An Efficient Parallel Architecture for Resource-Shareable Reed-Solomon Encoder

Yok Jye Tang
Dept. of Electrical & Computer Engineering
The Ohio State University
Columbus, OH 43210 U.S.A.
tang.1121@osu.edu

Xinmiao Zhang
Dept. of Electrical & Computer Engineering
The Ohio State University
Columbus, OH 43210 U.S.A.
zhang.8952@osu.edu

***Abstract*—Reed-Solomon (RS) codes are adopted in many digital communication and storage systems to ensure data reliability. For many of these systems, the encoder and decoder are not active at the same time. In previous designs, RS encoders implemented as linear feedback shift registers in a concatenated structure are reused to compute the syndromes so that the decoder complexity is reduced. However, the parallel versions of such encoders have very long critical path and hence can not achieve high speed. This paper proposes a new parallel resource-shareable RS encoder architecture based on the Chinese Remainder Theorem (CRT). The generator polynomial of RS codes is decomposed into factors of degree one and state transformation is developed to enable the sharing of the hardware units for syndrome computation. As a result, the critical path is reduced to only one multiplier and one adder, regardless of the parallelism. Additionally, by utilizing the property that the degrees of the decomposed polynomial factors are one, optimizations are also developed to greatly simplify the CRT-based encoder. For example encoders of a (255, 229) RS code over $GF(2^8)$, our proposed design can achieve at least 29% higher efficiency in terms of area-time product for moderate or higher parallelisms compared to the previous resource-shareable RS encoder and traditional parallel RS encoders combined with syndrome computation units that implement the same functionality.**

***Index Terms*—Chinese Remainder Theorem, Encoder, Reed-Solomon codes, Syndrome computation.**

I. Introduction

Reed-Solomon (RS) error-correcting codes have been used in numerous communication and storage systems, such as mobile communication, magnetic recording, digital video broadcasting, and deep-space probing. In many of these systems, encoder and decoder will not be active at the same time. In this case, the encoder can be re-designed [1] so that it is shared to compute the syndromes, which is the first step of RS decoding and accounts for a significant portion of the decoder area.

BCH codes are subfield subcodes of RS codes. Resource-shareable encoders have also been investigated for binary BCH codes. In the design of [2], the generator polynomial of BCH code is decomposed to multiple short minimal polynomials and the concatenated linear feedback shift register (LFSR) structure from [3] is utilized to compute the remainders over each minimal polynomial. Then the complexity of syndrome computation is reduced by evaluating the short remainders over finite field elements. However, this design has large iteration bound, which limits the achievable critical path and hence the maximum clock frequency. Although the iteration bound can be reduced by modifying the encoding scheme [4], the resulted parities are not protected. In the case of RS codes, the concatenated LFSR structure in the design of [2] becomes very similar to that in [1].

Parallel encoders are required to achieve high speed. If unfolding is applied, the iteration bound in the resulted p-parallel architecture will be increased to p times. Although register state look-ahead [5]–[9] can be applied to a single LFSR to derive parallel architectures, the resource-shareable encoder of [1] concatenates multiple feedback loops and each feedback loop takes inputs from previous feedback loops. As a result, the register state look-ahead computation can not be easily extended to this case.

This paper proposes an efficient parallel architecture for resource-shareable RS encoder that can implement both RS encoding and syndrome computation. The proposed design is based on the Chinese Remainder Theorem (CRT). According to the CRT, the encoder has three stages of computation. Different from that for BCH codes, the generator polynomial of RS codes is decomposed into factors of degree one. In this case, the simple feedback loops in the second stage lead to a very short critical path of only one multiplier and one adder, regardless of the parallelism. Also state transformation is developed to enable the sharing of the hardware units for syndrome computation. Additionally, utilizing the property that each decomposed factor has degree one, the first stage of the CRT computation is replaced by constant scalar multiplications, which can be integrated with the other computations, and the third stage is greatly simplified. Unlike the design in [4], the parities are also protected in our encoding scheme. For an example RS (255, 229) code over $GF(2^8)$, our proposed design has much shorter critical path and achieves dozens of times higher efficiency in terms of area-time product (ATP) than the parallel encoder achieved by unfolding the resource-shareable design in [1] for moderate or higher parallelisms. Compared to the best traditional parallel LFSR designs from [5], [9] for RS encoding combined with syndrome computation units that implement the same functionality, our resource-

