On the High-Speed VLSI Implementation of Errors-and-Erasures Correcting Reed-Solomon Decoders

Tong Zhang and Keshab K. Parhi
Department of Electrical and Computer Engineering, University of Minnesota
{tzhang,parhi}@ece.umn.edu

ABSTRACT

Recently a novel algorithm transformation was proposed to reduce the critical path of Berlekamp-Massey algorithm implementation for errors-alone Reed-Solomon decoding. In this paper, we apply the same methodology to transform the Berlekamp-Massey algorithm for errors-and-erasures RS decoding. We present a regular hardware architecture to implement the reformulated Berlekamp-Massey algorithm, which can achieve high throughput. Moreover, an operation scheduling scheme is proposed to further reduce the hardware complexity without loss of throughput.

Categories and Subject Descriptors

B.2.4 [**ARITHMETIC AND LOGIC STRUCTURES**]: High-Speed Arithmetic

General Terms

Design

Keywords

Reed-Solomon codes, Berlekamp-Massey algorithm, erasure, VLSI architectures

1. INTRODUCTION

Reed-Solomon (RS) codes are widely used for forward error correcting (FEC) in numerous communication systems because of their good error correction capability for burst errors. A conventional errors-alone RS decoding procedure can be modified to correct both errors and erasures [1][4] (an erasure occurs when the position of a corrupted symbol is known). RS decoder with the capability of correcting errors as well as erasures will improve performance in various systems [9][3].

To achieve higher decoding speed, except applying the more advanced physical-level, e.g., sub-micron, technology to improve the throughput, the algorithm/architecture-level transformations and modifications are also playing important roles. In RS decoders, the throughput bottleneck is in solving the Berlekamp's key equation, where two algorithms are typically employed [1]: extended Euclidean (eE) algorithm and Berlekamp-Massey (BM) algorithm. Recently a novel algorithm transformation was proposed in [7] to reformulate the BM algorithm for errors-alone RS decoders so that the key-equation-solving block has a very regular semi-systolic architecture with the critical path of only $T_{mult} + T_{add}$, where T_{mult} and T_{add} are the delays of the finite field multiplier and adder, respectively. This amount of critical path is much smaller than that of the previous RS decoder implementations using BM algorithm and comparable to the best known result of implementations based on eE algorithm. The basic methodology behind such algorithm transformation is to remove the data dependency inside of one iteration at the cost of increased computation complexity so that the critical path can be reduced. In this paper, we show that this methodology can be readily applied to the errors-and-erasures RS decoder implementations to achieve regular semi-systolic architecture and critical path of $T_{mult} + T_{add}$. Moreover, exploiting the characteristics of errors-and-erasures RS decoding, we propose an operation scheduling scheme for the developed decoder architecture to reduce the hardware complexity without loss of throughput.

This paper is organized as follows. We introduce the fundamentals of RS decoding in Section 2. An inverse-free BM algorithm for errors-and-erasures RS decoding is described in Section 3. In Section 4, we apply the methodology proposed in [7] to transform the BM algorithm for errors-and-erasures RS decoding, based on which we present a semi-systolic high-speed hardware architecture. We note that in order to facilitate the reader's reference to [7], we use the same notations as much as possible in the algorithm and hardware architecture descriptions.

2. RS CODES DECODING

Let's consider an (n, k) RS code over $GF(2^m)$, where each code symbol is a m-bit data, $n = 2^m - 1$ is the codeword block length and k is the information length. Since RS codes are maximum distance separable (MDS) codes [6], the (n, k) RS code has a minimum distance $d = n - k + 1$. When such RS code is used on a channel that makes both errors and erasures, any pattern of v errors and ρ erasures can be corrected provided that $d \geq 2v + \rho + 1$. Let $C(z)$ denote the transmitted codeword polynomial and $R(z)$ denote the received word polynomial in which erased symbols are represented as

GLSVLSI'02, April 18-19, 2002, New York, New York, USA.

