IEICE TRANS. INF. & SYST., VOL.E93–D, NO.7 JULY 2010

PAPER

High-Speed Low-Complexity Architecture for Reed-Solomon Decoders

Yung-Kuei LU[†], *Student Member and* **Ming-Der SHIEH**[†a)], *Member*

SUMMARY This paper presents a high-speed, low-complexity VLSI architecture based on the modified Euclidean (ME) algorithm for Reed-Solomon decoders. The low-complexity feature of the proposed architecture is obtained by reformulating the error locator and error evaluator polynomials to remove redundant information in the ME algorithm proposed by Truong. This increases the hardware utilization of the processing elements used to solve the key equation and reduces hardware by 30.4%. The proposed architecture retains the high-speed feature of Truong's ME algorithm with a reduced latency, achieved by changing the initial settings of the design. Analytical results show that the proposed architecture has the smallest critical path delay, latency, and area-time complexity in comparison with similar studies. An example RS(255,239) decoder design, implemented using the TSMC 0.18 μm process, can reach a throughput rate of 3 Gbps at an operating frequency of 375 MHz and with a total gate count of 27,271.

key words: *channel decoder, modified Euclidean algorithm, Reed-Solomon codes, VLSI architectures*

1. Introduction

Reed-Solomon (RS) code possesses an excellent capability of correcting both random and burst errors and is widely used in digital communication and storage systems [1], [3]. Many RS decoding algorithms and architectures have been proposed [3]–[14]. A syndrome-based RS decoder generally consists of three main blocks: the syndrome computation (SC) unit, the key equation solver (KES) unit, and the Chien search and error evaluation (CSEE) unit.

The KES unit used to find the error locator and error evaluator polynomials is the most critical part in the design of RS decoders. The key equation is generally solved by employing the Berlekamp-Massey (BM) algorithm [2]–[4], [18] or the modified Euclidean (ME) algorithm [5]–[14]. Compared to the architecture derived based on the ME algorithm, the BM architecture is conventionally thought of as being irregular and having a longer critical path delay, although it uses simpler computations to find the error locator polynomial. Criticisms of the irregular architecture and longer delay of the BM algorithm have recently subsided due to the development of the reformulated inversionless BM (RiBM) architecture [4]. For a t-error-correcting RS code, the RiBM architecture is made up of $3t + 1$ identical basic cells with a critical path delay of $T_{\text{mult}} + T_{\text{add}}$, where T_{mult} and T_{add} denote the delays of the finite-field multiplier (FFM) and adder, respectively.

The architecture based on the conventional ME algorithm [5] is regular, but the hardware cost is high due to the required degree computation and comparison circuit. High-speed ME architectures using pipelined multipliers to reduce the critical path delay were presented in [8] and [9]. However, the architectures are not area-efficient and require a larger number of clock cycles to solve the key equation. Truong *et al.* [6] presented a fast decoding algorithm based on the ME algorithm together with the BM algorithm to remove the need for degree computation. The corresponding architecture [7] has a simple control circuit and can operate at high data rates with a critical path delay of $T_{\text{mult}} + T_{\text{add}}$. The main drawback of this architecture is its high hardware requirement of $4t + 2$ basic cells. Recently, Baek and Sunwoo presented a low-complexity architecture using the degree computationless modified Euclidean (DCME) algorithm [10]. Their following studies [11], [12] further reduced the hardware requirement to $3t$ basic cells and shortened the latency to $2t - 1$ clock cycles for solving the key equation. However, there is an extra 2-to-1 multiplexer delay in the critical path, which is longer than that presented in [7].

In this work, we reformulate Truong's ME algorithm in [6] and then propose a high-speed, low-complexity architecture for the RS decoder design. The hardware complexity is reduced by improving the hardware utilization of the basic cells using the reformulated ME algorithm. With the proposed high-utilization arrangement scheme, we can reduce the number of required basic cells in [7] from $4t + 2$ to $3t - 1$. Compared to the results in [12], the proposed architecture has a simpler control circuit and can achieve a higher clock rate with a shorter critical path delay of $T_{\text{mult}} + T_{\text{add}}$. Moreover, we change the initial settings in the proposed architecture to save one iteration when solving the key equation. One fewer iteration potentially lowers the hardware requirement for some specific applications, such as the area-efficient design in [14]. Note that the proposed architecture can operate at a speed comparable to that of the RiBM architecture, with one fewer iteration and lower hardware requirements. Experimental results show that the prototype circuit, targeting RS(255,239) decoder, can operate at 375 MHz with a total gate count of 27,271 based on the TSMC 0.18 μm process.

The rest of this paper is organized as follows. Section 2 reviews the background information and describes the reformulated Truong's ME algorithm. Section 3 presents the

Manuscript received November 9, 2009.
Manuscript revised February 4, 2010.

[†]The authors are with the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C.
a) E-mail: shiehm@mail.ncku.edu.tw

DOI: 10.1587/transinf.E93.D.1824

proposed high-speed, low-complexity RS decoder design. The performance evaluation and comparisons with related work are shown in Sect. 4. Section 5 concludes this work.

2. Background and Reformulated ME Algorithm

This section gives a brief review of the fast ME algorithm presented by Truong *et al.* [6], which is denoted as the ME-T algorithm hereafter. Then, we describe how to reformulate the ME-T algorithm to reduce the hardware resources and change the initial settings to shorten the latency required for solving the key equation.