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

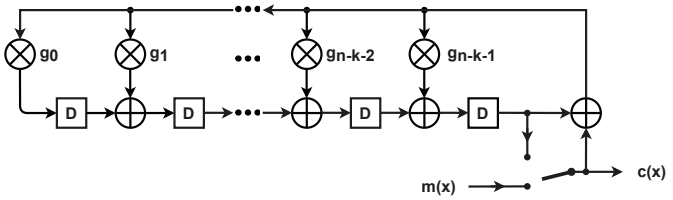

Fig. 1. Serial LFSR for (n,k) RS encoding

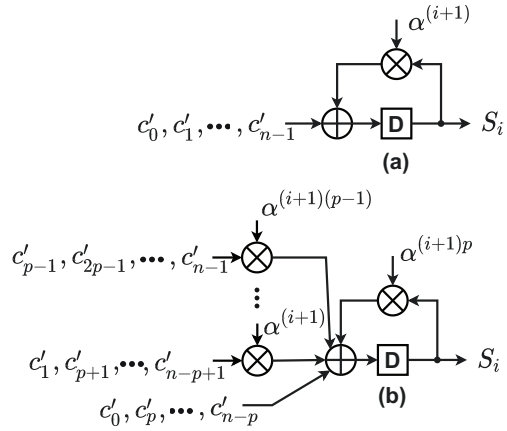

Fig. 2. Syndrome computation architectures: (a) serial; (b) p-parallel

shareable design can achieve at least 29% higher efficiency.

The structure of this paper is as follows. Section II introduces RS encoding and previous work on resource-shareable encoder design. The proposed parallel resource-shareable RS encoding scheme is detailed in Section III. Section IV provides complexity analysis for encoders with general parameters. More detailed complexity analysis and comparisons are carried out in Section V for an example code. Conclusions follow in Section VI.

II. Resource Shareable Reed-Solomon Encoder

For a narrow-sense (n, k) RS code with $t = (n-k)/2$ error-correction capability, the generator polynomial is constructed as

$$g(x) = (x + \alpha^1)(x + \alpha^2) \cdots (1 + \alpha^{2t}), \tag{1}$$

where α is a primitive element of $GF(2^q)$ and $n \leq 2^q - 1$. For systematic RS encoding, a k-symbol message, $m(x)$, is encoded into a n-symbol codeword $c(x) = m(x)x^{2t} + \text{Rem}_{g(x)}(m(x)x^{2t})$, where $\text{Rem}_{g(x)}(m(x)x^{2t})$ is the remainder polynomial of dividing $m(x)x^{2t}$ by $g(x)$. Rewrite the generator polynomial as $g(x) = x^{2t} + g_{2t-1}x^{2t-1} + \cdots + g_1x^1 + g_0$, where $g_{2t-1}, \cdots, g_1, g_0 \in GF(2^q)$. The serial LFSR in Fig. 1 computes $\text{Rem}_{g(x)}(m(x)x^{2t})$ by taking in the coefficients of $m(x)$ serially starting from the most significant one. After the last coefficient is processed, the coefficients of $\text{Rem}_{g(x)}(m(x)x^{2t})$ are located in the registers. To allow these coefficients to be serially shifted out in the next $2t$ clock cycles, the output of the right-most register is connected to the encoder output through the switch in Fig. 1.

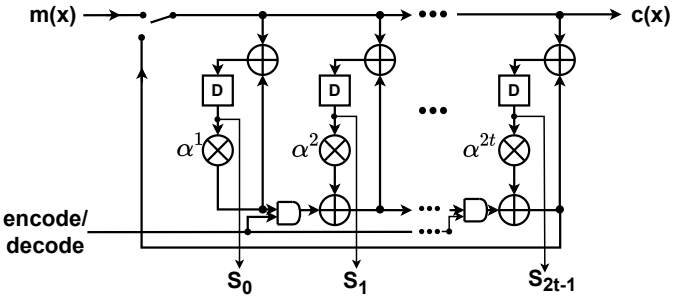

Fig. 3. Resource-shareable RS encoder with concatenated feedback loops

Let $c'(x) = c'_{n-1}x^{n-1} + \cdots + c'_1x^1 + c'_0$ be the received word. Syndromes defined as $S_i = c'(\alpha^{i+1})$ $(0 \leq i \leq 2t-1)$ need to be calculated for RS decoding. By applying Horner's rule, $S_i = (\cdots(c'_{n-1}\alpha^{i+1} + c'_{n-2})\alpha^{i+1} + \cdots)\alpha^{i+1} + c'_0$. Hence, it can be computed by using the serial architecture in Fig. 2 (a). By grouping p consecutive coefficients of $c'(x)$, the p-parallel syndrome computation architecture in Fig. 2(b) can be derived to compute S_i in $\lceil n/p \rceil$ clock cycles.

