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 1P6M 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 mm^2 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 $\mathrm{GF}(2^m)$ 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 errata 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

$$S(x)\sigma(x) = \Omega(x) \bmod x^{2t} \tag{1}$$

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

$$S(x)\lambda(x)\sigma(x) = \omega(x) \bmod x^{2t} \tag{2}$$

where $\lambda(x) = (1 - \alpha^{l_1}x)(1 - \alpha^{l_2}x)\cdots(1 - \alpha^{l_u}x)$ indicates *erasure-locator polynomial* with u erasure information $\alpha^{l_1}, \alpha^{l_2}, \ldots, \alpha^{l_u}$, 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].

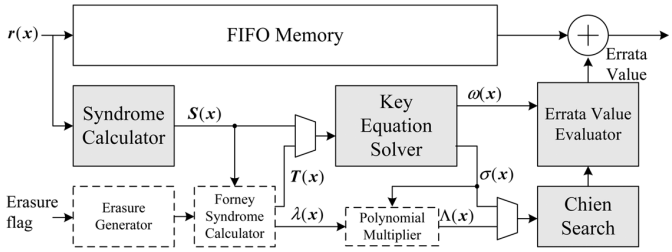

Fig. 1. Block diagram of the RS decoder. The dotted blocks are required for correcting both errors and erasures.

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 $\mathrm{GF}(2^m)$ 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 $\mathrm{GF}(2^m)$. 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.

Manuscript received August 27, 2008. First published December 02, 2008; current version published September 04, 2009. This work was supported by National Science Council (NSC) of Taiwan, R.O.C., under Grant NSC 97-2220-E-009-017, and by MOEA of Taiwan, R.O.C., under MOEA 96-EC-17-A-01-S1-048. This paper was recommended by Associate Editor V. Öwall.

H.-C. Chang and C.-Y. Lee are with the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan (e-mail: hcchang@si2lab.org).

C.-C. Lin was with the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan. He is now with Ambarella Taiwan Ltd., Hsinchu 300, Taiwan.

F.-K. Chang was with the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan. He is now with HIMAX Technologies, Inc., Hsinchu 300, Taiwan.

Digital Object Identifier 10.1109/TCSI.2008.2010143

TABLE I
RS CODE SPECIFICATIONS IN VARIOUS APPLICATIONS

Application		Code specification
DVB-T [8], DVB-S [9]		(204,188) RS code over $GF(2^8)$
DVB-H [10]		(255,191) RS code over $GF(2^8)$
ITU-T J.83 [11]	Annex A, C	(204,188) RS code over $GF(2^8)$
	Annex B	(128,122) RS code over $GF(2^7)$
	Annex D	(207,187) RS code over $GF(2^8)$
G.975 FEC [12]		(255,239) RS code over $GF(2^8)$
Flash memory [13]		(520,512) RS code over $GF(2^{10})$
DVD	ADIP	(13,8) RS code over $GF(2^4)$
	row	(182,172) RS code over $GF(2^8)$
	column	(208,192) RS code over $GF(2^8)$
Blu-ray Disc	LDC	(248,216) RS code over $GF(2^8)$
	BIS	(62,30) RS code over $GF(2^8)$

Then, the universal (n, k) RS decoder over $\mathrm{GF}(2^m)$ 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 $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ in $\mathrm{GF}(2^m)$ can be expressed as

$$\begin{aligned}\hat{\mathbf{c}} &= \hat{\mathbf{a}} \cdot \hat{\mathbf{b}} \\ &= [(a_0 + a_1 x + \cdots + a_{m-1}x^{m-1}) \\ &\quad \times (b_0 + b_1 x + \ldots + b_{m-1}x^{m-1})] \bmod f(x). \end{aligned} \tag{3}$$

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

$$\hat{\mathbf{c}}_{\mathbf{M}} = \hat{\mathbf{a}} \cdot \hat{\mathbf{b}} \cdot \mu^* \bmod f(x) \tag{4}$$

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

$$\mu \cdot \mu^* + f(x) \cdot f^*(x) = 1. \tag{5}$$

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

$$\hat{\mathbf{q}} = \hat{\mathbf{a}} \cdot \hat{\mathbf{b}} \cdot f^*(x) \bmod \mu \tag{6}$$

$$\hat{\mathbf{c}}_{\mathbf{M}} = (\hat{\mathbf{a}} \cdot \hat{\mathbf{b}} + \hat{\mathbf{q}} \cdot f(x))/u. \tag{7}$$

As compared with the modulo $f(x)$ operation in (4), the modular and division operations in (6) and (7) are much simple 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:

$$\begin{aligned}\hat{\mathbf{c}}_{\mathbf{M}} &= [a_{m-1}\hat{\mathbf{b}}x^{-1} \bmod f(x)] + [a_{m-2}\hat{\mathbf{b}}x^{-2} \bmod f(x)] \\ &\quad + \ldots + [a_0\hat{\mathbf{b}}x^{-m} \bmod f(x)] \\ &= [a_{m-1}\hat{\mathbf{b}} + [a_{m-2}\hat{\mathbf{b}} + \ldots + [a_0\hat{\mathbf{b}}x^{-1} \bmod f(x)]\cdots] \\ &\quad \times x^{-1} \bmod f(x)]x^{-1} \bmod f(x). \end{aligned} \tag{8}$$

