{n-k-1}+\cdots+g_0$. Let the input

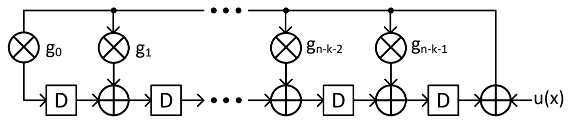

Fig. 1. Serial LFSR architecture with input added to the MST

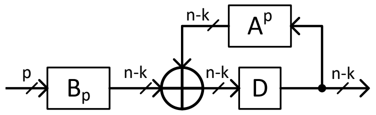

Fig. 2. Parallel LFSR architecture based on state transition

polynomial be $u(x)$. BCH encoding computes the remainder of $(u(x)x^{n-k})/g(x)$ denoted by $Rem(u(x)x^{n-k})_{g(x)}$. A serial LFSR for implementing this division is shown in Fig. 1. $u(x)$ is sent in serially starting from the most significant bit. The remainder is located in the registers after the last bit is input. This architecture can be also used to implement CRC.

Represent the input and state of the registers at clock cycle t by $u(t)$ and $\mathbf{r}(t) = [r_{n-k-1}(t), r_{n-k-2}(t), \cdots, r_0(t)]'$, respectively. Here '$'$' means transpose. Then a state transition equation can be derived from Fig. 1 as

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

where $\mathbf{b} = [g_{n-k-1}, \cdots, g_1, g_0]'$ and

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

It can be derived that

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

where $\mathbf{B}_p = [\mathbf{A}^{p-1}\mathbf{b}, \cdots, \mathbf{A}\mathbf{b}, \mathbf{b}]$ and $\mathbf{u}_p(t) = [u(t), \cdots, u(t+p-2), u(t+p-1)]'$ [1]. Hence, a p-parallel LFSR can be implemented according to Fig. 2. In [4], by using a transformed state vector $\mathbf{r}(t) = \mathbf{T} \times \mathbf{r}_T(t)$, where $\mathbf{T}$ is a non-singular matrix, the state equation is transformed to $\mathbf{r}_T(t+p) = \mathbf{A}_{pT} \times \mathbf{r}_T(t) + \mathbf{B}_{pT} \times \mathbf{u}_p(t)$ with $\mathbf{A}_{pT} = \mathbf{T}^{-1}\mathbf{A}^p\mathbf{T}$ and $\mathbf{B}_{pT} = \mathbf{T}^{-1}\mathbf{B}_p$. $\mathbf{T}$ in different formats have been proposed in [5]–[7] to achieve various optimization goals.

In [8], it was analyzed that when the input is added to the jth tap ($j < n-k$) of the LFSR, the same state transition in (3) still applies, except that $\mathbf{B}_p$ is replaced by $\mathbf{B}_p^{(j)} = [\mathbf{A}^{p-1}\mathbf{b}^{(j)}, \cdots, \mathbf{A}\mathbf{b}^{(j)}, \mathbf{b}^{(j)}]$, where $\mathbf{b}^{(j)}$ is an all-'0' vector with a single '1' at the jth bit. When $p \geq n-k$, $\mathbf{B}_p^{(j)}$ has the lowest complexity when $j = 0$, since $\mathbf{b}^{(0)} = [00\cdots01]$ and hence the rightmost $n-k$ columns of $\mathbf{B}_p^{(0)}$ are an identity matrix. Also the state transformation may further help to reduce the complexity when $p > n-k$. $j = 0$ means that $u(x)$ is added instead to the LST, which is the leftmost tap of the LFSR in Fig. 1. In this case, $Rem(u(x))_{g(x)}$ is in the registers after the last bit of $u(x)$ is sent in. To get the same remainder for BCH encoding, $n-k$ '0's need to be padded to $u(x)$, which costs $\lceil (n-k)/p \rceil$ extra clock cycles.