For RS codes, $g(x)$ consists of $2t$ factors in the format of $x + \alpha^{i+1}$ as shown in (1), where $0 \leq i < 2t$. Accordingly, by utilizing the z-transform, the LFSR in Fig. 1 can be converted to the structure in Fig. 3 that consists of multiple concatenated feedback loops [1]. The i-th feedback loop in Fig. 3 happens to have the same constant multiplier as that needed in the architecture for computing S_i shown in Fig. 2(a). Hence, by adding the AND gates and switching the lower inputs of the AND gates between '1' and '0', the architecture in Fig. 3 implements RS encoding and syndrome computation, respectively. For architecture with feedback loops, the minimum achievable critical path is lower bounded by the iteration bound T_∞, which is defined as the maximum of the loop bounds. The bound of a loop is the computation time of the loop divided by the number of registers in the loop [10]. For the architecture in Fig. 3, the α constant multiplier contributes to one XOR gate in the data path [11]. Due to the long chain of XOR and AND gates at the bottom, the iteration bound of this architecture is $T_\infty = (2t+1)T_{XOR} + (2t-1)T_{AND}$, where T_{XOR} and T_{AND} are the propagation delays of an XOR gate and an AND gate, respectively. Hence, its achievable critical path is long. Unlike that in the resource-shareable BCH encoder [2], the feedback loops in RS encoders can not be combined to shorten the chain of XOR and AND gates since the individual feedback loops themselves are also used to compute the syndromes.

A p-parallel version of the RS encoder in Fig. 3 can be derived by the unfolding technique [10]. However, the iteration bound in the p-unfolded architecture will be pT_∞. Parallel designs of a single LFSR, such as the one in Fig. 1, can be derived by applying look-ahead computation to the register state [5]–[9]. However, different from that in the single LFSR, the output of the rightmost XOR gate on the bottom of the architecture in Fig. 3 that is fed back to every register includes the contributions of every register in the previous clock cycle. Due to this reason, the look-ahead register state computation

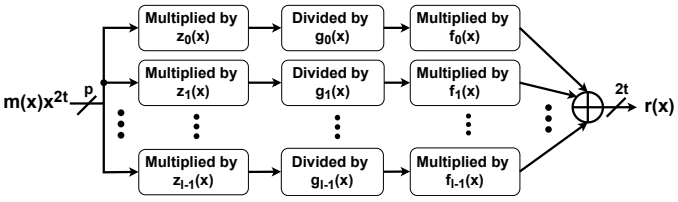

Fig. 4. Architecture for CRT-based remainder computation

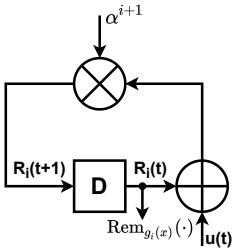

Fig. 5. Architecture for computing the remainder of the division by $g_i(x) = (x + \alpha^{i+1})$

can not be directly applied to the RS encoder in Fig. 3 to derive parallel designs.

III. CRT-Based Resource-Shareable RS Encoder

The CRT decomposes the division by a longer polynomial to the divisions by individual factors of the polynomial. Intuitively, this may help to disconnect the feedback loops that are concatenated in the RS encoder architecture in Fig. 3 and reduce the iteration bound. This section proposes a parallel CRT-based RS encoder architecture that supports both encoding and syndrome computation. Each feedback loop in the proposed architecture only consists of one multiplier and one adder. As a result, the iteration bound and hence the achievable critical path is substantially reduced. Additionally, by making use of the property that the generator polynomials of RS codes are decomposed into factors with degree one, optimizations are developed to greatly simplify the involved computations.