Similar to the derivation of (6) and (7), the Montgomery product $\hat{\mathbf{c}}_{\mathbf{M}}$ can be obtained by the following iterative computations:

- Initial conditions

$$\hat{\mathbf{A}}^{(0)} = 0.$$

- Iterations from $i = 0$ to $m - 1$

$$\hat{\mathbf{Q}} = [(\hat{\mathbf{A}}^{(\mathbf{i})} + a_i\hat{\mathbf{b}})f^*(x)] \bmod x \tag{10}$$

$$\hat{\mathbf{A}}^{(\mathbf{i}+1)} = (\hat{\mathbf{A}}^{(\mathbf{i})} + a_i\hat{\mathbf{b}} + \hat{\mathbf{Q}}f(x))/x. \tag{11}$$

After m iterations, $\hat{A}^{(m)}$ will be equal to $\hat{\mathbf{c}}_{\mathbf{M}}$. Since $f(x)$ is irreducible and all elements are represented in binary digit over $\mathrm{GF}(2^m)$, the term $f^*(x)$ in (10) indicating the multiplicative inverse of $f(x)$ modulo x is always equal to 1 and can be eliminated. Thus, the result $\hat{\mathbf{Q}}$ will be the constant term of $(\hat{\mathbf{A}}^{(\mathbf{i})} + a_i\hat{\mathbf{b}})$. For the iteration number varied with the field degree m, we define a constant integer d with $m \leq d$ and let $u^* = x^{-d} \bmod f(x)$. The modified computation process with the fixed iteration number can be shown as follows:

- Initial conditions

$$\hat{\mathbf{A}}^{(0)} = 0.$$

- Iterations from $i = 0$ to $d - 1$

$$a_i = 0, \qquad \text{for } i \geq m \tag{12}$$

$$\hat{\mathbf{T}} = \hat{\mathbf{A}}^{(\mathbf{i})} + a_i\hat{\mathbf{b}} \tag{13}$$

$$\hat{\mathbf{A}}^{(\mathbf{i}+1)} = (\hat{\mathbf{T}} + t_0 f(x))/x. \tag{14}$$

The final result is

$$\hat{\mathbf{c}}_{\mathbf{M}} = \hat{\mathbf{A}}^{(\mathbf{d})} = \hat{\mathbf{a}} \cdot \hat{\mathbf{b}} \cdot x^{-d} \bmod f(x). \tag{15}$$

Here we set $a_i = 0$ for $i \geq m$ to ensure correct operations and denote t_0 in (14) as a constant term of $\hat{\mathbf{T}}$. For any irreducible polynomial $f(x)$ with degree $m \leq d$, the Montgomery product (15) can be completed within d modular-free iterations of (12)–(14). However, there is still a factor $\mu^* = x^{-d}$ involved in the product $\hat{\mathbf{c}}_{\mathbf{M}}$ in contrast with the original result $\hat{\mathbf{c}}$. In order to remove this factor, one additional Montgomery multiplication

$$\hat{\mathbf{c}} = \hat{\mathbf{c}}_{\mathbf{M}} \cdot \hat{\boldsymbol{\delta}} \cdot x^{-d} \bmod f(x) \tag{16}$$

is applied with $\hat{\boldsymbol{\delta}} = x^{2d}$ to obtain the original product $\hat{\mathbf{c}} = \hat{\mathbf{a}} \cdot \hat{\mathbf{b}}$. In many applications, this additional product correction of (16) is required only after a series of Montgomery multiplications.

Fig. 2 illustrates an example of Montgomery multiplier structure with $d = 4$, in which any irreducible polynomial over $\mathrm{GF}(2^m)$ with $m \leq 4$ can be performed. The inputs $\hat{\mathbf{a}}, \hat{\mathbf{b}}$, and $f(x)$ can be represented

$$\begin{aligned}\hat{\mathbf{a}} &= a_3x^3 + a_2x^2 + a_1x + a_0 \\ \hat{\mathbf{b}} &= b_3x^3 + b_2x^2 + b_1x + b_0 \\ f(x) &= f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_1.\end{aligned}$$