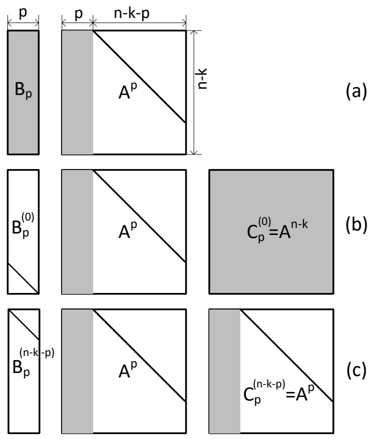

Fig. 3. Formats of the matrices involved in p-parallel LFSR when $p < n-k$. (a) input added to MST; (b) input added to the LST with post processing; (c) input added to the $(n-k-p)^{th}$ tap with post processing

III. Simplified parallel long BCH encoder based on state transition

For parallel long BCH encoders, p is much smaller than $n-k$. For example, $p = 32$ or larger can be employed to achieve throughput in the range of GigaByte/s using modern CMOS processes. However, even for a high-rate (8191,7684) BCH code, $n-k = 507 \sim 16p$. This is contrary to the case of CRCs, which typically have $n-k \leq 32$. As a result, the matrices involved in the state transition have different formats, and hence the design of parallel LFSRs for long BCH encoders needs to be re-evaluated. Moreover, in the case that the input is added to the LST, the $\lceil (n-k)/p \rceil$ extra clock cycles for padding '0's become significant when $p \ll (n-k)$. For the (8191,7684) code, the LST-input design reduces the throughput by 6.2%. Although the throughput penalty can be eliminated by using a higher parallelism, complicated input buffer is needed to regroup the bits according to the parallelism. Alternative methods need to be developed to overcome this penalty.

The 0 column of the companion matrix $\mathbf{A}$ in (2) consists of the coefficients of the BCH generator polynomial. For long BCH codes, around half of these coefficients are nonzero. From $\mathbf{A}$, it can be derived that columns 0 through $n-k-2$ of $\mathbf{A}^i$ are columns 1 through $n-k-1$ of $\mathbf{A}^{i+1}$. Let the column 0 of $\mathbf{A}^i$ be $[a_{n-k-1}, \cdots, a_1, a_0]'$. Then the column 0 of $\mathbf{A}^{i+1}$ is (a_{n-k-1} AND $[g_{n-k-1}, \cdots, g_1, g_0]'$) XOR $[a_{n-k-2}, \cdots, a_0, 0]'$. The rightmost $n-k-1$ columns of $\mathbf{A}$ is an identity with a zero row on the bottom. Hence, for $i < n-k$, the rightmost $n-k-i$ columns of $\mathbf{A}^i$ is an identity with zeros on the bottom and only the first i columns are dense. It can

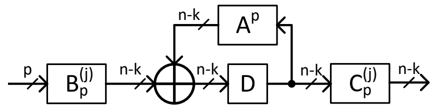

Fig. 4. p-parallel j^{th} tap-input long BCH encoder architecture with post processing to eliminate the throughput penalty

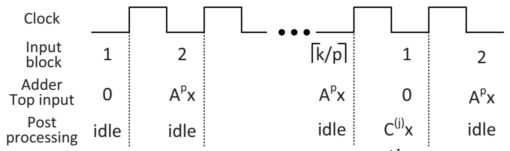

Fig. 5. Timing of computations in the p-parallel j^{th} tap-input long BCH encoder architecture with post processing

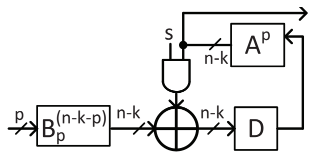

Fig. 6. Simplified p-parallel long BCH encoder architecture with post processing to eliminate the throughput penalty

be also derived that for a vector $\mathbf{y} = [y_{n-k-1}, \cdots, y_1, y_0]'$,

$$\begin{aligned}\mathbf{A}\mathbf{y} =& (y_{n-k-1} \text{ AND } [g_{n-k-1}, \cdots, g_1, g_0]') \\ & \text{XOR } [y_{n-k-2}, \cdots, y_0, 0]'.\end{aligned} \tag{4}$$