Suppose that $g(x)$ can be decomposed into l polynomials as $g(x) = g_0(x)g_1(x)\cdots g_{l-1}(x)$, where $g_i(x)$ and $g_j(x)$ for any $i \neq j$ do not have any common factor except 1. Let $f_i(x) = g(x)/g_i(x)$ $(0 \leq i < l)$. By using the Euclidean algorithm, for each $f_i(x)$, a polynomial $z_i(x)$ satisfying $z_i(x)f_i(x) \equiv 1 \bmod g_i(x)$ and $0 \leq \deg(z_i(x)) < \deg(g_i(x))$ can be derived. Here $\deg(\cdot)$ denotes the degree of the polynomial. Then, according to the CRT, the $\text{Rem}_{g(x)}(m(x)x^{2t})$ needed for RS encoding can be computed as

$$\text{Rem}_{g(x)}(m(x)x^{2t}) = \sum_{i=0}^{l-1} f_i(x)\text{Rem}_{g_i(x)}(z_i(x)m(x)x^{2t}). \tag{2}$$

Let $r(x) = \text{Rem}_{g(x)}(m(x)x^{2t}) = r_{2t-1}x^{2t-1} + \cdots + r_1x^1 + r_0$. The block diagram of the architecture that implements such computations is shown Fig. 4. It consists of three stages: 1) multiply the input with each $z_i(x)$; 2) divide the output of the first stage by each $g_i(x)$ to get the remainder; 3) multiply each remainder from the second stage by $f_i(x)$ and sum up all the products.

The generator polynomial of a RS code is a product of $(x + \alpha^{i+1})$ for $i = 0, 1, \cdots, 2t-1$ as shown in (1). Hence, to apply the CRT formula in (2), $l = 2t$ and $g_i(x) = (x + \alpha^{i+1})$. The first stage of the CRT-based RS encoding is to multiply $m(x)x^{2t}$ with $z_i(x)$. Since $\deg(g_i(x)) = 1$, $z_i(x)$ is a constant that can be pre-computed from the Euclidean algorithm. Assume that $z_i(x) = \beta_i$. The multiplication with β_i can be put off and combined with the constant multiplications needed in the third stage of the CRT calculation.

In the case that $g_i(x) = x + \alpha^{i+1}$, the division by $g_i(x)$ for the second stage of the CRT calculation can be implemented by the simple feedback loop shown in Fig. 5. This is a serial architecture. Denote the input and state of the register in clock cycle t as $u(t)$ and $\text{R}_i(t)$, respectively. From the feedback loop, it can be derived that

$$\text{R}_i(t+1) = \alpha^{i+1}\text{R}_i(t) + \alpha^{i+1}u(t).$$

Substituting this equation back to itself for p iterations [5], the state of the register after p clock cycles is

$$\text{R}_i(t+p) = \alpha^{(i+1)p} \times \text{R}_i(t) + \mathbf{B}_{p,i} \times \mathbf{u}_p(t), \tag{3}$$

where $\mathbf{B}_{p,i} = [\alpha^{(i+1)p}, \cdots, \alpha^{(i+1)2}, \alpha^{i+1}]$ and $\mathbf{u}_p(t) = [u(t), \cdots, u(t+p-2), u(t+p-1)]^\intercal$, where '$\intercal$' denotes transpose. From (3), a p-parallel architecture for the division by $(x + \alpha^{i+1})$ can be derived.

The p-parallel syndrome computation architecture in Fig. 2(b) can be also described by register state update. Denote the inputs to this architecture in clock cycle t also by $\mathbf{u}_p(t)$. Following the computations carried out in Fig. 2(b), it can be observed that

$$\text{R}_i(t+p) = \alpha^{(i+1)p} \times \text{R}_i(t) + \mathbf{B}'_{p,i} \times \mathbf{u}_p(t), \tag{4}$$

where $\mathbf{B}'_{p,i} = [\alpha^{(i+1)(p-1)}, \cdots, \alpha^{(i+1)}, \alpha^0]$.

Comparing (3) and (4), the only difference lies in the matrices multiplied to $\mathbf{u}_p(t)$. Besides, $\mathbf{B}_{p,i} = \alpha^{i+1}\mathbf{B}'_{p,i}$. To enable the sharing of hardware units for implementing the division by $(x + \alpha^{i+1})$ and S_i syndrome computation, the technique of state transformation that was originally proposed for the LFSR for cyclic redundancy check in [9] can be extended and applied to (3). The register state can be transformed to $\text{R}_i(t) = T_i \times \text{R}_{T,i}(t)$, where $T_i = \alpha^{i+1}$. Then the formula in (3) becomes

$$\text{R}_{T,i}(t+p) = \alpha^{(i+1)p} \times \text{R}_{T,i}(t) + \alpha^{-(i+1)}\mathbf{B}_{p,i} \times \mathbf{u}_p(t). \tag{5}$$

which is in exactly the same format as (4). Therefore, (4) can be implemented for both division by $(x + \alpha^{i+1})$ and S_i computation, except that $T_i = \alpha^{i+1}$ needs to be multiplied back afterwards to reverse the state transformation in the former case. Nevertheless, since T_i is a constant, its multiplication can be also combined with the third-stage calculation in the CRT-based encoding.