2.1 Background Information

Let $S(x) = s_0 + s_1 x + \ldots + s_{2t-1}x^{2t-1}$ denote the syndrome polynomial, where s_i, $0 \le i \le 2t-1$, are the computed syndrome values. Given $S(x)$, the error locator polynomial $\Lambda(x)$ and the error evaluator polynomial $\Omega(x)$ can be obtained using the ME algorithm or the BM algorithm to solve the key equation:

$$\Lambda(x)S(x) \equiv \Omega(x) \bmod x^{2t}. \tag{1}$$

The proposed architecture is based on the reformulated ME-T algorithm described later. For completeness, we briefly review the ME-T algorithm [6] below. Based on the ME algorithm, Truong *et al.* presented a new decoding algorithm by combining the mechanism of the BM algorithm to avoid polynomial division and field element inversion in conventional Euclidean [1] and BM [2] algorithms, respectively. Given an additional variable l, which is used in the BM algorithm, the ME-T algorithm is stated as follows:

(T.1) Initialization:

$$\Omega^{(a)}(x) = x^{2t},\ \Omega^{(b)}(x) = S(x),\ \Lambda^{(a)}(x) = 0,\ \Lambda^{(b)}(x) = 1,\ k = 0,\ l = 0; \tag{2}$$

(T.2) Polynomial Updates:

$$\Omega^{(b)}(x) = x\Omega^{(b)}(x),\ \Lambda^{(b)}(x) = x\Lambda^{(b)}(x); \tag{3}$$

$$u = \Omega^{(b)}_{2t},\ v = \Omega^{(a)}_{2t}; \tag{4}$$

$$\Omega^{(c)}(x) = u\Omega^{(a)}(x) + v\Omega^{(b)}(x),\quad \Lambda^{(c)}(x) = u\Lambda^{(a)}(x) + v\Lambda^{(b)}(x); \tag{5}$$

If $u \neq 0$ and $2l \le k$, then

$$\Omega^{(a)}(x) = \Omega^{(b)}(x),\ \Lambda^{(a)}(x) = \Lambda^{(b)}(x),\ l = k+1-l; \tag{6}$$

$$\Omega^{(b)}(x) = \Omega^{(c)}(x),\ \Lambda^{(b)}(x) = \Lambda^{(c)}(x); \tag{7}$$

(T.3) $k = k + 1$. If $k \le 2t - 1$, then go to (T.2).
(T.4) Output: $\Omega^{(b)}(x)$, $\Lambda^{(b)}(x)$.

In the algorithm, the superscripts a and b denote the previous and current states of polynomials, respectively. For example, $\Omega^{(a)}(x)$ is the dividend polynomial and $\Omega^{(b)}(x)$ is the corresponding divisor polynomial in each iteration. The superscript c in (5) represents the intermediate result of the polynomial. The variables u and v are the $(2t)^{th}$ coefficients, i.e., leading coefficients, of $\Omega^{(b)}(x)$ and $\Omega^{(a)}(x)$, respectively. The basic idea of this algorithm is to produce a zero coefficient at the x^{2t} term of $\Omega^{(c)}(x)$ by performing (3), (4), and (5) in each iteration. As a result, the ME-T algorithm eliminates the degree computation and comparison in conventional ME algorithms [5], [8]. Note that the condition $u \neq 0$ and $2l \le k$ is checked in (6) to avoid producing leading zero coefficients for both $\Omega^{(a)}(x)$ and $\Omega^{(b)}(x)$ in the next iteration. This implies that the zero polynomial will not appear when performing (5). After $2t$ iterations, the desired $\Omega^{(b)}(x)$ and $\Lambda^{(b)}(x)$ are obtained simultaneously.

2.2 Reformulation of the ME-T Algorithm

From the ME-T algorithm, the length of consecutive zero coefficients of $\Omega^{(b)}(x)$ grows incrementally, starting from the constant term, as the iterations proceed. Since the zero coefficients are not used in later iterations, they can be discarded. Before describing how to reformulate the ME-T algorithm for efficient hardware implementation, we introduce two properties, which are helpful for removing the redundant zero coefficients.

For clarity, we use the notations $\tilde{\Omega}(x)$ and $\tilde{\Lambda}(x)$ hereafter to denote the error evaluator and error locator polynomials, respectively, in the conventional ME algorithm [5] to distinguish them from those in the ME-T algorithm. As stated in [5] and [15], the following two properties hold during the iterations of the conventional ME algorithm. Note that except the first leading zero coefficient of the polynomials, the following zero coefficients, if any, are reserved for the next iteration; therefore, only one leading zero coefficient is removed in each iteration.

Property 1: Since the sum of the degrees of $\tilde{\Omega}^{(a)}(x)$ and $\tilde{\Omega}^{(b)}(x)$ is decreased by one at each iteration, the following equation holds at the beginning of the k-th iteration:

$$\deg(\tilde{\Omega}^{(a)}(x)) + \deg(\tilde{\Omega}^{(b)}(x)) = 4t - 1 - k \tag{8}$$

Property 2: The sum of the degrees of $\tilde{\Omega}^{(a)}(x)$ and $\tilde{\Lambda}^{(b)}(x)$ remains constant during the iterations of the conventional ME algorithm. It was stated in [15] that:

$$\deg(\tilde{\Omega}^{(a)}(x)) + \deg(\tilde{\Lambda}^{(b)}(x)) = 2t. \tag{9}$$

From the ME-T algorithm, one can show that the degrees of $\Omega^{(b)}(x)$ and $\Lambda^{(b)}(x)$ are $2t-1$ and k, respectively, at the beginning of the k-th iteration for $0 \le k \le 2t$. Together with Properties 1 and 2, we can derive the following equation:

$$\deg(\Omega^{(b)}(x)) - \deg(\tilde{\Omega}^{(b)}(x)) = \deg(\Lambda^{(b)}(x)) - \deg(\tilde{\Lambda}^{(b)}(x)) \tag{10}$$

