A Universal VLSI Architecture for Reed–Solomon Error-and-Erasure Decoders

Hsie-Chia Chang, Member, IEEE, Chien-Ching Lin, Fu-Ke Chang, and Chen-Yi Lee, Member, IEEE

Abstract—This paper presents a universal architecture for Reed–Solomon (RS) error-and-erasure decoder. In comparison with other reconfigurable RS decoders, our universal approach based on Montgomery multiplication algorithm can support not only arbitrary block length but various finite-field degree within different irreducible polynomials. Moreover, the decoder design also features the constant multipliers in the universal syndrome calculator and Chien search block, as well as an on-the-fly inversion table for calculating error or errata values. After implemented with 0.18-μm IP6M technology, the proposed universal RS decoder correcting up to 16 errors can be measured to reach a maximum 1.28 Gb/s data rate at 160 MHz. The total gates count is around 46.4 K with 1.21 mm² silicon area, and the average core power consumption is 68.1 mW.

Index Terms—Error-and-erasure correction, Montgomery multiplication, Reed–Solomon (RS) code, universal architecture.

I. INTRODUCTION

THE Reed–Solomon (RS) code is well acceptable in many storage and digital communication systems for its excellent burst error correction capability. An \((n,k)\) RS code contains \(k\) message symbols and \(n-k\) parity-check symbols and is capable of correcting up to \(t = \lfloor (n-k)/(2) \rfloor\) erroneous symbols. Each symbol over GF\((2^m)\) indicates a \(m\)-bit data. As shown in Fig. 1, RS decoders usually consist of a syndrome calculator, a key equation solver, a Chien search block, and an erasure value evaluator. While correcting both errors and erasures, the RS decoder requires an erasure generator, Forney syndrome calculator, and a polynomial multiplier, which are also illustrated in Fig. 1 as dotted blocks. Note that errata represents either error or erasure during transmission in a noisy channel.

For error-only correction, the key equation shown in Fig. 1 is defined as

\[
S(x)\sigma(x) = \Omega(x) \mod x^{2t}
\]

(1)

where \(S(x)\) is syndrome polynomial, \(\sigma(x)\) is error-locator polynomial, and \(\Omega(x)\) is error-evaluator polynomial [1]. For correcting both errors and erasures, the key equation should be modified to

\[
S(x)\lambda(x)\sigma(x) = \omega(x) \mod x^{2t}
\]

(2)

where \(\lambda(x) = (1-\alpha^i y_1)(1-\alpha^j y_2)\cdots(1-\alpha^l y_u)\) indicates erasure-locator polynomial with \(u\) erasure information \(\alpha^i_1, \alpha^i_2, \ldots, \alpha^l_u\), and \(\omega(x)\) is errata-evaluator polynomial. To perform RS error-and-erasure decoding procedure efficiently, Forney syndrome polynomial and errata-locator polynomial are exploited and denoted as \(T(x) = S(x)\lambda(x)\) and \(\Lambda(x) = \lambda(x)\sigma(x)\), respectively [2].

Although dedicated RS decoder designs have been reported as high-speed or low-power approaches recently [3]–[6], there has been little discussion on RS decoders with configurability or programmability [7]. Nevertheless, more and more communication and storage systems provide different design parameters to meet specific performance requirements. Table I lists several applications for RS codes with different code rates and GF\((2^m)\) definitions. For packet loss protection of multicasting or broadcasting communications, RS codes are utilized as a block erasure coding scheme and specified in DVB-H applications. Thus, it will be much complicated if all dedicated RS decoders are implemented within a single chip.

In this paper, a cost-effective RS decoder that meets various system specifications is proposed. The proposed universal RS decoder can manipulate different code rates and block lengths defined in arbitrary GF\((2^m)\). The difficulty for the universal architecture is to provide finite-field operations in various field degree over different irreducible or primitive polynomials. As to our knowledge, only the software approach was proposed to support various field degree by using programmable digital signal processor [14]. Actually, the universal finite-field multiplier (FFM) can be achieved by Montgomery multiplication algorithm because of the modulo operation with configurable polynomials [15]. To efficiently accommodate different irreducible polynomials, the universal FFM derived from Montgomery multiplications is proposed in Section II.
Then, the universal \((n, k)\) RS decoder over \(GF(2^m)\) is described in Section III. The design example which supports \(n \leq 255, t \leq 16\) for error-only or \(t \leq 8\) for erasure and error correction and arbitrary irreducible polynomials with \(m \leq 8\) is provided. Section IV shows the corresponding chip implementation and measurement results. Finally, Section V gives the conclusion.