blanks (or zeroes), we have

$$R(z) = C(z) + E(z) + F(z),$$

where $E(z)$ and $F(z)$ represent the error polynomial and erasure polynomial. Define the *errata polynomial* as

$$\begin{aligned}\tilde{E}(z) &= E(z) + F(z) \\ &= \tilde{e}_{i_1} z^{i_1} + \tilde{e}_{i_2} z^{i_2} + \cdots + \tilde{e}_{i_{v+\rho}} z^{i_{v+\rho}}.\end{aligned}$$

Let α be the primitive element in $GF(2^m)$, we say that *errata magnitudes* $\tilde{Y}_l = \tilde{e}_{i_l}$ for $l = 1, \ldots, v+\rho$, occurred at *errata locations* $\tilde{X}_l = \alpha^{i_l}$ for $l = 1, \ldots, v+\rho$. Suppose that the $d-1$ consecutive powers of α used in RS code construction are $\alpha^{m_0}, \alpha^{m_0+1}, \cdots, \alpha^{m_0+d-2}$. The RS decoder always begin by computing the syndromes

$$s_k = R(\alpha^{m_0+k}) = \tilde{E}(\alpha^{m_0+k}), \quad 0 \le k \le d-2.$$

We introduce the syndrome polynomial $S(z) = \sum_{k=0}^{d-2} s_k z^k$, and define the *errata locator* polynomial $\Lambda(z)$ and *errata evaluator* polynomial $\Omega(z)$ as

$$\begin{aligned}\Lambda(z) &= \prod_{j=1}^{v+\rho} (1 - \tilde{X}_j z) = 1 + \lambda_1 z + \cdots + \lambda_{v+\rho} z^{v+\rho}, \\ \Omega(z) &= \sum_{i=1}^{v+\rho} \tilde{Y}_i \tilde{X}_i^{m_0} \prod_{j=1, j\neq i}^{v+\rho} (1 - \tilde{X}_j z) \\ &= \omega_0 + \omega_1 z + \cdots + \omega_{v+\rho-1} z^{v+\rho-1}.\end{aligned}$$

We can determine the above two unknown polynomials $\Lambda(z)$ and $\Omega(z)$ by solving the *key equation* [1]:

$$\Omega(z) \equiv \Lambda(z) S(z) \bmod z^{d-1}. \qquad (1)$$

Notice that the $v+\rho$ roots of $\Lambda(z)$ consist of the inverses of v unknown error locations and ρ known erasure locations. Once $\Lambda(z)$ and $\Omega(z)$ are found, the decoder can use Chien search to identify the v unknown roots of $\Lambda(z)$ so that all the $v+\rho$ errata locations $\tilde{X}_i$ become known. Then all the $v+\rho$ errata magnitudes $\tilde{Y}_i$ can be determined by using the Forney algorithm [4]

$$\tilde{Y}_i = -\frac{\tilde{X}_i^{1-m_0} \Omega(\tilde{X}_i^{-1})}{\Lambda'(\tilde{X}_i^{-1})}, \text{ for } i = 1, 2, \cdots, v+\rho. \qquad (2)$$

Using all the $\tilde{X}_i's$ and $\tilde{Y}_i's$ we can derive the errata polynomial $\tilde{E}(z)$, and finally reconstruct the correct codeword $C(z) = R(z) - \tilde{E}(z)$.

The VLSI implementation of the above decoding process typically contains four consecutive pipelined stages : 1. the syndrome computation (SC) block; 2. the key equation solver (KES) block; 3. the Chien search (CS) block; 4. the errata evaluator (EE) block. All the blocks work in parallel on consecutive codewords. Pipeline is also typically used in each block to achieve higher throughput. Compared with the other three blocks which can be pipelined in a straightforward manner, the KES block, based on BM or eE algorithm, cannot be easily pipelined due to the presence of inherent *feedback loops*. Thus we may say that the throughput bottleneck in RS decoders is in the KES block.