In the third stage of the encoding, the output from the second stage is multiplied with $f_i(x)$. Since the remainder of the

division by $(x+\alpha^{i+1})$ is a single coefficient, this polynomial multiplication is reduced to coefficient-by-coefficient multiplications. Also recall that β_i for the first stage calculation and $T_i = \alpha^{i+1}$ for reversing the state transformation need to be multiplied. Use γ_i to represent the output of the i-th block in the second stage. Then the coefficients of the overall remainder polynomial for RS encoding can be calculated as

$$\begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_{2t-1} \end{bmatrix} = \begin{bmatrix} v_{0,0} & v_{0,1} & \cdots & v_{0,2t-1} \\ v_{1,0} & v_{1,1} & \cdots & v_{1,2t-1} \\ \vdots & \vdots & \ddots & \vdots \\ v_{2t-1,0} & v_{2t-1,1} & \cdots & v_{2t-1,2t-1} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{2t-1} \end{bmatrix} \quad (6)$$

where $v_{j,i} = \beta_i \alpha^{i+1} f_{i,j}$ and $f_{i,j}$ is the coefficient of x^j in $f_i(x)$.

Let V be the $2t \times 2t$ matrix in (6). The proposed p-parallel CRT-based resource-shareable RS encoder is illustrated in Fig. 6. During the encoding process, p coefficients of $m(x)x^{2t}$ are sent as the inputs in each clock cycle starting with the most significant ones. In Fig. 5, by adding the input to the output of the register, the input polynomial is effectively multiplied by x. Hence, only $2t-1$ zeros need to be padded after the least significant coefficient of $m(x)$ to achieve $m(x)x^{2t}$. Block G_i implements the computation in (4). After $\lceil n/p \rceil$ clock cycles, the computations in the G_i blocks ($0 \le i < 2t$) are completed. Then their outputs are multiplied with the constant V matrix in (6) to derive the $2t$ coefficients of $\text{Rem}_{g(x)}(m(x)x^{2t})$. For syndrome computation, the coefficients of $c'(x)$ are sent in. After $\lceil n/p \rceil$ clock cycles, the syndrome S_i is available at the output of the G_i block. Since the multiplication with the V matrix does not have any feedback loops, the iteration bound of the overall encoder in Fig. 6 is decided by the feedback loops in the G_i blocks. From (4), the feedback loop in block G_i only consists of an adder and a multiplier with constant $\alpha^{(i+1)p}$. Therefore, the proposed architecture has much smaller iteration bound and hence can achieve much shorter critical path than the architecture in Fig. 3 and its parallel version.

Comparing Fig. 4 and Fig. 6, our proposed RS encoder is substantially simplified from straight-forward CRT computation. Utilizing the property that the generator polynomial of RS codes is decomposed into factors of degree one, the first stage polynomial multiplication in Fig. 4 is reduced to constant coefficient multiplications. Additionally, these constant multiplications can be combined with the third-stage calculation. Also the outputs of the second stage become single coefficients. As a result, the polynomial multiplications in the third stage of Fig. 4 become coefficient-by-coefficient multiplications. Due to these modifications, the CRT computation is substantially simplified in the case of RS encoding.

CRT-based encoder was previously exploited in [12] for BCH codes. Different from those of RS codes, the generator polynomial of a BCH code is a product of minimal polynomials, whose degrees are typically q for codes constructed over $GF(2^q)$ and can not be further decomposed into smaller factors. In this case, the first and third stages of the CRT-based remainder computation have high complexity. As a result, unlike our proposed CRT-based RS encoder, the CRT-based BCH encoder in [12] has much larger area compared to the encoder utilizing concatenated LFSRs [2], [3], especially for higher parallelisms.