II. UNIVERSAL FFM

With polynomial representation, the modular multiplication of \(\hat{a}\) and \(\hat{b}\) in \(GF(2^m)\) can be expressed as

\[
\hat{c} = \hat{a} \cdot \hat{b} = \left[(a_0 + a_b x + \cdots + a_{m-1} x^{m-1}) \times (b_0 + b_b x + \cdots + b_{m-1} x^{m-1})\right] \mod f(x),
\]

Note that \(\hat{c}\) is also an element of \(GF(2^m)\), and \(f(x)\) is an irreducible polynomial over \(GF(2)\) with degree \(m\). The Montgomery product can be defined as

\[
\hat{c}_M = \hat{a} \cdot \hat{b} \cdot \mu^* \mod f(x),
\]

where \(\mu^* \cdot \mu = 1 \mod f(x)\) for \(\mu = x^m\), and then \(\mu^* = x^{-m} \mod f(x)\) is a constant element in \(GF(2^m)\). Since \(f(x)\) is irreducible, we find that \(f(x)\) and \(\mu\) are relatively prime, and a polynomial \(f^*(x)\) is existed to satisfy the following property:

\[
\mu \cdot \mu^* \cdot f(x) \cdot f^*(x) = 1.
\]

From (5), the polynomial \(f^*(x)\) can be obtained by using Euclidean algorithm [16]. The Montgomery product in (4) can be determined by

\[
\hat{q} = \hat{a} \cdot \hat{b} \cdot f^*(x) \mod \mu
\]

and

\[
\hat{c}_M = (\hat{a} \cdot \hat{b} + \hat{q} \cdot f(x))/\mu^*.
\]

As compared with the modulo \(f(x)\) operation in (4), the modular and division operations in (6) and (7) are much simpler due to \(\mu = x^m\). To be further partitioned into a series of operations for less complexity, the polynomial representation of (4) can be decomposed as the following iterative form:

\[
\hat{c}_M = [a_{m-1} \hat{b} x^{m-1} \mod f(x)] + [a_{m-2} \hat{b} x^{m-2} \mod f(x)] + \cdots + [a_0 \hat{b} x^{m-1} \mod f(x)]
\]

Then, the universal \((n, k)\) RS decoder over \(GF(2^m)\) is described in Section III. The design example which supports \(n \leq 255, t \leq 16\) for error-only or \(t \leq 8\) for error-and-erasure correcting and arbitrary irreducible polynomials with \(m \leq 8\) is provided as well. Section IV shows the corresponding chip implementation and measurement results. Finally, Section V gives the conclusion.

II. UNIVERSAL FFM

With polynomial representation, the modular multiplication of \(\hat{a}\) and \(\hat{b}\) in \(GF(2^m)\) can be expressed as

\[
\hat{c} = \hat{a} \cdot \hat{b} = \left[(a_0 + a_b x + \cdots + a_{m-1} x^{m-1}) \times (b_0 + b_b x + \cdots + b_{m-1} x^{m-1})\right] \mod f(x),
\]

Note that \(\hat{c}\) is also an element of \(GF(2^m)\), and \(f(x)\) is an irreducible polynomial over \(GF(2)\) with degree \(m\). The Montgomery product can be defined as

\[
\hat{c}_M = \hat{a} \cdot \hat{b} \cdot \mu^* \mod f(x)
\]

where \(\mu^* \cdot \mu = 1 \mod f(x)\) for \(\mu = x^m\), and then \(\mu^* = x^{-m} \mod f(x)\) is a constant element in \(GF(2^m)\). Since \(f(x)\) is irreducible, we find that \(f(x)\) and \(\mu\) are relatively prime, and a polynomial \(f^*(x)\) is existed to satisfy the following property:

\[
\mu \cdot \mu^* \cdot f(x) \cdot f^*(x) = 1.
\]