This implies that the left-shift operation (3) has the same effect on both $\Omega^{(b)}(x)$ and $\Lambda^{(b)}(x)$ during the iterations of the ME-T algorithm. That is, $\Omega^{(b)}(x)$ and $\Lambda^{(b)}(x)$ can be represented as:

$$\Omega^{(b)}(x) = x^{p(k)}\tilde{\Omega}^{(b)}(x) \text{ and } \Lambda^{(b)}(x) = x^{p(k)}\tilde{\Lambda}^{(b)}(x), \tag{11}$$

where $p(k) \geq 0$. Thus, $\Omega^{(b)}(x)$ consists of $p(k)$ consecutive zero coefficients, starting from the constant term, at the beginning of the k-th iteration. The $p(k)$ terms can be treated as redundant information, which can be removed with no information loss. The degree of $\tilde{\Lambda}^{(b)}(x)$ is no more than t during each iteration because it is a non-decreasing value and is equal to t at the end of $2t$ iterations [5]. This implies that $p(k) \geq k - t$; therefore, we can derive the following equation:

$$\deg(\tilde{\Omega}^{(b)}(x)) = \deg(\Omega^{(b)}(x)) - p(k) \leq 3t - 1 - k. \quad (12)$$

Note the value of $p(2t)$ is equal to t. Knowing that there are $p(k)$ consecutive zero coefficients, we define a new polynomial $\hat{\Omega}^{(*)}(x)$ as the concatenation of the two polynomials $\Omega^{(*)}(x)$ and $\Lambda^{(*)}(x)$:

$$\hat{\Omega}^{(*)}(x) = \Omega^{(*)}(x) \cdot x^{t+1} + \Lambda^{(*)}(x) \quad (13)$$

The notation $\hat{\Omega}^{(*)}(x)$ is used to represent $\hat{\Omega}^{(a)}(x)$ or $\hat{\Omega}^{(b)}(x)$ when appropriate. According to (12), one can show that the useful information of $\Omega^{(b)}(x)$ and $\Lambda^{(b)}(x)$, i.e, $\tilde{\Omega}^{(b)}(x)$ and $\tilde{\Lambda}^{(b)}(x)$, respectively, does not overlap during iterations. In other words, the coefficients of $\Omega^{(b)}(x)$ overwritten by those of $\Lambda^{(b)}(x)$ in the reformulated algorithm are the redundant zero coefficients mentioned above. Therefore, the concatenated polynomial $\hat{\Omega}(x)$ is enough to find the error locator and error evaluator polynomials. From (13), the degrees of $\hat{\Omega}^{(a)}(x)$ and $\hat{\Omega}^{(b)}(x)$ are $3t + 1$ and $3t$, respectively, because $\deg(\Omega^{(a)}(x)) = 2t$ and $\deg(\Omega^{(b)}(x)) = 2t - 1$.

Since the syndrome polynomial $S(x)$ is available before performing the ME-T algorithm, we can use the computed results of the first iteration as its initial settings to save one iteration. This implies that the total number of iterations required to solve the key equation can be reduced from $2t$ to $2t - 1$. The benefit of reducing the number of iterations by one is described later in the performance evaluation. Using the defined concatenated polynomial $\hat{\Omega}(x)$ together with the initial settings, the reformulated ME-T algorithm, denoted as the ME-R algorithm, is proposed below:

(R.1) Initialization:

$$\hat{\Omega}^{(a)}(x) = \begin{cases} x^{3t+1}, & \text{if } s_{2t-1} = 0 \\ x^{t+2}S(x) + x, & \text{if } s_{2t-1} \neq 0 \end{cases},$$
$$\hat{\Omega}^{(b)}(x) = [x^{t+2}S(x) + x] \bmod x^{3t+1}, \quad (14)$$
$$k = 1,\ l = \begin{cases} 0, & \text{if } s_{2t-1} = 0 \\ 1, & \text{if } s_{2t-1} \neq 0 \end{cases}.$$

(R.2) Polynomial Updates:

$$\hat{\Omega}^{(b)}(x) = x\hat{\Omega}^{(b)}(x); \quad (15)$$
$$u = \hat{\Omega}^{(b)}_{3t+1},\ v = \hat{\Omega}^{(a)}_{3t+1}; \quad (16)$$
$$\hat{\Omega}^{(c)}(x) = u\hat{\Omega}^{(a)}(x) + v\hat{\Omega}^{(b)}(x); \quad (17)$$
If $u \neq 0$ and $2l \leq k$, then
$$\hat{\Omega}^{(a)}(x) = \hat{\Omega}^{(b)}(x),\ l = k + 1 - l; \quad (18)$$
$$\hat{\Omega}^{(b)}(x) = \hat{\Omega}^{(c)}(x); \quad (19)$$

(R.3) $k = k + 1$. If $k \leq 2t - 1$, then go to (R.2).

Table 1 Example for showing iterations of the ME-R algorithm.

k	l	$\hat{\Omega}^{(a)}(x)$	$\hat{\Omega}^{(b)}(x)$
		$(x^7,x^6,x^5,x^4,x^3,x^2,x^1,x^0)$	$(x^6,x^5,x^4,x^3,x^2,x^1,x^0)$
1	1	$(\alpha^3,\alpha^4,\alpha^3,\alpha^6,0,0,1,0)$	$(\alpha^4,\alpha^3,\alpha^6,0,0,1,0)$
2	1	$(\alpha^3,\alpha^4,\alpha^3,\alpha^6,0,0,1,0)$	$(\alpha^5,\alpha^6,\alpha^3,0,\alpha^3,\alpha^4,0)$
3	2	$(\alpha^5,\alpha^6,\alpha^3,0,\alpha^3,\alpha^4,0,0)$	$(0,\alpha^5,\alpha^4,\alpha^6,1,\alpha^5,0)$
4	2	$(\alpha^5,\alpha^6,\alpha^3,0,\alpha^3,\alpha^4,0,0)$	$(\alpha^3,\alpha^2,\alpha^4,\alpha^5,\alpha^3,0,0)$

(R.4) Output: $\hat{\Omega}^{(b)}_i$, for $t \leq i \leq 3t$.