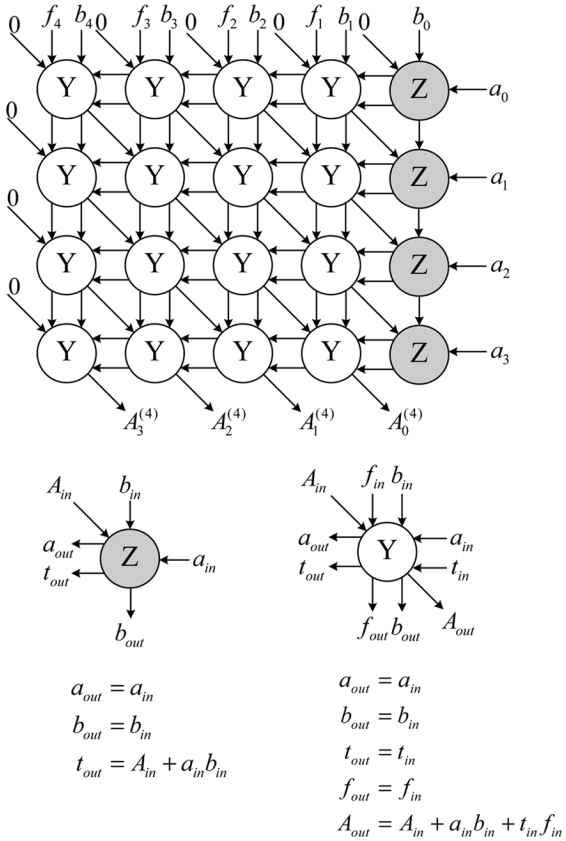

Fig. 2. Montgomery multiplier structure for $\mathrm{GF}(2^m)$ with $m \leq 4$.

As derived in (15), the result $\hat{\mathbf{c}}_{\mathbf{M}}$ will be

$$\hat{\mathbf{A}}^{(4)} = A_3^{(4)}x^3 + A_2^{(4)}x^2 + A_1^{(4)}x + A_0^{(4)}.$$

III. Universal RS Decoder Architecture

As shown in Fig. 1, the syndrome calculator generates $S(x)$ with $2t$ syndromes from the received polynomial $r(x)$. If there is erasure information, the Forney syndrome calculator will deliver Forney syndrome polynomial $T(x)$ and erasure-locator polynomial $\lambda(x)$. From $S(x)$ or $T(x)$, the key equation solver evaluates both $\sigma(x)$ and $\omega(x)$ by using either Berlekamp–Massey [17], [18] or Euclidean algorithm [6], [19]. Then the errata-locator polynomial $\Lambda(x) = \lambda(x)\sigma(x)$ can be calculated. After the Chien search block identifies error or erasure locations, the errata value evaluator computes error values for error-only decoding or errata values for error-and-erasure decoding. There is also a first-in and first-out (FIFO) memory storing the received vector $r(x)$. All correctable errors can be corrected by adding $r(x)$ with corresponding error or errata values. Based on our approach, the constant FFMs 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

$$S_i = r(\alpha^i) \tag{17}$$

$$= \sum_{j=0}^{n-1} r_j \alpha^{i\cdot j} \tag{18}$$

$$= (\cdots((r_{n-1}\alpha^i + r_{n-2})\alpha^i + \cdots)\alpha^i + r_0 \tag{19}$$

where α is the primitive element of $\mathrm{GF}(2^m)$. 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 α^i-FFM. For the universal syndrome calculator with Montgomery multiplications, the constant input of the α^i-FFM should be α^{i+d} instead of α^i. However, the term α^{i+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:

$$S_i = \sum_{j=0}^{n-1} r_j \alpha^{i\cdot j} = \sum_{j=0}^{n-1} r_j \alpha^{d\cdot j + (i-d)\cdot j} = \sum_{j=0}^{n-1} (r_j \alpha^{d\cdot j})\alpha^{(i-d)\cdot j}. \tag{20}$$

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

$$S_i = (\cdots(r'_{n-1}\alpha^{i-d} + r'_{n-2})\alpha^{i-d} + \cdots + r'_1)\alpha^{i-d} + r'_0. \tag{21}$$

Recalling the Montgomery multiplication defined in (15), the term $\hat{\mathbf{b}} = \alpha^i$ can be taken as a constant input if $i < d$, regardless of different $f(x)$. It is also clear that $\alpha^{i-d} = \alpha^0$ 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 $\tilde{S}_i = \alpha^d S_i$. Fig. 4 illustrates the proposed syndrome calculator for $t \leq 8$ and $d = 8$. Although there are at most 16 syndromes

$$S_i = \begin{cases} \sum_{j=0}^{n-1}(r_j\alpha^{d\cdot j})\alpha^{(i-d)\cdot j}, & 0 < i \leq d \\ \sum_{j=0}^{n-1}(r_j\alpha^{2d\cdot j})\alpha^{((i-d)-d)\cdot j}, & d < i \leq 2d \\ \vdots & \vdots \\ \sum_{j=0}^{n-1}\left(r_j\alpha^{\left(\lfloor\frac{2t}{d}\rfloor+1\right)d\cdot j}\right)\alpha^{\left(\left(i-\lfloor\frac{2t}{d}\rfloor d\right)-d\right)\cdot j}, & \left(\lceil\frac{2t}{d}\rceil - 1\right)d < i \leq 2t \end{cases} \tag{22}$$

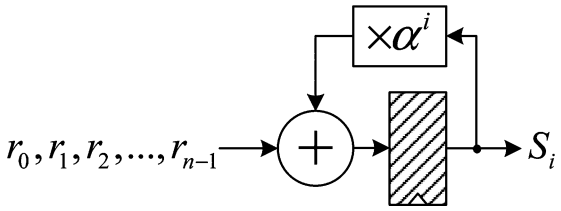

Fig. 3. Syndrome calculator for S_i.