From (5), the polynomial \(f^*(x)\) can be obtained by using Euclidean algorithm [16]. The Montgomery product in (4) can be determined by

\[
\hat{q} = \hat{a} \cdot \hat{b} \cdot f^*(x) \mod \mu
\]

and

\[
\hat{c}_M = (\hat{a} \cdot \hat{b} + \hat{q} \cdot f(x))/\mu^*.
\]

As compared with the modulo \(f(x)\) operation in (4), the modular and division operations in (6) and (7) are much simpler due to \(\mu = x^m\). To be further partitioned into a series of operations for less complexity, the polynomial representation of (4) can be decomposed as the following iterative form:

\[
\hat{c}_M = [a_{m-1} \hat{b} x^{m-1} \mod f(x)] + [a_{m-2} \hat{b} x^{m-2} \mod f(x)] + \cdots + [a_0 \hat{b} x^{m-1} \mod f(x)]
\]
values. Based on our approach, the constant FFM s are also necessary to be universal in computing syndromes and error (or errata) values, which will be discussed in Section III-A, III-C, and III-D. Furthermore, an area-efficient key equation solver using the decomposed Berlekamp–Massey architecture is introduced in Section III-B.

A. Syndrome Calculator

The syndrome calculator computes 2t syndromes that can be expressed as

\[ S_i = r(\alpha^i) \]

\[ = \sum_{j=0}^{n-1} r_j \alpha^{i \cdot j} \]

\[ = (\cdots (r_{n-1 \cdot \alpha^{i \cdot d} + r_{n-2} \alpha^{i} + \cdots + \alpha^{i}) + r_0 \]

where \( \alpha \) is the primitive element of GF(2^m). The conventional syndrome calculator for \( S_i \) can be constructed in Fig. 3, which consists of a register, a finite-field adder, and a constant \( \alpha^j \)-FFM. For the universal syndrome calculator with Montgomery multiplications, the constant input of the \( \alpha^j \)-FFM should be \( \alpha^{i \cdot d} \) instead of \( \alpha^j \). However, the term \( \alpha^{i \cdot d} \) varies with the irreducible polynomial \( f(x) \), and the modified syndrome computation should be proposed for the constant Montgomery multiplication [20]. We first rewrite (19) as follows:

\[ S_i = \sum_{j=0}^{n-1} r_j \alpha^{i \cdot j} = \sum_{j=0}^{n-1} r_j \alpha^{d \cdot j} = \sum_{j=0}^{n-1} (r_j \alpha^{d \cdot j}) \alpha^{(i-d) \cdot j} \]

Then, the received symbol can be denoted by \( r'_j = r_j \alpha^{d \cdot j} \), and (20) can also be represented as

\[ S_i = (\cdots (r'_{n-1 \cdot \alpha^{i \cdot d} + r'_{n-2} \alpha^{i-d} + \cdots + r'_1) \alpha^{i-d} + r'_0 \]

Recalling the Montgomery multiplication defined in (15), the term \( \tilde{b} = \alpha^d \) can be taken as a constant input if \( i < d \), regardless of different \( f(x) \). It is also clear that \( \alpha^{i-d} = \alpha^d \) while \( i = d \), and the constant multiplier can be eliminated. Once \( i \) is larger than \( d \), the calculation of \( S_i \) can be processed through the conditions in (22), shown at the bottom of the page. To facilitate the key equation solver, the syndrome \( S_i \) should be modified to \( \tilde{S}_i = \alpha^{d \cdot S_i} \). Fig. 4 illustrates the proposed syndrome calculator for \( t \leq 8 \) and \( d = 8 \). Although there are at most 16 syndromes
should be computed, only 8 syndrome cells (SC$_1$ $\sim$ SC$_8$) are constructed. Based on (22), we can express $\hat{S}_{k}$ as follows:

$$\hat{S}_{k} = \alpha^{8}S_{k} = \begin{cases} \sum_{j=0}^{n-1} (r_{j} \alpha^{8(j-2)} \alpha^{-8}) \alpha^{(i-8)} s_{j} & , 0 < i \leq 8 \\
\sum_{j=0}^{t-1} (r_{j} \alpha^{8(j+2)} \alpha^{-8}) \alpha^{(i-8)+8} s_{j} & , 8 < i \leq 16. \end{cases}$$

(23)