Note that the symbol s_{2t-1} is the leading coefficient of the syndrome polynomial $S(x)$, and that the initial settings in (14) can be easily obtained in the hardware implementation. At the end of the ME-R algorithm, the coefficients of the output $\hat{\Omega}^{(b)}_i$ for $t \leq i \leq 2t$ and $2t+1 \leq i \leq 3t$, respectively, represent the desired error locator and error evaluator polynomials, which are the same as the useful information of $\Lambda^{(b)}(x)$ and $\Omega^{(b)}(x)$ derived from the ME-T algorithm. Compared to the original ME-T algorithm, the presented ME-R algorithm significantly reduces the corresponding hardware requirement without degrading the operation speed, as described later. Moreover, a simple example is given below to demonstrate the proposed ME-R algorithm.

Example: Consider a 2-error-correcting RS(7,3) code over GF(2^3), which is constructed by the primitive polynomial $x^3 + x + 1$ with a primitive element α. Let the generator polynomial and the transmitted information polynomial be $G(x) = x^4 + \alpha^3x^3 + x^2 + \alpha x + \alpha$ and $D(x) = \alpha^2x^2 + \alpha^5x + \alpha^2$, respectively. According to the systematic encoding, the corresponding codeword $C(x)$ can be derived as $C(x) = \alpha^2x^6 + \alpha^5x^5 + \alpha^2x^4 + \alpha^3x^3 + \alpha^5x^2$. Assume the error polynomial is $E(x) = \alpha^5x^5 + \alpha x^3$, the received word can be expressed as $R(x) = C(x) + E(x) = \alpha^2x^6 + \alpha^2x^4 + x^3 + \alpha^5x^2$. Then, we can calculate the syndrome polynomial $S(x) = \alpha^3x^3 + \alpha^4x^2 + \alpha^3x + \alpha^6$. Table 1 shows the initial conditions and the computed values in each iteration of the proposed ME-R algorithm. In this table, we use the n-tuple representation to denote a polynomial with degree $n - 1$. For instance, $(\alpha^4, \alpha^3, \alpha^6, 0, 0, 1, 0)$ represents the coefficients of $\hat{\Omega}^{(b)}(x)$ scanning from the high-order term. After 3 iterations, i.e., $k = 4$, the error locator polynomial and the error evaluator polynomial are computed as $\alpha^4x^2 + \alpha^5x + \alpha^3$ and $\alpha^3x + \alpha^2$, respectively. Finally, we can find the error locations α^3 and α^5 by the Chien search and use the Forney algorithm to obtain the corresponding error values α and α^5, respectively.

3. Proposed High-Speed, Low-Complexity Architecture

Since the degree of the defined polynomial $\hat{\Omega}(x)$ is restricted to $3t + 1$ in the ME-R algorithm, the corresponding architecture can be constructed using $3t + 2$ basic cells together with a simple control circuit. Note that the design based on the original ME-T algorithm takes $4t + 2$ cells [7]. The developed architecture inherits the high-speed feature of the

ME-T algorithm; therefore, it can operate at a higher speed than those of related studies [10]–[12] that applied various refinement strategies to the conventional ME algorithm.

3.1 Basic Cell Design and Boundary Cell Simplification

From (15)–(19) of the ME-R algorithm, the basic cell design is shown in Fig. 1 (a), in which $\hat{\Omega}_i$ denotes the coefficient of the i-th term of $\hat{\Omega}(x)$; u and v are the leading coefficients of $\hat{\Omega}^{(b)}(x)$ and $\hat{\Omega}^{(a)}(x)$, respectively. The control signal w is used to determine the previous state $\hat{\Omega}^{(a)}(x)$ for the next iteration, as shown in (18). The critical path delay of the basic cell is equal to $T_{\text{mult}} + T_{\text{add}}$, which is independent of t.

Since $\hat{\Omega}^{(c)}_{3t+1}$ in (17) is always reduced to zero, there is no need to store this value. As a result, the circuit used to update $\hat{\Omega}^{(b)}_{3t+1}$ can be removed and the basic cell is further simplified to include only one register and one multiplexer for updating $\hat{\Omega}^{(a)}_{3t+1}$. Figure 1 (b) depicts the simplified boundary cell, denoted as SBC_0, which includes the simplified basic cell to find $\hat{\Omega}^{(a)}_{3t+1}$, the $(3t+1)^{th}$ term of $\hat{\Omega}^{(a)}(x)$, and the control circuit adopted to check whether the conditions $u \neq 0$ and $2l \leq k$ are satisfied. The control unit includes a counter for the variable k and a circuit for checking the conditions. When the conditions $u \neq 0$ and $2l \leq k$ are satisfied, the control signal w is set to 1; otherwise $w = 0$. Techniques such as those used in [4] and [16] can be employed to design the control unit so that it does not dominate the overall critical path delay of decoders for RS codes of practical interest. Hence, the circuit of checking these two conditions is simpler than those designed with the finite state machines presented in [9] and [19]. Moreover, using the new initial settings in (14), the constant terms $\hat{\Omega}^{(a)}_0$ and $\hat{\Omega}^{(b)}_0$ remain zero at each iteration; therefore, the cell for updating and storing $\hat{\Omega}^{(a)}_0$ and $\hat{\Omega}^{(b)}_0$ can be eliminated.