should be computed, only 8 syndrome cells ($\mathrm{SC}_1 \sim \mathrm{SC}_8$) are constructed. Based on (22), we can express $\tilde{S}_i$ as follows:

$$\tilde{S}_i = \alpha^8 S_i = \begin{cases} \sum_{j=0}^{n-1}(r_j\alpha^{8(j+2)}\alpha^{-8})\alpha^{(i-8)\cdot j}, & 0 < i \le 8 \\ \sum_{j=0}^{n-1}(r_j\alpha^{8(2j+2)}\alpha^{-8})\alpha^{((i-8)-8)\cdot j}, & 8 < i \le 16. \end{cases} \tag{23}$$

For the case of $0 < i \le 8$ and $8 < i \le 16$, 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 α^{i-8}-FFM in Fig. 4(b) is α^i.

Although the syndrome calculator in Fig. 4 is proposed for $t \le 8$, it can be extended to handle syndrome calculation for larger t. Assuming the case of $t \le 16$, the first 16 syndromes $\tilde{S}_1 \sim \tilde{S}_{16}$ can be computed from the same configuration, and other syndromes $\tilde{S}_{17} \sim \tilde{S}_{32}$ can also be calculated by

$$\tilde{S}_i = \alpha^8 S_i = \begin{cases} \sum_{j=0}^{n-1}(r_j\alpha^{8(3j+2)}\alpha^{-8})\alpha^{((i-16)-8)\cdot j}, & 16 < i \le 24 \\ \sum_{j=0}^{n-1}(r_j\alpha^{8(4j+2)}\alpha^{-8})\alpha^{((i-24)-8)\cdot j}, & 24 < i \le 32. \end{cases} \tag{24}$$

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_1, the input B_1 and the initial value becomes α^{-16} and $\alpha^{8(3n-1)}$, whereas the input B_2 becomes α^{-24} with the initial value $\alpha^{8(4n-2)}$. Because there are only 16 computation cells in Fig. 4, it will double the calculation 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 $\alpha^{l_1} \sim \alpha^{l_u}$ should be generated for solving the key equation. Similar to $\tilde{S}_i = \alpha^d S_i$, we also modify the erasure information as $\alpha^{l_1+d} \sim \alpha^{l_u+d}$. Fig. 5 illustrates the erasure generator with a constant α^{-1}-FFM, where the register initially contains $\alpha^{(n-1)+d}$ and sequentially multiplies by α^{-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 α^{d-1} is the constant input of the α^{-1}-FFM in Fig. 5.

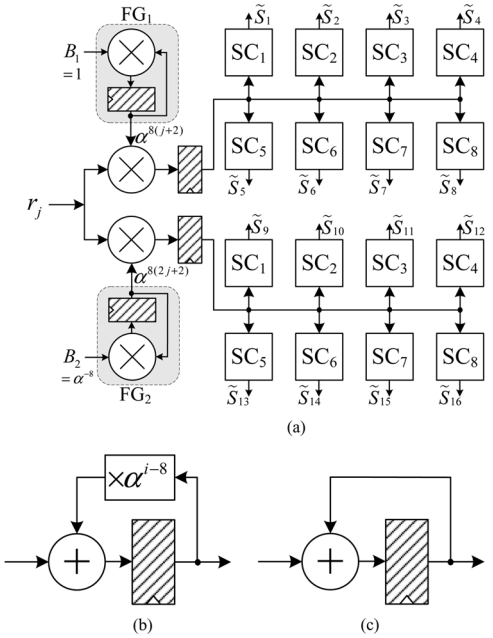

Fig. 4. (a) Syndrome calculator with $d = 8$ and $t \le 8$. (b) Syndrome cell SC_i for $i = 1 \sim 7$. (c) Syndrome cell SC_8.

B. Key Equation Solver

The algorithm in solving key equation (1) or (2) can be either Berlekamp–Massey algorithm or Euclidean algorithm. Since Berlekamp–Massey algorithm has fixed $2t$ iterations, it is much regular and suitable for our universal RS decoder. Moreover, the inversionless architecture is also applied to avoid the finite-field division [5], [21]. As reported in [22], those computations of Forney syndrome polynomial and errata-locator polynomial can be combined with Berlekamp–Massey algorithm. From the syndrome polynomial, $\tilde{S}(x) = \sum_{i=1}^{2t} \tilde{S}_i x^{i-1} = \sum_{i=1}^{2t} \alpha^d S_i x^{i-1}$, the inversionless Berlekamp–Massey algorithm with u erasure information can be proposed as follows:

- Initial conditions:

$$\sigma^{(0)}(x) = \alpha^d, \quad \tau^{(0)}(x) = \alpha^d$$
$$\Delta_0 = \alpha^{l_1+d}, \quad \delta = \alpha^d$$