For the case of $0 < i \leq 8$ and $8 < i \leq 16$, the received symbol $r_{j}$ should be multiplied by factors $\alpha^{8(j-2)}$ and $\alpha^{8(2j+2)}$, respectively. As shown in Fig. 4, two factor generators (FG$_1$ and FG$_2$) are allocated to produce the scaling factors with Montgomery multipliers. Since $j$ counts from $(n-1)$ to 0, the scaling factor $\alpha^{8(j-2)}$ and $\alpha^{8(2j+2)}$ can be obtained by sequentially multiplying $B_1 = 1$ and $B_2 = \alpha^{-8}$ with the initial value $\alpha^{8(n-1)}$ and $\alpha^{8(2n)}$. As described in (21), the constant input of the $\alpha^{8-FFM}$ in Fig. 4(b) is $\alpha^{i}$.

Although the syndrome calculator in Fig. 4 is proposed for $t \leq 8$, it can be extended to handle syndrome calculation for larger $t$. Assuming the case of $t \leq 16$, the first 16 syndromes $\hat{S}_{1} \sim \hat{S}_{16}$ can be computed from the same configuration, and other syndromes $\hat{S}_{17} \sim \hat{S}_{32}$ can also be calculated by

$$\hat{S}_{k} = \alpha^{8}S_{k} = \begin{cases} \sum_{j=0}^{n-1} (r_{j} \alpha^{8(j+2)} \alpha^{-8}) \alpha^{(i-16)-8} s_{j} & , 16 < i \leq 24 \\
\sum_{j=0}^{t-1} (r_{j} \alpha^{8(j+2)} \alpha^{-8}) \alpha^{(i-24)+8} s_{j} & , 24 < i \leq 32. \end{cases}$$

(24)

In (24), the constant Montgomery multiplication remains the same as compared with (23). The only difference is the scaling factors, $\alpha^{8(3j)+2}$ and $\alpha^{8(4j+2)}$, which can be generated by modifying FG$_1$ and FG$_2$ as well. In FG$_3$, the input $B_3$ and the initial value becomes $\alpha^{8-10}$ and $\alpha^{8(3n-1)}$, whereas the input $B_2$ becomes $\alpha^{8-24}$ with the initial value $\alpha^{8(4n-2)}$. Because there are only 16 computation cells in Fig. 4, it will double the computation time to complete 32 syndromes. Generally, the tradeoff between the number of syndrome cells and the computation time should depend on system specifications.

The erasure information $\alpha^{8k} \sim \alpha^{8k}$ should be generated for solving the key equation. Similar to $\hat{S}_{k} = \alpha^{8}S_{k}$, we also modify the erasure information as $\alpha^{8k+d} \sim \alpha^{8k+d-8}$. Fig. 5 illustrates the erasure generator with a constant $\alpha^{8-1}$-FFM, where the register initially contains $\alpha^{8(n-1)+d}$ and sequentially multiplies by $\alpha^{8-1}$. The register content will be the erasure value whenever the erasure flag (see Fig. 1) is activated according to the received data. Due to $\alpha^{-1} = \alpha^{(d-1)-d}$, the term $\alpha^{8-1}$ is the constant input of the $\alpha^{8-1}$-FFM in Fig. 5.

\[ \sigma^{(0)}(x) = \alpha^{d}, \quad \tau^{(0)}(x) = \alpha^{d} \]
\[ \Delta_0 = \alpha^{d+1}, \quad \delta = \alpha^{d} \]

- **Iterations from $i = 1$ to $u$:**
\[ \sigma^{(i)}(x) = \delta \sigma^{(i-1)}(x) + \Delta_{i-1} \tau^{(i-1)}(x) \cdot x \]
\[ \Delta_{i} = \alpha^{d+1}, \quad \tau^{(i)}(x) = \sigma^{(i)}(x) \]
\[ \begin{align*} \quad & \text{When $i = u$, the erasure-locator polynomial is obtained by} \\
& \sigma^{(u)}(x) = \alpha^{d+1} \lambda(x). \text{Before we start to calculate the errata-locator polynomial, several initial conditions should be modified as} \\
& \tau^{(0)}(x) = \alpha^{d+1} \lambda(x), \Delta_{u} = \hat{S}_{u+1}, \text{and $D_{u} = 0$.} \]