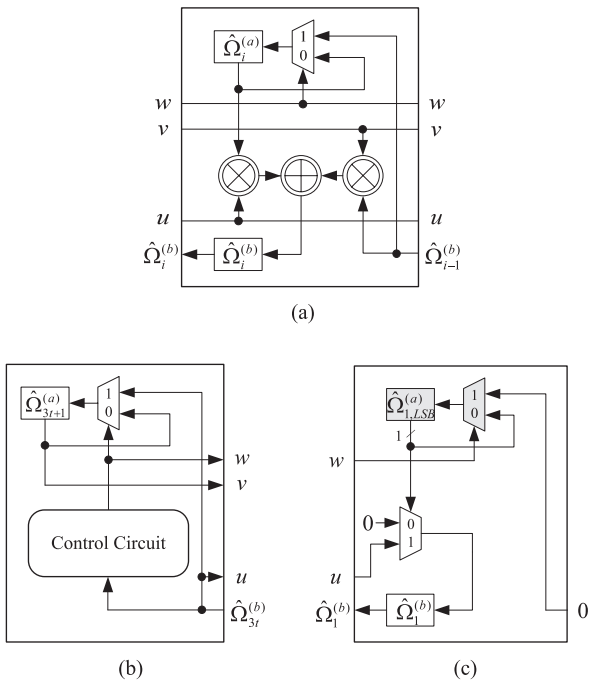

Fig. 1 (a) Basic cell (PE). (b) SBC_0: Simplified basic cell combined with the control circuit. (c) SBC_1: Simplified cell for storing and updating $\hat{\Omega}^{(a)}_1$ and $\hat{\Omega}^{(b)}_1$.

The condition that $\hat{\Omega}^{(a)}_0 = \hat{\Omega}^{(b)}_0 = 0$ always holds can be further applied to simplify the circuit for updating $\hat{\Omega}^{(a)}_1$ and $\hat{\Omega}^{(b)}_1$, described as follows: First, $\hat{\Omega}^{(b)}_0 = 0$ implies that $\hat{\Omega}^{(b)}_1 = u\hat{\Omega}^{(a)}_1$ in each iteration. From (18), the updated value of $\hat{\Omega}^{(a)}_1$ for the next iteration can thus be assigned as $w\hat{\Omega}^{(b)}_0 + \bar{w}\hat{\Omega}^{(a)}_1 = \bar{w}\hat{\Omega}^{(a)}_1$, where $\bar{w}$ denotes the complement of the 1-bit control signal w. Secondly, the data width of the associated register and multiplexer required for updating $\hat{\Omega}^{(a)}_1$ is only one bit as the initial value of $\hat{\Omega}^{(a)}_1$ is either 0 or 1, depending on whether s_{2t-1} is zero or not. This in turn indicates that the circuit used to update $\hat{\Omega}^{(b)}_1$ can be also simplified because of $\hat{\Omega}^{(b)}_1 = u\hat{\Omega}^{(a)}_1$. Thirdly, the hardware requirement of this simplified cell is much smaller than that of the basic cell because no multiplier is needed for updating $\hat{\Omega}^{(a)}_1$ and $\hat{\Omega}^{(b)}_1$. Figure 1 (c) shows the corresponding simplified cell, denoted as SBC_1, in which the symbol $\hat{\Omega}^{(a)}_{1,LSB}$ denotes the least significant bit (LSB) of $\hat{\Omega}^{(a)}_1$.

3.2 High-Speed, Low-Complexity Architecture

Figure 2 depicts the proposed architecture with its initial values indicated in rectangular boxes inside the basic cells, denoted as PE for short. The architecture consists of $3t-1$ basic cells together with two boundary cells, as shown in Fig. 1 (b) and Fig. 1 (c). The values in parentheses within the upper registers of PEs are the initial settings of $\hat{\Omega}^{(a)}(x)$ for $s_{2t-1} \neq 0$ and those outside the parentheses are for $s_{2t-1} = 0$,

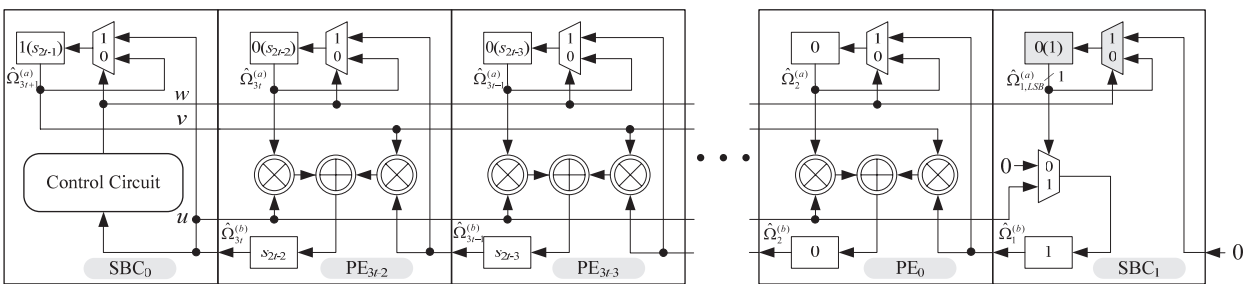

Fig. 2 VLSI architecture for the proposed ME-R algorithm.

as defined in (14). The three control signals, u, v, and w, are broadcasted to all the PE cells. Note that global control signals are inevitable in this kind of KES design.

As the iterations proceed, the coefficients of $\hat{\Omega}^{(a)}(x)$ and $\hat{\Omega}^{(b)}(x)$ are shifted to the left and updated based on the values of control signals. After $2t-1$ iterations, the desired error locator and error evaluator polynomials are obtained in PE_i for $t-2 \le i \le 2t-2$ and $2t-1 \le i \le 3t-2$, respectively, at the same time. The results can be directly used in the Forney algorithm [1] to compute the error values. The architecture designed based on the proposed ME-R algorithm can operate at very high speeds with a significant reduction in the number of building cells. One iteration is saved as compared to that in [7] developed from the original ME-T algorithm [6].

4. Performance Evaluation and Comparison

This section shows the performance evaluation of our development and comparisons with related work. For the analytical results, we used the RS(255,239) decoder, one of the most popular designs in practical applications, as the example design to demonstrate the effectiveness of the proposed work.