3. BERLEKAMP-MASSEY ALGORITHM

Recently an efficient inverse-free BM algorithm was proposed in [5] for errors-and-erasures RS decoding. We adopt such algorithm as the starting point of this work. In the following, we briefly describe this algorithm based on [5]. Notice that such algorithm actually finds scalar multiples $\beta \cdot \Lambda(z)$ and $\beta \cdot \Omega(z)$, where β is a field element in $GF(2^m)$. Since these scaled results will not affect the computation of errata locations using Chien search and the errata magnitudes using (2), we still refer to the polynomials output by the following BM algorithm as $\Lambda(z)$ and $\Omega(z)$.

Suppose errata locations $\tilde{X}_{l_1}, \tilde{X}_{l_2}, \cdots, \tilde{X}_{l_\rho}$ correspond to the ρ erasures, we define the erasure locator polynomial as

$$\Psi(z) = \prod_{i=1}^{\rho} (1 - \tilde{X}_{l_i} z) = 1 + \psi_1 z + \cdots + \psi_\rho z^\rho. \qquad (3)$$

Notice that, provided with $2v + \rho + 1 \le d$, when $\rho = d-1$ or $\rho = d-2$, no errors occurred (or $v = 0$) and the errata locator polynomial $\Lambda(z) = \Psi(z)$ is immediately available, the decoder can bypass the following BM algorithm and computes errata evaluator polynomial $\Omega(z)$ directly using (1).

Algorithm 3.1. *Inverse-Free BM Algorithm for Errors-and-Erasures Decoding*

1. *Initialization:* $r = k^{(0)} = 0$, $\gamma^{(0)} = 1$, $\Lambda^{(0)}(z) = B^{(0)}(z) = \Psi(z)$.

2. *Set* $r = r+1$. *If* $r > d - \rho - 1$, *go to step 6, else compute the discrepancy*

$$\delta^{(r-1)} = \lambda_0^{(r-1)} \cdot s_{r+\rho-1} + \cdots + \lambda_{r+\rho-1}^{(r-1)} \cdot s_0. \qquad (4)$$

3. *Compute* $\Lambda^{(r)}(z) = \gamma^{(r-1)} \cdot \Lambda^{(r-1)}(z) - z \cdot \delta^{(r-1)} \cdot B^{(r-1)}(z)$ *so that for* $i = 0, 1, \cdots, d-3$,

$$\lambda_i^{(r)} = \gamma^{(r-1)} \cdot \lambda_i^{(r-1)} - \delta^{(r-1)} \cdot b_{i-1}^{(r-1)}. \qquad (5)$$

4. *If* $\delta^{(r-1)} \neq 0$ *and* $k^{(r-1)} \ge 0$, *then*

$$\left\{\begin{array}{lcl} B^{(r)}(z) &=& \Lambda^{(r-1)}(z), \\ \gamma^{(r)} &=& \delta^{(r-1)}, \\ k^{(r)} &=& -k^{(r-1)} - 1. \end{array}\right.$$

else

$$\left\{\begin{array}{lcl} B^{(r)}(z) &=& z \cdot B^{(r-1)}(z), \\ \gamma^{(r)} &=& \gamma^{(r-1)}, \\ k^{(r)} &=& k^{(r-1)} + 1. \end{array}\right.$$

5. *Return to step 2.*

6. *Set* $\Lambda(z) = \Lambda^{(d-\rho-1)}(z)$, *and compute* $\Omega(z) = (\Lambda(z) \cdot S(z))$ *mod* z^{d-1} *so that for* $i = 0, 1, \cdots, d-3$

$$\omega_i = \lambda_0^{(d-\rho-1)} \cdot s_i + \lambda_1^{(d-\rho-1)} \cdot s_{i-1} \cdots + \lambda_i^{(d-\rho-1)} \cdot s_0.$$

Notice that $b_{-1}^{(r)}$ occurring in the above algorithm are set to zeroes. This algorithm provides the solutions of the key equation: $\Lambda(z)$ and $\Omega(z)$, where all the λ_i for $i > v+\rho$ and ω_j at least for $j > v+\rho-1$ are zeroes.