- Iterations from $i = 1$ to u:

$$\sigma^{(i)}(x) = \delta\sigma^{(i-1)}(x) + \Delta_{i-1}\tau^{(i-1)}(x)\cdot x \tag{25}$$
$$\Delta_i = \alpha^{l_{i+1}+d}, \quad \tau^{(i)}(x) = \sigma^{(i)}(x)$$

  When $i = u$, the erasure-locator polynomial is obtained by $\sigma^{(u)}(x) = \alpha^d\lambda(x)$. Before we start to calculate the errata-locator polynomial, several initial conditions should be modified as $\tau^{(u)}(x) = \sigma^{(u)}(x) = \alpha^d\lambda(x), \Delta_u = \tilde{S}_{u+1}$, and $D_u = 0$.
- Iterations from $i = (u+1)$ to $2t$:

$$\sigma^{(i)}(x) = \delta\sigma^{(i-1)}(x) + \Delta_{i-1}\tau^{(i-1)}(x)\cdot x \tag{26}$$
$$\Delta_i = \sum_{j=0}^{t-1}\sigma_j^{(i)}\tilde{S}_{i+1-j}. \tag{27}$$

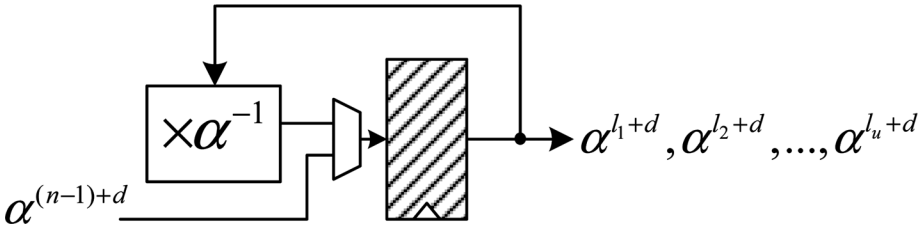

Fig. 5. Erasure generator corresponds to the received sequence r_j.

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

$$\sigma^{(2t)}(x) \equiv \tilde{\Lambda}(x) = \sum_{j=0}^{u+v} \tilde{\Lambda}_j x^j. \tag{28}$$

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

$$\tilde{\omega}_i = \sum_{j=0}^{i} \tilde{\Lambda}_j \tilde{S}_{i+1-j} \text{ for } i = 0 \sim u+v-1. \tag{29}$$

Since we apply the Montgomery multiplication to all FFM computations, each input containing an additional factor α^d will produce the product that also carries with the same factor α^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 η 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 η (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-σ and buffer-τ 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 $\Delta_{i-1} = \alpha^{l_i+d}$ and $\delta = \alpha^d$ in (25). After u iterations, the result $\alpha^d\lambda(x)$ will be stored in both buffer-σ and buffer-τ, 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-σ. 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-σ will be multiplied by $\tilde{S}_{i+1-j}$, and the product will be accumulated to be $\tilde{\omega}_i$. 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.

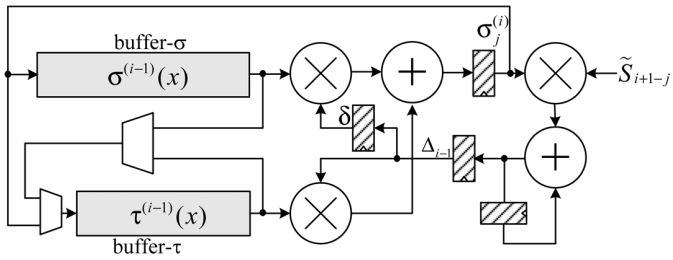

Fig. 6. Key equation solver to perform inversionless Berlekamp–Massey algorithm.

C. Chien Search

After the key equation solver, Chien search operations are used to repeatedly check $\tilde{\Lambda}(x) = 0$ or not for $x = \alpha^0, \alpha^{-1}, \ldots, \alpha^{-(n-1)}$. The calculation of Chien search can be represented as

$$\tilde{\Lambda}(\alpha^{-i}) = \sum_{j=0}^{u+v} \tilde{\Lambda}_j \alpha^{-i \cdot j}, \qquad \text{for } i = 0 \sim n-1 \tag{30}$$

which is similar to the syndrome calculation (19). The constant multiplier can be used after modifying (30) to

$$\tilde{\Lambda}(\alpha^{-i}) = \sum_{j=0}^{u+v} \tilde{\Lambda}_j \left(\alpha^{(d-j)-d}\right)^i \tag{31}$$

$$= \tilde{\Lambda}_0 + \sum_{\pi=0}^{\lceil \frac{2t}{d} \rceil - 1} (\alpha^{-\pi d})^i \cdot \sum_{j=1}^{d} \tilde{\Lambda}_{\pi d+j} \left(\alpha^{(d-j)-d}\right)^i. \tag{32}$$