4.1 Complexity Analysis

Table 2 lists the area and time complexities of various KES designs derived from the ME algorithm. From Table 2, the proposed architecture has the smallest critical path delay ($T_{\text{mult}} + T_{\text{xor2}}$) and latency ($2t-1$ clock cycles) as compared with those of related studies [7], [10]–[12]. Note that the symbol T_{xor2} represents the delay time of a 2-input XOR gate, which is the same as the delay time T_{add} of finite-field addition. The reduced critical path delays in [8] and [9] were obtained using pipelined multipliers, which result in the highest hardware complexity and latency. Excluding the control circuit, the proposed ME-R architecture consists of $6t-2$ multipliers, $3t-1$ adders, $6t$ registers, and $5t+1$ multiplexers ($3t$ multiplexers are used with the control signal w to determine $\hat{\Omega}_i^{(a)}$, one multiplexer is employed in SBC_1 cell to update $\hat{\Omega}_1^{(b)}$, and the remaining $2t$ multiplexers are used for initializing the upper register in the leftmost $2t$ cells, as indicated in Fig. 2). Compared to the design in [11], the proposed design can operate faster with comparable hardware requirement. Note that the hardware requirement of a multiplier is much higher than that of a multiplexer.

To further verify the effectiveness of the proposed design, we used the cell library information in [17] to analyze the resulting area and time complexities of the various KES designs in GF(2^8), assuming that a widely used primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$ is adopted to construct the field. For simplicity, the cell delay and area requirement of each cell are normalized with respect to those of the 2-input AND gate as given in Table 3. The compiled results including the area-time (AT) complexity and improvement are shown in the right half of Table 2. The AT improvement is defined as $(AT_2 - AT_1)/AT_2$, where AT_2 and AT_1 denote the AT complexity of the related work and ours, respectively. For a fair comparison, we used the same kind of parallel FFM, as presented in [8], for each design.

Note that Lee [8] further pipelined the parallel FFM into 3 stages to reduce the critical path delay, with an overhead of 23 pipelined flip-flops. Since his following study [9] did not show the detailed structure of the 2-stage pipelined FFM, we suppose that the same kind of FFM was used, but with only 15 pipelined flip-flops. The critical path delay and the area requirement of the non-pipelined FFM, employed in the proposed architecture, can be estimated as $T_{\text{mult}} = T_{\text{and2}} + 5T_{\text{xor2}}$ and $A_{\text{mult}} = 64A_{\text{and2}} + 77A_{\text{xor2}}$, respectively. Note that the widths of finite-field adders, registers, and multiplexers listed in Table 2 are 8 bits, and the value of t is set to 8 for area complexity estimation of practical applications ($0 \le t \le 8$). From Table 2, we can see that the proposed architecture greatly reduces the derived AT complexity compared to those of the related works. In particular, the proposed architecture provides a 30.4% improvement in AT complexity compared to the design in [7], which uses the ME-T algorithm. Note that the proposed architecture has a shorter latency and fewer basic cells than

Table 2 Comparison of time and area complexities of various KES designs using the ME algorithm.

KES architecture	Critical path delay	Latency	Multipliers	Adders	Registers	MUXs	Time complexity[(1)]	Area complexity[(1)]	AT complexity	AT improvement
ME-R(proposed)	$T_{\text{mult}}+T_{\text{xor2}}$[(2)]	$2t$-1	$6t$-2	$3t$-1	$6t$	$5t$+1	7.72	12684	97921	-
ME-T [7]	$T_{\text{mult}}+T_{\text{xor2}}$	$2t$	$8t$+4	$4t$+2	$8t$+4	$4t$+2	7.72	18224	140689	30.4%
ME [8][(3)]	$3T_{\text{or2}}+T_{\text{xnor2}}+T_{\text{mux2}}$	$10t$	$8t$	$8t$	$78t$+4	$40t$+2	5.59	47736	266844	63.3%
pDCME [9][(3)]	$T_{\text{inv}}+T_{\text{and2}}+3T_{\text{mux2}}$	$10t$	$8t$	$4t$	$54t$	$20t$	4.16	35792	148895	34.2%
DCME [10]	$T_{\text{mux4}}+T_{\text{mult}}+T_{\text{xor2}}$	$2t$	$6t$+4	$3t$+2	$6t$+4	$18t$+12	9.25	16016	148148	33.9%
S-DCME [11]	$T_{\text{mux2}}+T_{\text{mult}}+T_{\text{xor2}}$	$2t$	$6t$	$3t$	$6t$	$2t$-1	8.71	12720	110791	11.6%
E-DCME [12]	$T_{\text{mux2}}+T_{\text{mult}}+T_{\text{xor2}}$	$2t$-1	$6t$	$3t$	$6t$	$6t$	8.71	13248	115390	15.1%

(1) The values are estimated in finite field GF(2^8) constructed from the primitive polynomial $x^8 + x^4 + x^3 + x^2 + 1$. Note that $t = 8$ is used to calculate the area complexity.
(2) T_{Gn} notes the delay time of a logic cell G with n inputs. Note that T_{xor2} is equal to T_{add}.
(3) Designs using pipelined multipliers to reduce the critical path delay.

Table 3 Normalized delay and area of employed standard cells.

	2-input AND	2-input XOR	2-input OR	2-input XNOR	2-to-1 MUX	4-to-1 MUX	DFF	INV
delay ratio	1	1.12	1.16	1.12	0.99	1.53	1.54	0.19
area ratio	1	2	1	2	2	5	4.25	0.5

Technology	TSMC 0.18 μm
Clock rate	375 MHz
Throughput	3000 Mbps
Gate count	27271 (w/o FIFO)
Core size	0.68×0.68 mm^2

Fig. 3 Layout view of the proposed RS decoder.

Table 4 Performance comparison with other RS(255,239) decoders.