Notice that the loop between step 2 and 5 is completed in $d - \rho - 1$ iterations, and in each iteration there is a data dependency between the computations of (4) and (5) through the discrepancy $\delta^{(r-1)}$. Thus, if we want to realize the above algorithm in such a way that one iteration is completed in one clock cycle for the minimal latency, the critical path will be larger than $2 \cdot (T_{mult} + T_{add})$ that is typically much longer than the critical paths of all other building blocks in RS decoder. Therefore, reducing this critical path becomes the key to achieve higher RS decoding throughput.

4. HIGH-SPEED RS DECODER ARCHITECTURE

As mentioned earlier, the authors of [7] proposed a novel algorithm transformation to reformulate the errors-alone correcting BM algorithm to significantly reduce the critical path, where the basic idea is to remove the data dependency inside of one iteration at the cost of increased computation complexity so that the critical path can be reduced. In this section, we apply the same methodology to reformulate the above BM algorithm for errors-and-erasures RS decoding. The developed KES block has a regular semi-systolic architecture with the critical path of only $T_{mult} + T_{add}$. Moreover, to reduce the overall RS decoder hardware complexity without loss of latency, we propose an operation scheduling scheme to make KES block not only solve the key equation but also compute an intermediate polynomial that is a primary input to the reformulated BM algorithm.

4.1 Reformulated BM Algorithm

In the following we present how to apply the essential methodology behind the algorithm transformation proposed in [7] for errors-alone RS decoding to our interested errors-and-erasures RS decoding. The readers are highly recommended to read [7] for its detailed descriptions.

First notice that the discrepancy $\delta^{(r-1)}$ in Algorithm 3.1, the pivotal element along the data dependency path, is actually $\delta^{(r-1)}_{r+\rho-1}$, the coefficient of $z^{r+\rho-1}$ in the polynomial

$$\begin{aligned}\Delta^{(r-1)}(z) &= \Lambda^{(r-1)}(z) \cdot S(z)\\ &= \delta_0^{(r-1)} + \delta_1^{(r-1)} z + \cdots + \delta^{(r-1)}_{r+\rho-1} z^{r+\rho-1} + \cdots .\end{aligned}$$

If we introduce a new polynomial $\Theta^{(r)}(z) = B^{(r)}(z) \cdot S(z)$, according to Algorithm 3.1, we have

$$\begin{aligned}\Delta^{(r)}(z) &= \Lambda^{(r)}(z) \cdot S(z)\\ &= (\gamma^{(r-1)} \cdot \Lambda^{(r-1)}(z) - z \cdot \delta^{(r-1)} \cdot B^{(r-1)}(z)) \cdot S(z)\\ &= \gamma^{(r-1)} \cdot \Delta^{(r-1)}(z) - z \cdot \delta^{(r-1)} \cdot \Theta^{(r-1)}(z), \qquad (6)\end{aligned}$$

where $\Theta^{(r)}(z) = B^{(r)} \cdot S(z)$ will be either $\Delta^{(r-1)}(z) = \Lambda^{(r-1)}(z) \cdot S(z)$ or $z \cdot \Theta^{(r-1)}(z) = z \cdot B^{(r-1)} \cdot S(z)$. We can easily prove that, if $\Delta^{(0)}(z)$ is available at the beginning and we replace the computation of (4) in Algorithm 3.1 with (6), discrepancy $\delta^{(r-1)} = \delta^{(r-1)}_{r+\rho-1}$, the crucial element in each iteration, will be computed in the previous clock cycle instead of the current clock cycle (or we can say that discrepancy $\delta^{(r-1)}$ is computed in a *look-ahead* style). In this way, the original data dependency in each iteration (or *intra-iteration* data dependency) has been transformed into successive iterations (or replaced by an *inter-iteration* data dependency). Thus, with the disappearance of intra-iteration data dependency, all the computations in one iteration can be performed in parallel and, as shown later, the corresponding KES block can achieve the critical path of only $T_{mult} + T_{add}$.