- **Iterations from $i = (u + 1)$ to $2l$:**
\[ \sigma^{(i)}(x) = \delta \sigma^{(i-1)}(x) + \Delta_{i-1} \tau^{(i-1)}(x) \cdot x \]
\[ \Delta_{i} = \sum_{j=0}^{i-1} \sigma^{(j)}(x) \hat{S}_{i+1-j} \]

\[ \text{Fig. 3. Syndrome calculator for $S_i$.} \]

\[ \text{Fig. 4. (a) Syndrome calculator with $d = 8$ and $t \leq 8$. (b) Syndrome cell SC$_i$ for $i = 1 \sim 7$. (c) Syndrome cell SC$_8$.} \]
If \( \Delta_{i-1} = 0 \) or \( D_{i-1} \geq i - u - D_{i-1} \)
\[
\tau^{(i)}(x) = \tau^{(i-1)}(x) \cdot x
\]
\( D_i = D_{i-1} \).
Otherwise
\[
\tau^{(i)}(x) = \sigma^{(i-1)}(x)
\]
\( D_i = i - u - D_{i-1}, \quad \delta = \Delta_{i-1} \).
If there are \( u \) erasures and \( v \) errors, the errata-locator polynomial will be finally obtained by
\[
\sigma^{(2)}(x) = \tilde{\Lambda}(x) = \sum_{j=0}^{u+v} \tilde{\Lambda}_j x^j.
\] (28)

According to the key equation, all coefficients of the errata-evaluator polynomial \( \tilde{\omega}(x) = \sum_{i=0}^{u+v-1} \tilde{\omega}_i x^i \) can be derived as
\[
\tilde{\omega}_i = \sum_{j=0}^{i} \tilde{\Lambda}_j S_{i+1-j} \text{ for } i = 0 \sim u + v - 1.
\] (29)

Since we apply the Montgomery multiplication to all FFM computations, each input containing an additional factor \( \alpha^d \) will produce the product that also carries with the same factor \( \alpha^d \). Thus, the erasure-locator polynomial can be obtained as \( \alpha^d \lambda(x) \) by (25). The final result of (26) will be \( \tilde{\Lambda}(x) \equiv \eta \sigma(x) \Lambda(x) \), where \( \eta \) is ineffective for searching roots of \( \sigma(x) \lambda(x) = 0 \). It is also clear that the same errata value will be evaluated since the errata-evaluator polynomial \( \tilde{\omega}(x) = \eta \cdot \omega(x) \) has the same factor \( \eta \) (23).

Based on the decomposed architecture in [5], the key equation solver with only three Montgomery multipliers is demonstrated in Fig. 6. There are two memory buffers denoted by buffer-\( \sigma \) and buffer-\( \tau \) for storing \( \sigma^{(i-1)}(x) \) and \( \tau^{(i-1)}(x) \). Due to the uniformity of (25) and (26), this architecture can be configured to not only calculate the erasure-locator polynomial but perform the inversionless Berlekamp–Massey algorithm. For \( i = 1 \sim u \), it is in polynomial expansion mode that calculates the erasure-locator polynomial with \( \Delta_{i-1} = \alpha^{2i+d} \) and \( \delta = \alpha^d \) in (25). After \( u \) iterations, the result \( \alpha^d \lambda(x) \) will be stored in both buffer-\( \sigma \) and buffer-\( \tau \), which are ready for the following Berlekamp–Massey algorithm. As the syndrome polynomial \( \tilde{S}(x) \) is available, (26) and (27) will be executed from \( i = u + 1 \) to \( 2t \), and finally \( \tilde{\Lambda}(x) \) will be in buffer-\( \sigma \). Notice that the same computational structure in Fig. 6 can also calculate the errata-evaluator polynomial \( \tilde{\omega}(x) \) according to (29), which is quite similar to the discrepancy evaluation in (27). We let \( \Delta_{i-1} = 0 \) and \( \delta = \alpha^d \). The coefficient \( \tilde{\Lambda}_j \) from buffer-\( \sigma \) will be multiplied by \( S_{i+1-j} \), and the product will be accumulated to \( \tilde{\omega}_j \). Furthermore, the polynomial expansion in (25) can work in parallel with syndrome calculator because it is independent of the syndromes \( \tilde{S}_i \), leading to less decoding latency.