KES architecture	ME [8]	pDCME [9]	DCME [10]	S-DCME [11]	E-DCME [12]	ME-R (Proposed)
Technology (μm)	0.13	0.13	0.25	0.25	0.18	0.18
# Gates in KES unit	102500	46200	21760	17800	18000	21338
# Gates in Decoder (w/o FIFO)	115500	53200	42213	-	-	27271
Clock rate (MHz)	770	660	200	200	200	375
Throughput (Mbps)	6160	5280	1600	1600	1600	3000
Latency of KES unit (cycles)	80	80	16	16	15	15
TSNT of KES unit (Mbps/gate)	0.060	0.114	0.141	0.173	0.123	0.195

the RiBM architecture for solving the key equation. One fewer iteration potentially lowers the hardware requirement for some specific applications. For example, instead of decreasing the value of the folding factor in the direct folding approach, Hsu [14] employed a pre-computation scheme to save one iteration so that the hardware utilization of the resulting folded architecture could be increased to lower the hardware cost.

4.2 Experimental Results and Comparisons

Based on the proposed architecture, an RS(255,239) decoder was coded in Verilog language and synthesized using Synopsys tools based on TSMC 0.18 μm 1.8 V CMOS technology. The post-layout simulation shows that our decoder design can operate at 375 MHz with a total gate count of 27,271. The layout view of the proposed RS(255,239) decoder designed with a first-in first-out (FIFO) buffer for storing input symbols is shown in Fig. 3, which has a core size of about $0.68 \times 0.68\,\text{mm}^2$. Table 4 shows comparisons between our design and those of related studies. To consider the scaling effect of fabrication technology, we adopt the definition of technology scaled normalized throughput rate (*TSNT*) in [14] as

Table 5 Comparison of time and area complexities of the proposed ME-R architecture and relative BM-based architectures.

KES architecture	ME-R	RiBM [4]	iBM [1] (Blahut)	iBM [1] (Berlekamp)
Critical path delay	$T_{\text{mult}}+T_{\text{xor2}}$	$T_{\text{mult}}+T_{\text{xor2}}$	$>2(T_{\text{mult}}+T_{\text{xor2}})$	$>2(T_{\text{mult}}+T_{\text{xor2}})$
Latency	$2t$-1	$2t$	$3t$	$2t$
Multipliers	$6t$-2	$6t$+2	$3t$+3	$5t$+3
Adders	$3t$-1	$3t$+1	$2t$+1	$3t$+1
Registers	$6t$	$6t$+2	$4t$+2	$6t$+2
MUXs[2]	$5t$+1	$3t$+1	t+1	$2t$+1
Time complexity	7.72	7.72	>15.44	>15.44
Area complexity	12684	13400	7458	11746
AT complexity	97921	103448	>115152	>181358
AT improvement	-	5.3%	>15.0%	>46.0%

[1] The hardware complexity and path delay are estimated in [4].
[2] A multiplexer has much smaller hardware complexity than a multiplier does.

$$TSNT = (\text{Throughput rate}) \times (\text{Tech.}/0.13\,\mu\text{m})/\#\,\text{Gates}.$$

As can be seen from the table, the proposed design has the best *TSNT* for the KES unit. Compared with the latest design [11], the proposed design can achieve about 12.7% improvements in the *TSNT* index. Note that the comparison of the total gate counts of decoders was made by excluding the FIFO required to buffer input symbols for error correction. In addition, the total gate count of the SC and CSEE units in [8] and [9] are slightly more than ours since their designs need extra pipelined registers and a pipelined multiplier in the CSEE unit to operate at higher clock rates.

Compared to the RiBM architecture, the proposed ME-R architecture possesses one fewer iteration and lower hardware requirement with the same critical path delay, as shown in Table 5. The main differences between these two architectures are the opposite data flow and different initial settings. The opposite data flow comes from the fact that the ME algorithm processes the polynomials starting from the term with the highest degree, while the RiBM algorithm is from the one with lowest degree. More importantly, the new initial values of the ME-R algorithm can be manipulated to reduce the number of iterations and to further simplify the boundary cells of the corresponding architecture. In contrast, the number of iterations of the RiBM algorithm is hard to directly reduce to $2t-1$ because complicated operations are required for computing the new initial setting, which will result in high area overhead. Even though the boundary cell in the RiBM architecture could be simplified, the total required basic cells are more than those of the ME-R architecture. In consequence, the proposed ME-R algorithm leads to a high-speed, low-complexity architecture. For completeness, Table 5 lists the information of area and time complexities of related architectures developed by applying the concept of the inversionless BM (iBM) algorithm in [20] to the typical BM algorithms in [21] and [22]. Note that the hardware complexity and the path delay estimation of these two iBM architectures were derived in [4]. We choose the two iBM architectures because their design

constraints are similar to ours. Although the two iBM architectures take a smaller hardware resource than ours, they demand a much larger critical path delay and longer latency for solving the key equation. Table 5 reveals that the proposed ME-R architecture has the shortest latency and the smallest AT complexity.

Since the proposed architecture is a regular array structure, we can apply the folding technique to develop an area-efficient architecture, which consist of about $\lceil(6t-2)/f\rceil$ multipliers, $\lceil(3t-1)/f\rceil$ adders, $6t$ registers, and $2t+2+\lceil(3t-1)/f\rceil$ multiplexers, where $\lceil\ \rceil$ denotes the ceiling function and f is the folding factor. Compared with the architecture in [14] which requires $\lceil 8t/f\rceil$ multipliers, $\lceil 4t/f\rceil$ adders, $8t$ registers, and $\lceil 8t/f\rceil$ multiplexers, the proposed folded architecture has lower hardware complexity with a comparable operating speed. The proposed folded architecture is also more cost-effective than the pipeline recursive structures in [13] and [19] if the pipelined multipliers used in their work are also employed in our design.

5. Conclusion

This paper presented a high-speed, low-complexity VLSI architecture based on the proposed ME-R algorithm and evaluated its performance for an RS(255,239) decoder design. The proposed architecture can reduce hardware by 30.4% as compared to that developed using the original ME-T algorithm without sacrificing throughput. We also showed how to shorten the latency when solving the key equation using the proposed architecture. Experimental results demonstrated the effectiveness of the developed algorithm and VLSI architecture. The proposed architecture is well suited for high-speed low-complexity RS decoder design.