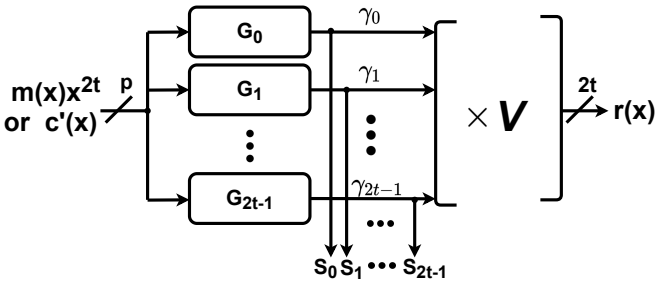

Fig. 6. Proposed p-parallel CRT-based resource-shareable RS encoder architecture

TABLE I
COMPLEXITY AND ITERATION BOUND FOR p-PARALLEL t-ERROR-CORRECTING RS ENCODERS OVER $GF(2^q)$

	[1] unfolded	Proposed
# of const. mult. over $GF(2^q)$	$2tp$	$2tp + (2t)^2$
# of adder over $GF(2^q)$	$p(4t-1)$	$2tp + 2t(2t-1)$
# of q-bit AND gates	$p(2t-1)$	0
# of q-bit registers	$2t$	$2t$
Iteration bounds (# of T_{XOR})	$4tp$	$log_2 q + 1$

IV. Complexity and Iteration Bound Analyses

This section analyzes the complexity of our proposed p-parallel recourse-shareable encoder for a t-error-correcting RS code constructed over $GF(2^q)$.

The complexity and iteration bound of the proposed p-parallel CRT-based RS encoder are summarized in the last column of Table I. In our design, each G_i block implements (4) and the architecture is the same as that in Fig. 2(b). The feedback loop consists of one constant multiplier over $GF(2^q)$, one $GF(2^q)$ adder, and one q-bit register. Additionally, there are $p-1$ constant multipliers and $p-1$ adders outside of the feedback loop. As shown in Fig. 6, the outputs of the G_i blocks are multiplied by the $2t \times 2t$ constant V matrix. This matrix multiplication can be carried out by $(2t)^2$ constant multipliers and $2t(2t-1)$ adders. Over $GF(2^q)$, each adder is implemented by a q-bit XOR and each constant multiplier can be implemented as a $q \times q$ binary constant matrix multiplication. Since only the G_i blocks have feedback loops, the iteration bound of the overall encoder is decided by these feedback loops. A $q \times q$ constant matrix can have at most q '1's in a row. Hence the iteration bound of our proposed encoder is at most $(log_2 q + 1)T_{XOR}$.

For comparisons, the complexity and iteration bound of the p-unfolded version of the serial design in [1] are also listed in Table I. Fig. 3 shows the serial architecture from [1]. It consists of $2t$ constant multipliers, $4t-1$ adders, $2t$ q-bit registers,

TABLE II
COMPLEXITY AND CRITICAL PATH COMPARISONS FOR RESOURCE-SHAREABLE ENCODERS WITH DIFFERENT PARALLELISMS FOR A 13-ERROR-CORRECTING RS (255,229) CODE

p	Design	Encoding (# of XORs)	Syndrome computation (# of XORs)	Total area (normalized)	Critical path (# of gates)	ATP (normalized)	Latency (# of clks)
13	[1] unfolded	13208	-	13208 (1)	676	8928608 (65)	20
	[5]*†	32528	11583	44111 (3.34)	4	176444 (1.29)	22
	[8]†	11706	10959	22665 (1.72)	8	181320 (1.32)	20
	Proposed*	34285	-	34285 (2.60)	4	137140 (1)	22
26	[1] unfolded	25792	-	25792 (1)	1352	34870784 (190)	10
	[5]*†	43199	23151	66350 (2.57)	4	265400 (1.45)	13
	[8]†	22388	21903	44291 (1.72)	8	354328 (1.93)	10
	Proposed*	45853	-	45853 (1.78)	4	183412 (1)	13
52	[1] unfolded	50960	-	50960 (1)	2704	137795840 (510)	5
	[5]*†	64156	44820	108976 (2.14)	4	435904 (1.61)	8
	[8]†	43786	43572	87358 (1.71)	8	698864 (2.59)	5
	Proposed*	67522	-	67522 (1.33)	4	270088 (1)	8