Note that all the coefficients of $\tilde{\Lambda}(x)$ in (32) except $\tilde{\Lambda}_0$ are divided into $\lceil (2t)/(d) \rceil$ groups and $\tilde{\Lambda}_{\pi d+j} = 0$ if $\pi d + j > u + v$. The term $\tilde{\Lambda}_{\pi d+j} \cdot \alpha^{((d-j)-d)}$ 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 ($\mathrm{CC}_1 \sim \mathrm{CC}_8$) is presented in Fig. 7. Based on (32), the jth Chien search cell, CC_j, uses a constant multiplier in which the constant input is α^{8-j}. From Fig. 7, the polynomial $\tilde{\Lambda}_{\mathrm{odd}}(x)$ is defined to be $\tilde{\Lambda}(x)$ with zero coefficients in the even degree terms, and the output $\tilde{\Lambda}_{\mathrm{odd}}(\alpha^{-i})$ will be determined for calculating errata values. In addition, the value $\tilde{\Lambda}_{\mathrm{odd}}(\alpha^{-i})$ is equal to $\alpha^{-i}\tilde{\Lambda}'(\alpha^{-i})$ because

$$\tilde{\Lambda}'(x) = \sum_{j=0}^{\lfloor \frac{u+v-1}{2} \rfloor} \tilde{\Lambda}_{2j+1} x^{2j}$$

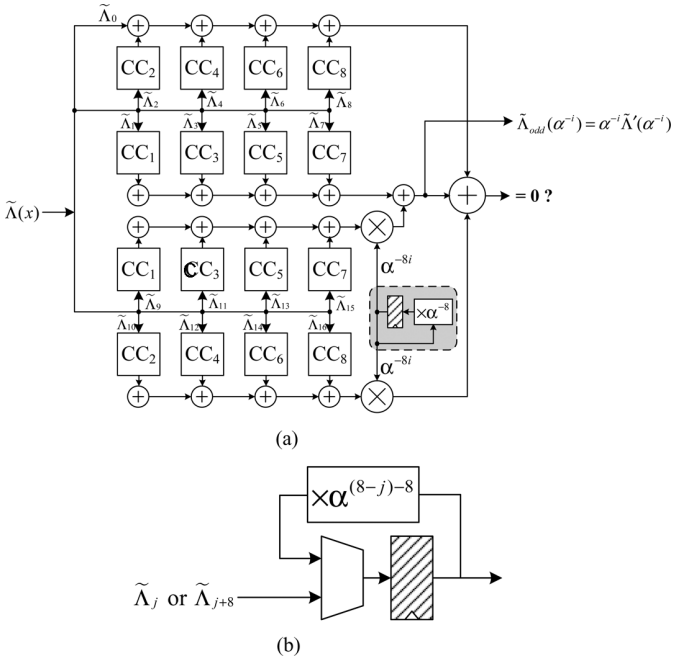

Fig. 7. (a) Chien search module with $d = 8$ and $t \leq 16$. (b) Chien cell CC_j.

$$= x^{-1} \cdot \sum_{j=0}^{\lfloor \frac{u+v-1}{2} \rfloor} \tilde{\Lambda}_{2j+1} x^{2j+1} \triangle x^{-1} \cdot \tilde{\Lambda}_{\mathrm{odd}}(x). \quad (33)$$

D. Errata Value Evaluator

In order to comply the data from Chien search, the errata value derived from Forney algorithm is modified as

$$e_l = \frac{\tilde{\omega}\left(\beta_l^{-1}\right)}{\tilde{\Lambda}'\left(\beta_l^{-1}\right)} = \frac{\beta_l^{-1}\tilde{\omega}\left(\beta_l^{-1}\right)}{\beta_l^{-1}\tilde{\Lambda}'\left(\beta_l^{-1}\right)} = \frac{\beta_l^{-1}\tilde{\omega}\left(\beta_l^{-1}\right)}{\tilde{\Lambda}_{\mathrm{odd}}\left(\beta_l^{-1}\right)} \quad (34)$$

where β_l^{-1} indicates the lth root of $\tilde{\Lambda}(x)$. The corresponding architecture to calculate the term $\beta_l^{-1}\tilde{\omega}(\beta_l^{-1})$ with $d = 8$ and $t \leq 16$ is shown in Fig. 8, where the cell CC_j is identical to the jth Chien search cell. The difference is the initial value being $\tilde{\omega}_{j-1}$ instead of $\tilde{\Lambda}_j$ 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 α^{-i+d} will be written to the address α^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 $\mathrm{GF}(2^m)$ 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

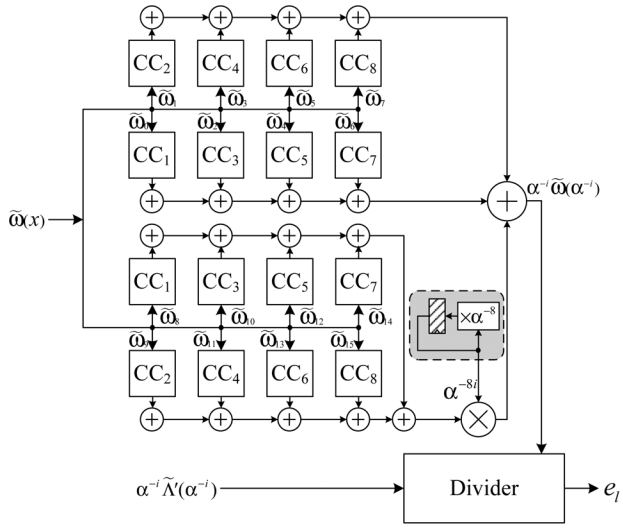

Fig. 8. Error value evaluator with $d = 8$ and $t \leq 16$.