C. Chien Search

After the key equation solver, Chien search operations are used to repeatedly check \( \tilde{\Lambda}(x) \equiv 0 \) or not for \( x = \alpha^0, \alpha^{-1}, \cdots, \alpha^{-(n-1)} \). The calculation of Chien search can be represented as
\[
\tilde{\Lambda}(\alpha^{-i}) = \sum_{j=0}^{u+v} \tilde{\Lambda}_j \alpha^{-i+j}, \quad \text{for } i = 0 \sim n - 1
\] (30)
which is similar to the syndrome calculation (19). The constant multiplier can be used after modifying (30) to
\[
\tilde{\Lambda}(\alpha^{-i}) = \sum_{j=0}^{u+v} \tilde{\Lambda}_j \left( \alpha^{(d-j)-d} \right)^i
\] (31)
\[
= \tilde{\Lambda}_0 + \sum_{p=0}^{\lfloor \frac{d}{2} \rfloor - 1} \left( \alpha^{-pd} \right)^i \cdot \sum_{j=1}^{d} \tilde{\Lambda}_{pd+j} \left( \alpha^{(d-j)-d} \right)^i.
\] (32)

Note that all the coefficients of \( \tilde{\Lambda}(x) \) in (32) except \( \tilde{\Lambda}_0 \) are divided into \( \left( \left( \lfloor \frac{d}{2} \rfloor \right) / (d) \right) \) groups and \( \tilde{\Lambda}_{pd+j} = 0 \) if \( pd+j > u + v \). The term \( \tilde{\Lambda}_{pd+j} \cdot \alpha^{(d-j)-d} \) can be represented as a constant Montgomery multiplication because \( 0 \leq d - j < d \). With \( d = 8 \) and \( t \leq 16 \), the Chien search structure with two groups of 8 Chien search cells (CC1 \( \sim \) CC8) is presented in Fig. 7. Based on (32), the \( j \)th Chien search cell, CC\( j \), uses a constant multiplier in which the constant input is \( \alpha^{-pd} \). From Fig. 7, the polynomial \( \tilde{\Lambda}_{odd}(x) \) is defined to be \( \tilde{\Lambda}(x) \) with zero coefficients in the even degree terms, and the output \( \tilde{\Lambda}_{odd}(\alpha^{-i}) \) will be determined for calculating errata values. In addition, the value \( \tilde{\Lambda}_{odd}(\alpha^{-i}) \) is equal to \( \alpha^{-i} \tilde{\Lambda}(\alpha^{-i}) \) because
\[
\tilde{\Lambda}(x) = \sum_{j=0}^{u+v} \tilde{\Lambda}_j x^{2j}
\]
\[
\lambda^{-1} = \sum_{j=0}^{\lceil \log_2 \Delta t \rceil - 1} \hat{A}_{2j+1} x^{2j+1} \Delta t^{-1} \hat{A}_{\text{odd}}(x),
\]

where \(\hat{\alpha} = \beta^{-1}\) indicates the \(i\)th root of \(\hat{\Lambda}(x)\). The corresponding architecture to calculate the term \(\hat{\beta}^{-1} \hat{\omega}(\hat{\beta}^{-1})\) with \(d = 8\) and \(t \leq 16\) is shown in Fig. 8, where the cell \(CC_j\) is identical to the \(j\)th Chien search cell. The difference is the initial value being \(\hat{\omega}_{j-1}\) instead of \(\hat{\lambda}_j\) in Fig. 7(a). The divider performs the finite-field division by using a Montgomery multiplier and an inversion table. To satisfy different finite-field definitions in the universal architecture, an on-the-fly inversion table is realized with a RAM. As shown in Fig. 9, each value \(\alpha^{-i+d}\) will be written to the address \(\alpha^i\) as counting \(i\) counts from 0 to \(n - 1\). Note that the on-the-fly inversion table can be created in parallel with the syndrome calculation.