* Pipelining is applied to reduce the critical path

† Extra units are added to enable syndrome computation

and $2t-1$ q-bit AND gates. In a p-unfolded architecture, p times more computation units are needed while the number of registers remains the same [10]. Hence, the p-unfolded architecture of the encoder in Fig. 3 requires $2tp$ constant multipliers, $p(4t-1)$ adders, $2t$ q-bit registers, and $p(2t-1)$ q-bit AND gates. Due to the concatenation, the computations carried out by the XOR and AND gates at the bottom of Fig. 3 can not be implemented in a tree structure. Accordingly, as mentioned in Section II, the iteration bound of this architecture is $T_\infty = (2t+1)T_{XOR} + (2t-1)T_{AND}$, which is around $4tT_{XOR}$. As a result, the iteration bound of the p-unfolded architecture is $pT_\infty = 4tpT_{XOR}$.

The RS codes utilized in practical applications are typically constructed over moderate or smaller finite fields, such as $GF(2^8)$, and have non-trivial error-correcting capability t. In this case, it can be observed from Table I that our proposed design apparently has much lower iteration bound and hence can achieve much shorter critical path compared to the p-unfolded version of the design in [1]. Also unlike the design in [1], the iteration bound of our architecture does not change with p or t. Hence, the speed advantage of our design would be even more significant for higher parallelism and/or codes with higher error-correcting capability.

The complexity of each constant multiplier is dependent on the constant. The multiplication with α^i when i is not small requires more XOR gates than an adder. Hence, the areas of both the proposed design and that from [1] are dominated by the multipliers. For highly parallel design such that p is much larger than t, the area of the proposed encoder would become close to that of the unfolded version of the encoder from [1].

V. COMPLEXITY AND ITERATION BOUND COMPARISONS FOR EXAMPLE RS ENCODERS

To further illustrate the advantages of the proposed resource-shareable RS encoder, this section compares the proposed design with different parallelisms to the unfolded version of the design in [1] for an example (255, 229) RS code over $GF(2^8)$ with $t = (n-k)/2 = 13$. There are no other existing design for resource-shareable RS encoder that can implement both encoding and syndrome computation. Hence, the combination of traditional parallel LFSR RS encoders and syndrome computation units are utilized for further comparison.

The complexities of our proposed design with $p = 13, 26, 52$ are listed in Table II. The numbers of XOR gates are derived according to the actual numbers of '1's in the binary constant matrices for the involved finite field element multiplications. It is also assumed that the area of a register is around 3 times the area of an XOR gate. The iteration bound of our architecture is $4T_{XOR}$ for codes over $GF(2^8)$ regardless of p or t. Since the data paths in the non-feedback parts of G_i and the V matrix multiplication are longer, pipelining registers can be inserted into each part to reduce the critical path to 4 XOR gates. The total area in Table II includes the complexity of the pipelining registers. Similarly, the numbers of XOR gates for the unfolded versions of [1] listed in Table II are also derived according to the actual numbers of '1's in

the corresponding binary constant matrices. In the total area estimation, it is assumed that each AND gate takes 1/2 the area of an XOR gate to implement. For comparisons, the areas of our proposed design are normalized with respect to the areas of the design from [1].

Although the pipelining leads to a few additional clock cycles in the latency, the critical path of our design is only a fraction of that from [1]. Hence, our design can achieve much higher clock frequency and processing speed. The area of our design is larger. However, the difference becomes smaller for larger p. The number of XOR gates needed for a matrix multiplication can be further reduced by applying substructure sharing without changing the critical path [7]. The multiplication of the V matrix with $2t \times 2t$ finite field elements can be converted to the multiplication of a $2tq \times 2tq$ constant binary matrix multiplication. More shareable substructures can be typically found for matrices of larger dimension. Hence, it is expected that area overhead of our design compared to that from [1] would become smaller if substructure sharing is adopted. To compare designs with different area and throughput, the efficiency in terms of ATP can be adopted. It should be noted that although pipelining increases the latency, it does not change the number of clock cycles needed to process each encoding since p data symbols are still processed in each clock cycle. Hence, the ATP used for comparison can be defined as (total area) $\times$ (critical path). It can be observed from Table II, the ATP of the unfolded version of the design from [1] is 65 times higher than the proposed design when $p = t$ and this ratio further increases for larger p.