Since $\mathbf{b}$ is dense, from (4), each of the columns in $\mathbf{B}_p = [\mathbf{A}^{p-1}\mathbf{b}, \cdots, \mathbf{A}\mathbf{b}, \mathbf{b}]$ is dense. Fig. 3(a) shows the formats of the $\mathbf{A}^p$ and $\mathbf{B}_p$ matrices for the MST-input parallel LFSR architecture when $p \ll n-k$. The shaded areas denote the dense parts. The diagonal line indicates '1's, and the white spaces correspond to '0's. Larger dense portion translates to more logic gates in the matrix multiplication implementation.

Unlike that for CRC, it is very difficult to apply the transformation technique introduced in [4] to reduce the complexity of parallel long LFSRs. When $p \ll n-k$, a very large part of $\mathbf{A}^p$ is identity or zero as shown in Fig. 3(a). Finding a $\mathbf{T}$ such that $\mathbf{A}_{pT} = \mathbf{T}^{-1}\mathbf{A}^p\mathbf{T}$ is less dense than $\mathbf{A}^p$ is already very challenging. Besides, $\mathbf{T}$, whose dimension is $(n-k)\times(n-k)$, needs to be multiplied at the end in the transformed encoder. Additionally, due to the large dimension, it is infeasible to carry out exhaustive search to find an optimized $\mathbf{T}$, even if it is structured. Hence, transformed design is not further considered for parallel long BCH encoders.

When the input is added to the j^{th} tap of the LFSR, $n-k-j$ '0's need to be padded to the input in order to derive the same parities for BCH encoding. From (1), when the input bit is '0', then $\mathbf{r}(t+1) = \mathbf{A} \times \mathbf{r}(t)$. Hence, padding a '0' to the input can be equivalently done as multiplying the register state vector by $\mathbf{A}$. Similarly, padding i '0's can be handled as multiplying the state vector by $\mathbf{A}^i$. As a result, the extra clock cycles needed for handling the $n-k-j$ padded '0's can be eliminated by multiplying a post-processing matrix $\mathbf{C}^{(j)} = \mathbf{A}^{n-k-j}$ to the register state after the last bit of $u(x)$ is sent in as shown in the encoder architecture in Fig. 4.

For CRC, which does not need '0's padded to the input, it was found in [8] that the LFSR complexity is the lowest when the input is added to the LST. In this case, $\mathbf{b}^{(0)} = [00\cdots01]$ and the $\min\{n-k, p\}$ rightmost columns in $\mathbf{B}_p^{(0)}$ are shifted versions of $\mathbf{b}^{(0)}$. The $\mathbf{A}^p$ matrix in the feedback loop does not change with the input tap. However, $\mathbf{C}^{(0)} = \mathbf{A}^{n-k}$ is multiplied at the end to handle the $n-k$ '0's need to be padded to the input in the case of BCH encoding. The three matrices for the LST-input long BCH encoder with post processing to eliminate the throughput penalty are illustrated in Fig. 3(b). Since $\mathbf{A}^{n-k}$ is a dense matrix, its multiplication leads to very high overhead.

Actually, from (4), any $\mathbf{B}_p^{(j)} = [\mathbf{A}^{p-1}\mathbf{b}^{(j)}, \cdots, \mathbf{A}\mathbf{b}^{(j)}, \mathbf{b}^{(j)}]$ for $j \leq n-k-p$ consists of p consecutive columns of an $(n-k)\times(n-k)$ identity matrix. Multiplying any one of these $\mathbf{B}_p^{(j)}$ matrices does not require any logic gates as in the case of multiplying $\mathbf{B}_p^{(0)}$ in the LST-input design. On the other hand, the smaller $n-k-j$, the fewer dense columns $\mathbf{C}^{(j)}$ has. Considering this, the complexity of the proposed long BCH encoder is minimized when j is set to $n-k-p$, which means that the input is added to the $(n-k-p)^{th}$ tap of the LFSR. The formats of the matrices in this case are shown in Fig. 3(c). Compared to the LST-input design, the number of dense columns in the matrices is reduced by a factor of $(p+n-k)/(2p)$.