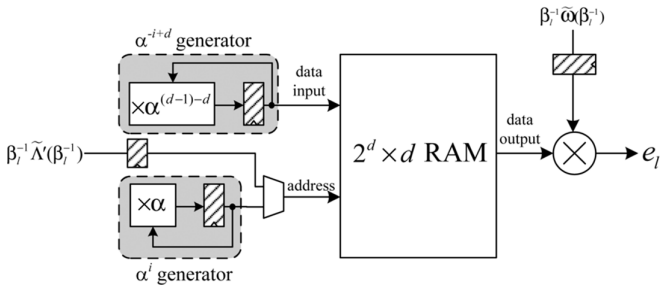

Fig. 9. Finite-field divider with on-the-fly inversion table.

TABLE II
Universal RS Decoder Chip Summary

Technology	0.18-μm 1P6M CMOS
Chip size	2.25 mm^2
Core size	1.21 mm^2
Gate count	46.4K
Embedded SRAM	8K bits (FIFO memory) 2K bits (Inversion table)
Supply voltage	1.62V$\sim$ 1.98V
Clock rate	160MHz
Power consumption	68.1mW (1.8V and 160MHz)

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), $\tilde{S}_1 \sim \tilde{S}_{16}$ can be calculated from the received codeword that is written into the FIFO memory as well, and $\tilde{S}_{17} \sim \tilde{S}_{32}$ are subsequently obtained from the same codeword read from the FIFO memory. The erasure generator produces the erasure information $\alpha^{l_1+8} \sim \alpha^{l_u+8}$ according to the erasure flag. Based on the inversionless Berlekamp–Massey algorithm, we implement the key equation solver to determine the erasure-locator polynomial $\lambda(x)$, the errata-locator polynomial $\tilde{\Lambda}(x)$, and the errata-evaluator polynomial $\tilde{\omega}(x)$. 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 $\tilde{\Lambda}(x)$ but also

TABLE III
COMPARISON AMONG RS DECODERS

Design	[25]*		[7]**	[24]*	Proposed***
Technology	0.18-μm		0.18-μm	0.25-μm	0.18-μm
(n,k)	fixed	configurable	configurable	configurable	configurable
t	8	≤ 8	≤ 8	≤ 8	≤ 16
Erasure	No	No	No	No	Yes
$GF(2^m)$	$m=8$	$m=8$	$m=8$	$m\leq 8$	$m\leq 8$
$f(x)$	fixed	fixed	fixed	arbitrary	arbitrary
Gates count	20K	23K	24K	44K	46K
Max. throughput	3.2Gb/s	3.2Gb/s	560Mb/s	48Mb/s	1.28Gb/s
Power consumption	N/A	N/A	48.4mW	N/A	68.1mW

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

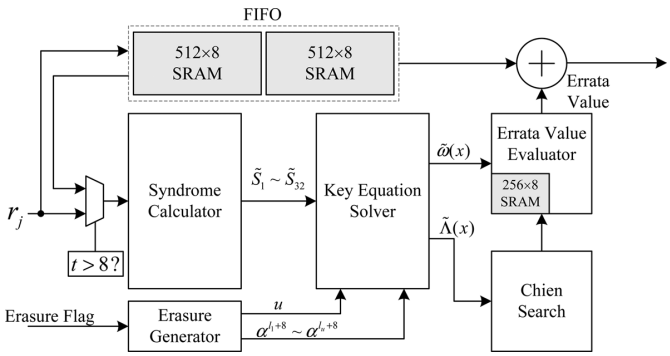

Fig. 10. Universal RS decoder architecture to correct both errors and erasures.

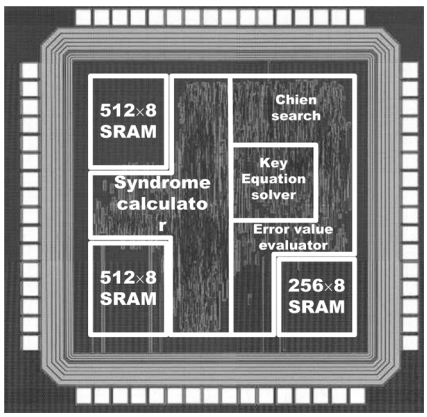

Fig. 11. 0.18-μm universal RS decoder chip photo.

generates $\beta_l^{-1}\tilde{\Lambda}'(\beta_l^{-1})$ 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 $\mathrm{GF}(2^8)$ mode, the maximum measured throughput is $8\ \text{bits} \times 160\ \text{MHz} = 1.28\ \text{Gb/s}$ 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