### IV. Chip Implementation

Based on Montgomery multiplication algorithm, Fig. 10 shows the universal \((n, k)\) RS decoder over \(GF(2^m)\) with an on-the-fly inversion table. The related interface of control signals with arbitrary \(n \leq 2^m - 1\), \(t\), and the irreducible polynomial \(f(x)\) are ignored for simplification. The dual-bank static RAM (SRAM) of 1 K-byte is embedded to buffer 4 received codewords. In the syndrome calculator, there are 16 syndrome cells that concurrently compute syndrome values. To support the case of \(t \leq 8\) with error-and-erasure corrections, 16 syndrome cells are sufficient. However, they can support the case of \(t \leq 16\) with error-only corrections. According to (23) and (24), \(S_1 \sim S_{16}\) can be calculated from the received codeword that is written into the FIFO memory as well, and \(S_{17} \sim S_{32}\) are subsequently obtained from the same codeword read from the FIFO memory. The erasure generator produces the erasure information \(\alpha^{t+8} \sim \alpha^{t+16}\) according to the erasure flag. Based on the inversionless Berlekamp–Massey algorithm, we implement the key equation solver to determine the erasure-locator polynomial \(\hat{\Lambda}(x)\), the errata-locator polynomial \(\hat{\Lambda}(x)\), and the errata-evaluator polynomial \(\hat{\omega}(x)\). As shown in Fig. 6, only three Montgomery multipliers are required in our decomposed architecture. In the Chien search block, the architecture in Fig. 7 not only checks roots of \(\hat{\Lambda}(x)\) but also

---

**Fig. 7.** (a) Chien search module with \(d = 8\) and \(t \leq 16\). (b) Chien cell \(CC_j\).

**Fig. 8.** Error value evaluator with \(d = 8\) and \(t \leq 16\).

**Fig. 9.** Finite-field divider with on-the-fly inversion table.

---

**TABLE II**

**UNIVERSAL RS DECODER CHIP SUMMARY**

<table>
<thead>
<tr>
<th>Technology</th>
<th>0.18-(\mu)m 1P6M CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip size</td>
<td>2.25 mm(^2)</td>
</tr>
<tr>
<td>Core size</td>
<td>1.21 mm(^2)</td>
</tr>
<tr>
<td>Gate count</td>
<td>46.4K</td>
</tr>
<tr>
<td>Embedded SRAM</td>
<td>8K bits (FIFO memory)</td>
</tr>
<tr>
<td>Supply voltage</td>
<td>1.62V~1.98V</td>
</tr>
<tr>
<td>Clock rate</td>
<td>160MHz</td>
</tr>
<tr>
<td>Power consumption</td>
<td>68.1mW (1.8V and 160MHz)</td>
</tr>
</tbody>
</table>
TABLE III

<table>
<thead>
<tr>
<th>Design</th>
<th>[25]*</th>
<th>[7]**</th>
<th>[24]*</th>
<th>Proposed***</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>0.18-μm</td>
<td>0.18-μm</td>
<td>0.25-μm</td>
<td>0.18-μm</td>
</tr>
<tr>
<td>(n, k)</td>
<td>fixed</td>
<td>configurable</td>
<td>configurable</td>
<td>configurable</td>
</tr>
<tr>
<td>G</td>
<td>≤ 8</td>
<td>≤ 8</td>
<td>≤ 8</td>
<td>≤ 16</td>
</tr>
<tr>
<td>gf(2^n)</td>
<td>m = 8</td>
<td>m = 8</td>
<td>m ≤ 8</td>
<td>m ≤ 8</td>
</tr>
<tr>
<td>f(x)</td>
<td>fixed</td>
<td>fixed</td>
<td>arbitrary</td>
<td>arbitrary</td>
</tr>
<tr>
<td>Gates count</td>
<td>20K</td>
<td>23K</td>
<td>24K</td>
<td>44K</td>
</tr>
<tr>
<td>Max. throughput</td>
<td>3.2Gb/s</td>
<td>3.2Gb/s</td>
<td>560Mb/s</td>
<td>48Mb/s</td>
</tr>
<tr>
<td>Power consumption</td>
<td>N/A</td>
<td>N/A</td>
<td>48.4mW</td>
<td>68.1mW</td>
</tr>
</tbody>
</table>

* Synthesis Results
** Post-layout Simulation Results
*** Measurement Results

Fig. 10. Universal RS decoder architecture to correct both errors and erasures.