Let $z^i \bmod z^{d-1} = 0$ for $i < 0$, we introduce the polynomial $\Phi(z)$:

$$\Phi(z) = (z^{-\rho} \cdot \Psi(z) \cdot S(z)) \bmod z^{d-1} = \sum_{i=0}^{d-2} \phi_i z^i, \qquad (7)$$

where

$$\phi_i = \psi_0 \cdot s_{i+\rho} + \psi_1 \cdot s_{i+\rho-1} + \cdots + \psi_\rho \cdot s_i .$$

Moreover, notice that for any $i < r + \rho$, $\delta_i^{(r)}$ and $\theta_i^{(r)}$ cannot affect the value of any later discrepancy $\delta^{(r+j)}_{r+\rho+j}$. Consequently, we need not store $\delta_i^{(r)}$ and $\theta_i^{(r)}$ for $i < r + \rho$. Thus we define $\hat{\delta}_i^{(r)} = \delta^{(r)}_{i+r+\rho}$, $\hat{\theta}_i^{(r)} = \theta^{(r)}_{i+r+\rho}$ and the polynomials

$$\hat{\Delta}^{(r)}(z) = \sum_{i=0}^{d-2} \hat{\delta}_i^{(r)} z^i \quad \text{and} \quad \hat{\Theta}^{(r)}(z) = \sum_{i=0}^{d-2} \hat{\theta}_i^{(r)} z^i \qquad (8)$$

with initial values $\hat{\Delta}^{(0)}(z) = \hat{\Theta}^{(0)}(z) = \Phi(z)$. Based on the above discussion and introduced polynomials, we can readily transform Algorithm 3.1 to the following algorithm suitable for high-speed implementations:

ALGORITHM 4.1. *Reformulated Inverse-Free BM Algorithm for Errors-and-Erasures Decoding*

1. *Initialization:* $r = k^{(0)} = 0$, $\gamma^{(0)} = 1$, $\Lambda^{(0)}(z) = B^{(0)}(z) = \Psi(z)$ *and* $\hat{\Delta}^{(0)}(z) = \hat{\Theta}_i^{(0)}(z) = \Phi(z)$.
2. *Set* $r = r + 1$. *If* $r > d - \rho - 1$, *algorithm terminates, else compute* $\hat{\Delta}^{(r)}(z) = z \cdot \gamma^{(r-1)} \cdot \hat{\Delta}^{(r-1)}(z) - \hat{\delta}_0^{(r-1)} \cdot \hat{\Theta}^{(r-1)}(z)$ *so that for* $i = 0, 1, \cdots, d-2$,
$$\hat{\delta}_i^{(r)} = \gamma^{(r-1)} \cdot \hat{\delta}_{i+1} - \hat{\delta}_0^{(r-1)} \cdot \hat{\theta}_i^{(r-1)}. \qquad (9)$$
3. *Compute* $\Lambda^{(r)}(z) = \gamma^{(r-1)} \cdot \Lambda^{(r-1)}(z) - z \cdot \hat{\delta}_0^{(r-1)} \cdot B^{(r-1)}(z)$ *so that for* $i = 0, 1, \cdots, d-3$,
$$\lambda_i^{(r)} = \gamma^{(r-1)} \cdot \lambda_i^{(r-1)} - \hat{\delta}_0^{(r-1)} \cdot b_{i-1}^{(r-1)}. \qquad (10)$$
4. *If* $\hat{\delta}_0^{(r-1)} \neq 0$ *and* $k^{(r-1)} \geq 0$, *then*
$$\begin{cases} B^{(r)}(z) &= \Lambda^{(r-1)}(z),\\ \hat{\Theta}^{(r)}(z) &= z^{-1} \cdot \hat{\Delta}^{(r-1)}(z),\\ \gamma^{(r)} &= \hat{\delta}_0^{(r-1)},\\ k^{(r)} &= -k^{(r-1)} - 1. \end{cases}$$
else
$$\begin{cases} B^{(r)}(z) &= z \cdot B^{(r-1)}(z),\\ \hat{\Theta}^{(r)}(z) &= \hat{\Theta}^{(r-1)}(z),\\ \gamma^{(r)} &= \gamma^{(r-1)},\\ k^{(r)} &= k^{(r-1)} + 1. \end{cases}$$
5. *Return to step 2.*