[1] E. R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.
[2] G. D. Forney Jr., "On decoding BCH codes," *IEEE Trans. Inf. Theory*, vol. IT-11, no. 5, pp. 549–557, Oct. 1965.
[3] L. Song, M. L. Yu, and M. S. Shaffer, "A 10 Gb/s and 40 Gb/s forward-error-correction device for optical communications," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1565–1573, Nov. 2002.
[4] T. K. Truong, J. H. Jeng, and K. C. Hung, "Inversionless decoding of both errors and erasures of Reed-Solomon code," *IEEE Trans. Commun.*, vol. 46, pp. 973–976, Aug. 1998.
[5] H. C. Chang, C. B. Shung, and C. Y. Lee, "A Reed-Solomon product-code (RS-PC) decoder chip for DVD applications," *IEEE J. Solid-State Circuits*, vol. 36, no. 2, pp. 229–237, Feb. 2001.
[6] H.-C. Chang, C.-C. Chung, C.-C. Lin, and C.-Y. Lee, "A 300 mhz Reed-Solomon decoder chip using inversionless decomposed architecture for euclidean algorithm," in *28th Eur. Solid-State Circuits Conf. (ESSCIRC)*, Florence, Italy, 2002, pp. 519–522.
[7] H.-Y. Hsu, J.-C. Yeo, and A.-Y. Wu, "Multi-symbol-sliced dynamically reconfigurable Reed-Solomon decoder design based on unified finite-field processing element," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 489–500, May 2006.
[8] *Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television*, ETSI Std. EN 300 744, 1998, Rev. 1.1.2.
[9] *Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for 11/12 GHz Satellite Services*, ETSI Std. EN 300 421, 1997, Rev. 1.1.2.
[10] *Digital Video Broadcasting (DVB); DVB Specification for Data Broadcasting*, ETSI Std. EN 301 192, 2008, Rev. 1.4.2.
[11] *Digital Multiprogramme Systems for Television Sound and Data Services for Cable Distribution*, ITU-T Std. J.83, 1997.
[12] *Forward Error Correction for Submarine Systems*, ITU-T Std. G.975, 2000.

[13] T. Tanzawa, T. Tanaka, K. Takeuchi, R. Shirota, S. Aritome, H. Watanabe, G. Hemink, K. Shimizu, S. Sato, Y. Takeuchi, and K. Ohuchi, "A compact on-chip ECC for low cost flash memories," *IEEE J. Solid-State Circuits*, vol. 32, no. 5, pp. 662–669, May 1997.
[14] L. Song, K. K. Parhi, I. Kuroda, and T. Nishitani, "Hardware/software codesign of finite field datapath for low energy Reed-Solomon codecs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 4, pp. 160–172, Apr. 2000.
[15] C.-C. Lin, F.-K. Chang, H.-C. Chang, and C.-Y. Lee, "A universal VLSI architecture for bit-parallel computation in $\mathrm{GF}(2^m)$," in *Proc. IEEE Asia-Pacific Conf. Circuits Syst.*, Dec. 2004, pp. 229–232.
[16] R. J. McEliece, *Finite Field for Computer Scientists and Engineers*. Boston, MA: Kluwer, 1987.
[17] E. Berlekamp, "On decoding binary Bose-Chaudhuri-Hocquenghem codes," *IEEE Trans. Inf. Theory*, vol. IT-11, pp. 577–579, Oct. 1965.
[18] J. Massey, "Step-by-step decoding of the Bose-Chaudhuri-Hocquenghem codes," *IEEE Trans. Inf. Theory*, vol. IT-11, pp. 580–585, Oct. 1965.
[19] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding Goppa codes," *Inf. Contr.*, vol. 27, pp. 87–99, 1975.
[20] F.-K. Chang, C.-C. Lin, H.-C. Chang, and C.-Y. Lee, "Universal architectures for Reed-Solomon error-and-erasure decoder," in *Proc. IEEE Asia Solid State Circuits Conf. (ASSCC)*, Nov. 2005, pp. 125–128.
[21] H. Burton, "Inversionless decoding of binary BCH codes," *IEEE Trans. Inf. Theory*, vol. IT-17, pp. 464–466, Jul. 1971.
[22] J. H. Jeng and T. K. Truong, "On decoding of both errors and erasures of a Reed-Solomon code using an inverse-free Berlekamp-Massey algorithm," *IEEE Trans. Commun.*, vol. 47, no. 10, pp. 1488–1494, Oct. 1999.
[23] H. C. Chang, "Research on Reed-Solomon decoder-design and implementation," Ph.D. dissertation, National Chiao Tung Univ., Hsinchu, Taiwan, 2002.
[24] J. C. Huang, C. M. Wu, M. D. Shieh, and C. H. Wu, "An area-efficient versatile Reed-Solomon decoder for ADSL," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, June 1999, pp. 517–520.
[25] M.-D. Shieh, Y.-K. Lu, S.-M. Chung, and J.-H. Chen, "Design and implementation of efficient Reed-Solomon decoders for multi-mode applications," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2006, pp. 289–292.

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/DE1 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.

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.

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.

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.