Another advantage of adding the input to the $(n-k-p)^{th}$ tap is that the feedback matrix $\mathbf{A}^p$ and the post-processing matrix $\mathbf{C}^{(n-k-p)}$ are the same. This enables hardware sharing. The timing of the encoding process is shown in Fig. 5. The registers should be initialized to zero when the first block of p input bits arrive. Hence, effectively, the top adder input in Fig. 4 is zero in this clock cycle. In the rest clock cycles, the register contents are multiplied with $\mathbf{A}^p$ and the product is added to the input bits. The post-processing matrix multiplication is only active for one clock cycle right after the last input block is sent in. From Fig. 5, it can be observed that the multiplications of $\mathbf{A}^p$ and $\mathbf{C}^{(j)}$ are not needed at the same time. As a result, when $j = n-k-p$ and hence $\mathbf{C}^{(j)} = \mathbf{A}^p$, a single unit can be shared to implement both multiplications. Accordingly, the long BCH encoder is simplified to that in Fig. 6. The 's' signal is set to '0' when the first input block arrives and is '1' to pass the $\mathbf{A}^p$ multiplication result in the other clock cycles. In the clock cycle right after the last input block is sent in, the parities are available from the $\mathbf{A}^p$ multiplication output. Since the $\mathbf{B}_p^{n-k-p}$ multiplication does not require any logic gate, the simplified design in Fig. 6 reduces the complexity by almost a half compared to the MST-input design in Fig. 2 or the design in Fig. 4 with $j = n-k-p$.

In the MST-input design of Fig. 2, the $\mathbf{B}_p$ matrix is the same as the dense part of $\mathbf{A}^p$. However, from the timing of the encoder, the multiplications with $\mathbf{B}_p$ and $\mathbf{A}^p$ are active in every clock cycle. Hence, these two multiplications can not share the same hardware as in our proposed design.

TABLE I
COMPARISONS OF (8191,7684) STATE-TRANSITION BCH ENCODERS WITH p=32

	# of XORs in pre-proc. matrix mult.	# of XORs in feedback matrix mult.	# of XORs in post-proc. matrix mult.	total # of XORs	# of gates in critical path
Simplified (Fig. 6)	0 ($\mathbf{B}_p^{(n-k-p)}$)	8191 ($\mathbf{A}^p$)	none	8952	7
MST-input (Fig. 2)	7716 ($\mathbf{B}_p$)	8191 ($\mathbf{A}^p$)	none	16414	6
LST-input (Fig. 4)	0 ($\mathbf{B}_p^{(0)}$)	8191 ($\mathbf{A}^p$)	128373 ($\mathbf{C}_p^{(0)}$)	137071	10

IV. HARDWARE COMPLEXITY ANALYSES AND COMPARISONS

In this section, the complexities of different parallel long BCH encoders based on state transition are analyzed for an example code. The proposed simpified encoder is also compared with the long BCH encoder in [10] derived from unfolding.

Consider the encoder of a 39-error-correcting (8191, 7684) BCH code over $GF(2^{13})$ with $p = 32$. The generator polynomial is $x^{507} + x^{506} + x^{503} + \cdots + x^6 + x^2 + 1$. In our proposed design, the matrix densities are minimized by adding the input to the $(n - k - p)^{th} = 475^{th}$ tap. In this case, the simplified parallel encoder architecture in Fig. 6 can be adopted. It implements the multiplication of only one matrix, $\mathbf{A}^p$, which has p dense columns, and the $\mathbf{B}_p^{(n-k-p)}$ multiplication does not require any logic gates. For a matrix multiplication, the number of XOR gates required can be estimated as the sum of the numbers of '1's in each row minus one. The complexity of the proposed encoder is listed in Table I. The overall XOR gate count includes the contributions of the adder and AND gates in Fig. 6. There are at most 25 '1's in each row of $\mathbf{A}^p$. Hence, the critical path of this matrix multiplication consists of 5 gates, and that of the overall encoder has 7 gates.