It is clear that, in the above algorithm, all the computations in one iteration, (9) and (10), can be carried out in parallel, from which we may easily develop its hardware realization architecture, as shown later, with a critical path of only $T_{mult} + T_{add}$. Notice that $\Lambda^{(d-\rho-1)}(z)$ is exactly the solution of the key equation for $\Lambda(z)$. Although $\hat{\Omega}(z) = \hat{\Delta}^{(d-\rho-1)}(z)$ is *not* the solution of the key equation for $\Omega(z)$, following the argument in [7], we have

$$\Omega(\tilde{X}_i^{-1}) = -\tilde{X}_i^{-(d-1)} \hat{\Omega}(\tilde{X}_i^{-1}), \quad \text{for} \quad i = 1, 2, \cdots, v + \rho.$$

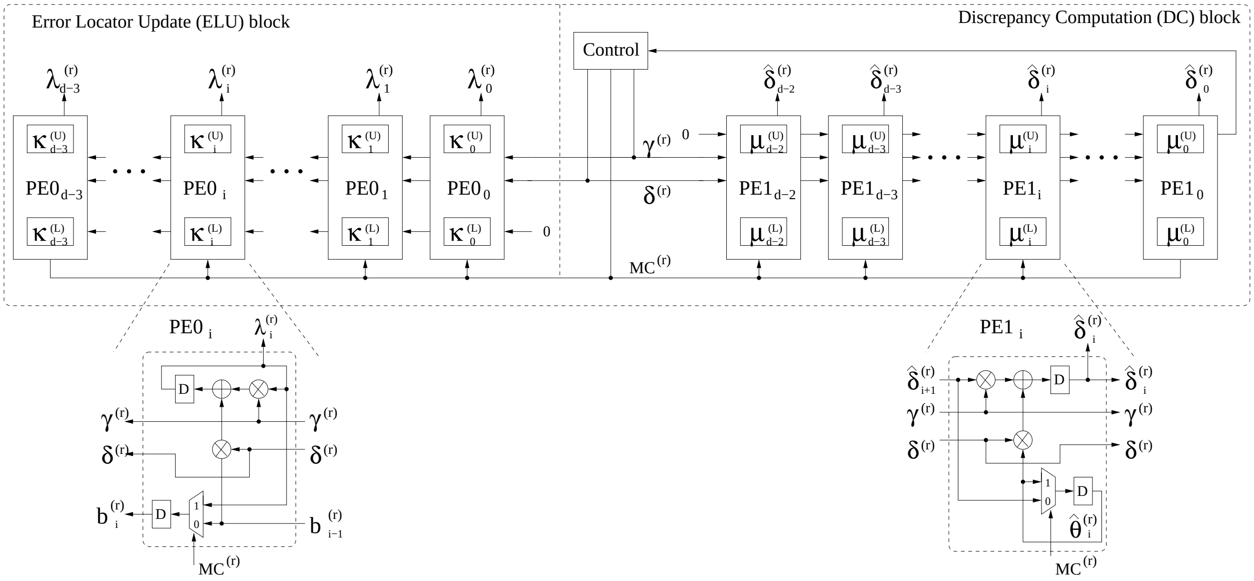

Figure 1: Key equation solver (KES) block architecture.

Thus, using the above reformulated algorithm, we can find all the errata locations $\tilde{X}_i$ by applying Chien search on $\Lambda(z)$ and derive all the errata magnitudes by rewriting (2) as

$$\tilde{Y}_i = -\frac{\tilde{X}_i^{2-d-m_0}\hat{\Omega}(\tilde{X}_i^{-1})}{\Lambda'(\tilde{X}_i^{-1})}, \text{ for } i = 1, 2, \cdots, v+\rho.$$

In this way, we still say that the Algorithm 4.1 solves the RS decoding key equation and the corresponding hardware block is referred as key equation solver (KES).