Parallel designs of binary LFSRs based on state look-ahead are available in [5]–[9]. These architectures consist of an input pre-processing matrix multiplication and a matrix multiplication in a feedback loop. These designs can be extended to implement non-binary LFSRs for RS encoding by replacing the constant finite field elements in the matrices with the corresponding $q \times q$ binary matrices. The designs in [6], [7] try to find a transformation matrix by exhaustive search to reduce the complexities of the pre-processing and feedback matrix multiplications. In the encoders for RS codes over $GF(2^q)$, the sizes of the matrices are increased by q^2 times compared to those for binary LFSRs. Hence, the exhaustive search becomes unfeasible. For RS encoding, the design in [9] does not have any advantage over [8]. Hence, the encoder-only design in [8] combined with units for syndrome computation is compared with the proposed resource-shareable design that implements both encoding and syndrome computation in Table II. The design from [5] combined with syndrome computation units is also included for comparison since it has the shortest critical path among previous designs.

It can be observed from Table II that the proposed encoder achieves the same shortest critical path as the design from [5]. Also the efficiency of our design in terms of ATP is $(1.29 - 1) \times 100 = 29\%$ higher than that of the encoder from [5] combined with syndrome computation units when $p = t$ and the efficiency further improves for larger p. Although our design requires larger area than the design from [8] combined with syndrome computation units when $p = t$, the encoder from [8] has twice longer critical path. As a result, our design is 32% more efficient when $p = t$. Additionally, the area requirement of the design from [8] increases fast and its efficiency becomes much lower for larger p. Overall, it is evident that the proposed design is much more efficient than traditional parallel LFSR RS encoders combined with syndrome computation units that implement the same functionality, especially for larger p.

VI. Conclusion

This paper proposes a new parallel resource-shareable RS encoder based on the CRT. By utilizing the property that the generator polynomial of RS codes can be decomposed to factors of degree one, optimizations are developed to substantially reduce the complexity of the CRT-based encoder. Additionally, state transformation is developed to enable the sharing of the hardware units of the RS encoder for syndrome computation. As a result, the proposed design has much shorter critical path than the previous resource-shareable RS encoder. Compared to traditional parallel LFSR RS encoders combined with syndrome computation units that implement the same functionality, the proposed encoder also achieves much higher efficiency, especially when a larger parallelism is needed. Future work will exploit alternative methods for implementing resource-shareable RS/BCH encoders.

References

[1] G. Fettweis and M. Hassner, "A combined Reed-Solomon encoder and syndrome generator with small hardware complexity," *Proc. of IEEE Int. Symp. on Circuits and Syst.*, San Diego, CA, USA, 1992, pp. 1871-1874.

[2] H. Tang, G. Jung and J. Park, "A hybrid multimode BCH encoder architecture for area efficient re-encoding approach," *Proc. of IEEE Int. Symp. on Circuits and Syst.*, Lisbon, Portugal, 2015, pp. 1997-2000.

[3] H. Yoo, J. Jung, J. Jo and I. Park, "Area-efficient multimode encoding architecture for long BCH codes," *IEEE Trans. on Circuits and Syst.-II*, vol. 60, no. 12, pp. 872-876, Dec. 2013.

[4] A. K. Subbiah and T. Ogunfunmi, "Area-effcient re-encoding scheme for NAND flash memory with multimode BCH error correction," *Proc. of IEEE Int. Symp. on Circuits and Syst.*, Florence, Italy, 2018, pp. 1-5.

[5] J. H. Derby, "High-speed CRC computation using state-space transformations," *Proc. of IEEE Global Telecom. Conf.*, San Antonio, TX, USA, 2001, pp. 166-170.

[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. 412-416, Mar. 2019.

[8] X. Zhang, and Y. J. Tang, "Reducing parallel linear feedback shift register complexity through input tap modification," *Proc. of IEEE Int. Symp. on Circuits and Syst.*, Sapporo, Japan, 2019, pp. 1-5.

[9] X. Zhang, and Y. J. Tang, "Low-complexity parallel cyclic redundancy check," *Proc. of IEEE Int. Symp. on Circuits and Syst.*, May 2021.

[10] K. K Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, John Wiley & Sons, 1999.

[11] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," *IEEE Trans. on VLSI Syst.*, vol. 12, no. 9, pp. 957-967, Sept. 2004.

[12] H. Chen, "CRT-based high-speed parallel architecture for long BCH encoding," *IEEE Trans. on Circuits and Syst.-II*, vol. 56, no. 8, pp. 684-686, Aug. 2009.