The complexities of the MST and LST-input designs are also listed in Table I. Due to the non-trivial $\mathbf{B}_p$ matrix, the MST-input encoder has 16414/8952-1=83% larger area, although the critical path has one less gate. In terms of throughput/area ratio, the simplified design is $(16414 \times 6)/(8952 \times 7) - 1 = 57\%$ better. The LST-input design has a large and dense post-processing matrix $\mathbf{C}^{(0)}$, although its multiplication is only active for one clock cycle for encoding each word. Overall, the gate count is 137071/8952=15.3 times compared to that of the simplified design with input added to the $(n-k-p)^{th}$ tap. Moreover, the rows of $\mathbf{C}^{(0)}$ have up to 283 '1's, and hence the critical path has $\lceil \log_2 283 \rceil + 1 = 10$ gates.

Substructure sharing can be utilized to reduce the gate count of matrix multiplications at the cost of potential critical path increase [7]. Since the dense columns are derived from shifting and XORing with the generator polynomial coefficients, the relative complexities of the three encoders would be similar after substructure sharing.

The numbers of dense columns in $\mathbf{B}_p$ and $\mathbf{A}^p$ are both p. Hence, for a different parallelism, the total gate counts of the simplified and MST-input encoders change proportionally, while their relative complexities remain about the same. The $\mathbf{C}^{(0)}$ matrix does not change with p, and it accounts for the majority of the LST-input encoder complexity. Therefore, for a larger p, the complexity of the LST-input encoder is increased by a smaller portion. For a code with larger $n-k$, the relative complexities of the simplified and MST-input designs also remain about the same. Nevertheless, the complexity overhead of the LST-input encoder will be even more significant.

TABLE II
COMPARISONS OF (8191,7684) BCH ENCODERS

		gate count (# of XORs)	critical path (# of gates)	# of clks per codeword
$p = 32$	simplified (Fig. 6)	8952 (72%)	7	241
	unfolded [10]	12512 (100%)	15	244
$p = 16$	simplified (Fig. 6)	4821 (88%)	6	481
	unfolded [10]	5469 (100%)	8	484

The proposed simplified encoder is also compared to the best encoder based on unfolding from [10] in Table II for different p. To eliminate the large fanout issue caused by unfolding, the LFSR in the design of [10] implements the division of a modified generator polynomial. Another two steps are necessary to incorporate the modification. As a result, it requires larger area than the proposed design and a few extra clock cycles. The normalized areas in terms of percentages are listed in Table II. Besides, the iteration bound, which is the lower bound of the critical path, increases proportionally with p in an unfolded architecture. On the other hand, the critical path of our design has at most $\lceil \log_2 p \rceil + 2$ gates. Hence, our design has much shorter critical path than that of the unfolded encoders and the advantage is more significant for larger p.

V. CONCLUSION

This paper proposed a high-speed simplified parallel architecture based on state transition for long BCH encoders. A post-processing technique is developed to allow the input to be added to any tap of the LFSR without penalizing the throughput. By analyzing the matrices involved in the encoding, the optimal tap for adding the input is found to not only minimize the number of nonzero entries in the matrices but also enable the sharing of hardware units for matrix multiplications. The proposed design achieves substantial complexity reduction compared to prior designs. Future research will investigate possible transformations on the state transition.

VI. ACKNOWLEDGMENT

The author would like to acknowledge Dr. Hironori Uchikawa for the discussions on BCH encoders.

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] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] X. Zhang, "A low-power parallel architecture for linear feedback shift registers," *IEEE Trans. on Circuits and Syst.-II*, vol. 66, no. 3, pp. 4112-416, Mar. 2019.

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

[9] 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.

[10] 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.