4.2 Decoder Architecture

Following the above discussion, we know that such errors-and-erasures RS decoder principally contains six blocks: 1. the syndrome computation (SC) block; 2. the erasure locator (EL) block to compute erasure locator polynomial $\Psi(z)$; 3. the Φ-block to compute polynomial $\Phi(z)$; 4. the key equation solver (KES) block; 5. the Chien search (CS) block; 6. errata evaluator (EE) block. Since all other blocks have been well-investigated in conventional errors-alone RS decoder implementations, in what follows, we only consider the architecture design for the EL, Φ-block and KES blocks. For the implementations of other blocks, readers are referred to [2][8], etc.

The architecture design for EL block can be performed in a quite straightforward manner. Let $\alpha^{t_1}, \alpha^{t_2}, \cdots, \alpha^{t_\rho}$ be the ρ erasure locations where $t_1 < t_2 < \cdots < t_\rho$. Suppose, before the erasure at α^{t_s} is detected, we have obtained the *intermediate* erasure locator polynomial

$$\begin{aligned}\Psi^{(s-1)}(z) &= \prod_{i=1}^{s-1}(1-\alpha^{t_i}z)\\ &= 1+\psi_1^{(s-1)}z+\cdots+\psi_{s-1}^{(s-1)}z^{s-1}.\end{aligned}$$

Then, after detecting the erasure at α^{t_s}, we update the intermediate erasure locator polynomial as

$$\begin{aligned}\Psi^{(s)}(z) &= \Psi^{(s-1)}(z)\cdot(1-\alpha^{t_s}z)\\ &= 1+\psi_1^{(s)}z+\cdots+\psi_s^{(s)}z^s,\end{aligned}$$

where $\psi_i^{(s)} = \psi_i^{(s-1)} - \alpha^{t_s}\cdot\psi_{i-1}^{(s-1)}$. Based on such iterative update rule, we can easily develop one possible hardware architecture as shown in Fig. 2 for the EL block if the received codeword is fed to the decoder one code symbol per clock cycle, which is a typical configuration in conventional RS decoder design. Notice that the critical path here is $T_{mult} + T_{add}$.

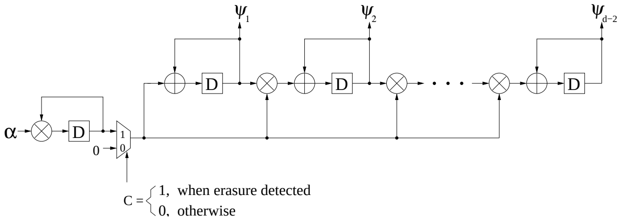

Figure 2: . Erasure locator (EL) block architecture.

Next let's consider the KES architecture. Directly exploring the above reformulated BM algorithm and referring to the semi-systolic KES architecture presented in [7], we may easily obtain the semi-systolic KES architecture, as shown in Fig. 1, for the implementations of errors-and-erasures RS decoder. This KES architecture mainly consists of two sub-blocks: discrepancy computation (DC) block and error locator update (ELU) block, which perform the computations of (9) and (10) in each iteration, respectively. Compared with the KES for errors-alone RS decoder [7], this architecture contains $\lfloor\frac{d-1}{2}\rfloor - 1$ more $PE0$ blocks due to the higher possible degree of errata locator polynomial $\Lambda(z)$.

Notice that, as shown in Fig. 1, $\kappa_i^{(U)}$ and $\kappa_i^{(L)}$ in each $PE0_i$ and $\mu_i^{(U)}$ and $\mu_i^{(L)}$ in each $PE1_i$ represent the initialization value of the corresponding upper and lower DFF (D flip-flop) in each $PE0_i$ and $PE1_i$ blocks. According to Algorithm 4.1, if we initialize all the DFF's with

$$\begin{cases}\kappa_i^{(U)} = \kappa_i^{(L)} = \psi_i, & \text{for } i = 0, 1, \cdots, d-3,\\ \mu_i^{(U)} = \mu_i^{(L)} = \phi_i, & \text{for } i = 0, 1, \cdots, d-2,\end{cases}$$

such KES block will produce $\Lambda(z)$ and $\hat{\Omega}(z)$ in $d-\rho-1$ clock cycles.