Acknowledgments

This work was supported in part by the National Science Council of R.O.C. under contract NSC 96-2221-E-006-296.

References

[1] S.B. Wicker and V.K. Bhargava, Reed-Solomon Codes and Their Applications, IEEE Press, New York, 1994.
[2] J.L. Massey, "Shift-register synthesis and BCH decoding," IEEE Trans. Inf. Theory, vol.IT-15, no.1, pp.122–127, Jan. 1969.
[3] 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.229–238, Feb. 2001.
[4] D.V. Sarwate and N.R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.9, no.5, pp.641–655, Oct. 2001.
[5] H.M. Shao, T.K. Truong, L.J. Deutsch, J.H. Yuen, and I.S. Reed, "A VLSI design of a pipeline Reed-Solomon decoder," IEEE Trans. Comput., vol.C-34, no.5, pp.393–402, May 1985.
[6] T.K. Truong, J.H. Jeng, and T.C. Cheng, "A new decoding algorithm for correcting both erasures and errors of Reed-Solomon codes," IEEE Trans. Commun., vol.51, no.3, pp.381–388, March 2003.
[7] Y.W. Chang, T.K. Truong, and J.H. Jeng, "VLSI architecture of modified Euclidean algorithm for Reed-Solomon code," ELSEVIER J. Inform. Sciences, vol.155, pp.139–150, May 2003.
[8] H. Lee, "High-speed VLSI architecture for parallel Reed-Solomon decoder," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.11, no.2, pp.288–294, April 2003.
[9] S. Lee, H. Lee, J. Shin, and J.S. Ko, "A high-speed pipelined degree-computationless modified Euclidean algorithm architecture for Reed-Solomon decoder," Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'2007), pp.901–904, May 2007.
[10] J.H. Baek and M.H. Sunwoo, "New degree computationless modified Euclidean algorithm and architecture for Reed-Solomon decoder," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.14, no.8, pp.915–920, Aug. 2006.
[11] J.H. Baek and M.H. Sunwoo, "Simplified degree computationless modified Euclid's algorithm and its architecture," Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'2007), pp.905–908, May 2007.
[12] J.H. Baek and M.H. Sunwoo, "Enhanced degree computationless modified Euclid's algorithm for Reed-Solomon decoders," IET Electronics Letter, vol.43, no.3, pp.175–176, Feb. 2007.
[13] H. Lee, "A high-speed low-complexity Reed-Solomon decoder for optical communications," IEEE Trans. Circuits Syst. II, Express Briefs, vol.52, no.8, pp.461–465, Aug. 2005.
[14] H.Y. Hsu, A.Y. Wu, and J.C. Yeo, "Area-efficient VLSI design of Reed-Solomon decoder for 10 GBase-LX4 optical communication systems," IEEE Trans. Circuits Syst. II, Express Briefs, vol.43, no.4, pp.1019–1027, Nov. 2006.
[15] R.J. McEliece, The Theory of Information and Coding: A Mathematical Framework for Communication, Addison-Wesley, MA, 1977.
[16] C.H. Wu, C.M. Wu, M.D. Shieh, and Y.T. Hwang, "High-speed, low-complexity systolic designs of novel iterative division algorithms in GF(2^m)," IEEE Trans. Comput., vol.53, no.3, pp.375–380, March 2004.
[17] Artisan Components, TSMC 0.18-μm Process 1.8-Volt SAGE-X^{TM} Standard Cell Library Databook, Sunnyvale, CA, 2003.
[18] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," Proc. IEEE Workshop on Signal Process. Syst. (SIPS '06), pp.303–308, Oct. 2006.
[19] B. Yuan, Z.F. Wang, L. Li, M.L. Gao, J. Sha, and C. Zhang, "Area-efficient Reed-Solomon decoder design for optical communications," IEEE Trans. Circuits Syst. II, Express Briefs, vol.56, no.6, pp.469–473, June 2009.
[20] I.S. Reed, M.T. Shih, and T.K. Truong, "VLSI design of inverse-free Berlekamp–Massey algorithm," Proc. Inst. Elect. Eng., pt. E, vol.138, pp.295–298, Sept. 1991.
[21] R.E. Blahut, Theory and Practice of Error-Control Codes, Reading, Addison-Wesley, MA, 1983.
[22] E.R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York, 1968. (revised ed.—Laguna Hills, Aegean Park, CA, 1984).

Yung-Kuei Lu received the B.S. and M.S. degree in electrical engineering from National Cheng Kung University, Taiwan, in 2000 and 2005 respectively. He is pursuing his Ph.D. degree in National Cheng Kung University, Taiwan, since 2005. His research interests include VLSI implementation in digital signal processing architectures and error control coding.

Ming-Der Shieh received the B.S. degree in electrical engineering from National Cheng Kung University, in 1984, the M.S. degree in electronic engineering from National Chiao Tung University, Taiwan, in 1986, and the Ph.D. degree in electrical engineering from Michigan State University, East Lansing, in 1993. From 1988 to 1989, he was an engineer at United Microelectronic Corporation, Taiwan. From 1993 to 2002, he was with the faculty of Department of Electronic Engineering, National Yunlin University of Science & Technology. He received the teaching award from NYUST in 1998 and was the department chairman from 1999 to 2002. Since 2002, he has been with the Department of Electrical Engineering, National Cheng Kung University, where he is currently a professor. His research interests include VLSI design and testing, VLSI for signal processing, digital communication, and computer-aided design. He was a general co-chair and program co-chair of Asian Test Symposium in 2004 and 2009, respectively, and is the chair of Tainan Chapter of IEEE Circuits and Systems from 2009 to 2010.