Fig. 11. 0.18-μm universal RS decoder chip photo.

generates \( \beta^{-1} \tilde{\lambda}'(\beta^{-1}) \) for errata value evaluation. Finally, the errata value according to (34) will be calculated.

The universal RS decoder is implemented with the standard 0.18-μm 1P6M CMOS technology and measured to achieve the maximum 160 MHz clock rate at the supply voltage 1.62–1.98 V. The die photo and the chip summary are shown in Fig. 11 and Table II. If the chip works in the GF(2^n) mode, the maximum measured throughput is 8 bits \( \times 160 \text{ MHz} = 1.28 \text{ Gb/s} \) with 68.1-mW core power consumption. Compared with other approaches listed in Table III, the proposed design has more flexibility while achieving high decoding throughput. Notice that the decoder in [24] applies the serial architecture to realize the universality with the limited throughput. The gates count of the present decoder is also comparable with other fixed or configurable (n, k) RS decoders.

V. CONCLUSION

We present the universal RS architecture for error-and-erasure decoding. The proposed architecture can accommodate variable codeword length and correctable errors, as well as arbitrary finite-field degrees and different irreducible polynomials. Without extra FFMs, the proposed decomposed architecture can support error-and-erasure corrections. In summary, the universal RS decoder is both flexible and cost-efficient as well.

ACKNOWLEDGMENT

The authors appreciate National Chip Implementation Center for chip measurement assistance.

REFERENCES

is currently an Associate Professor. His research interests include coding theory, VLSI architectures and integrated circuit design for communications, and signal processing. His research interests include algorithms and VLSI architectures in signal processing, especially for error control codes and crypto-systems. Recently, he also committed himself to joint source/channel coding schemes and multi-Gb/s chip implementation for wireless communications.

Fu-Ke Chang received the B.S. degree from the Department of Electronics Engineering, National Cheng Kung University, Tainan, Taiwan, and the Master’s degree from the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu, Taiwan, in 2003 and 2005, respectively. He is currently working for HIMAX Inc., Hsinchu, Taiwan, for three years. His recent research interests include error control code algorithm and architecture and TFT-LCD driver implementation.

Chien-Ching Lin received the B.S. degree in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 2001, and the Ph.D. degree in electronics engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 2006. From 2007 to 2008, he was a Post-Doctoral researcher in the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu, Taiwan. In February 2008, he joined Ambarella Taiwan Ltd., Hsinchu, Taiwan, where he is currently an Engineer working on the design of multimedia systems. His recent research interests include coding theory, VLSI architectures and integrated circuit design for communications, and signal processing.

Hsie-Chia Chang (S’01–M’03) received the B.S. and M.S., and the Ph.D. degrees in electronics engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1995, 1997, and 2002, respectively. From 2002 to 2003, he was with OSP/DEI in MediaTek Corp., working in the area of decoding architectures for Combo single chip. In February 2003, he joined the faculty of the Electronics Engineering Department, National Chiao-Tung University, where he is currently an Associate Professor. His research interests include algorithms and VLSI architectures in signal processing, especially for error control codes and crypto-systems. Recently, he also committed himself to joint source/channel coding schemes and multi-Gb/s chip implementation for wireless communications.

Chen-Yi Lee (S’89–M’90) received the B.S. degree from the National Chiao-Tung University, Hsinchu, Taiwan, in 1982, and the M.S. and Ph.D. degrees from Katholieke Universiteit Leuven (KUL), Leuven, Belgium, in 1986 and 1990, respectively, all in electrical engineering.

From 1986 to 1990, he was with IMEC/VSDM, working in the area of architecture synthesis for digital signal processor (DSP). From 2000 to 2003, he served as the Director of Chip Implementation Center (CIC), an organization for IC design promotion in Taiwan. In February 1991, he joined the faculty of the Electronics Engineering Department, National Chiao-Tung University, where he is currently a Professor and Department Chair. His recent research interests include VLSI algorithms and architectures for high-throughput DSP applications. He is also active in various aspects of short-range wireless communications, system-on-chip design technology, very low power designs, and multimedia signal processing.

Dr. Lee was the former IEEE CAS Taipei Chapter Chair from 2000 to 2001, the SIP task leader of National SoC Research Program from 2003 to 2005, and the microelectronics program coordinator of Engineering Division under National Science Council of Taiwan from 2003 to 2005.