Finally let's consider the implementation of Φ-block for the computation of polynomial $\Phi(z)$. From (7) we know that $\Phi(z)$ is obtained through a polynomial multiplication followed by modular z^{d-1}. It's well-known that the polynomial multiplication can be realized using a FIR filter with the coefficients initialized by one of the two multiplying polynomials [1], thus we can implement the Φ-block using a $d-1$ order FIR as shown in Fig. 3: each coefficient is set to s_i, the coefficient of $S(z)$, and each coefficient ψ_i of $\Psi(z)$ is sequentially fed to this FIR as primary input.

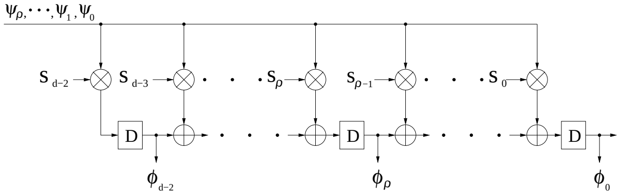

Figure 3: FIR Architecture for Φ-block.

Each DFF in the such FIR filter is reset to zeroes at the beginning, then each coefficient ψ_i is input sequentially with the subscript i increasing from 0 to ρ. We can easily prove that after the last element ψ_ρ is fed to FIR at $(\rho+1)$-th clock cycle, as shown in Fig. 3, the output of all the DFF's in this FIR are exactly $\phi_{d-2}, \phi_{d-3}, \cdots, \phi_0$, the coefficients of desired polynomial $\Phi(z)$.

From the above discussion, we know that the Φ-block and the KES block complete computation for one codeword frame in $\rho+1$ and $d-\rho-1$ clock cycles, respectively. Since the erasure number ρ may varies from 0 to $d-1$, it's desirable to implement the Φ-block and the KES block in the same pipeline stage so that they work on the same codeword frame and produce polynomial $\Lambda(z)$ and $\hat{\Omega}(z)$ in the fixed d clock cycles. However, in each d clock cycles, the Φ-block and the KES block will be idle in $d-\rho-1$ and $\rho+1$ clock cycles, respectively, which is a somehow waste of silicon area. In the following, we propose a simple scheduling scheme for the KES block in such a way that it can compute $\Phi(z)$ and solve the key equation in total d clock cycles. Therefore, we don't need to physically implement the above $d-1$ order FIR for the Φ-block.

Notice that, as shown in Fig. 1, if we set $\gamma^{(r)} = 1$ and $MC = 0$ and consecutively assign $\delta^{(r)}$ as ψ_i with the subscript i increasing from 0 to ρ, the DC block will perform in the exactly same way as the $d-1$ order FIR for the Φ-block as in Fig. 3. Based on the above observation, we propose the following scheduling scheme for the KES block to complete both the computation of $\Phi(z)$ and solve the key equation in total d clock cycles:

PROCEDURE 4.1. *KES Block Operation Scheduling*

1. *In the first $\rho+1$ clock cycles, compute $\Phi(z)$ using the DC block only:*
   - *(a) Initialize all the DFF's in DC block with $\mu_i^{(U)} = 0$ and $\mu_i^{(L)} = s_i$, for $i = 0, 1, \cdots, d-2$, set $\gamma^{(r)} = 1$ and $MC = 0$.*
   - *(b) For $0 \le r \le \rho$, in each clock cycle, set $\delta^{(r)} = \psi_r$.*

   After $\rho+1$ clock cycles, the content of the upper DFF in each $PE1_i$ block, $\hat{\delta}_i^{(\rho+1)}$, is equal to ϕ_i.
2. *In the following $d-\rho-1$ clock cycles, compute the $\Lambda(z)$ using ELU block and $\hat{\Omega}(z)$ using DC block:*
   - *(a) Re-initialize all the DFF's in DC and ELU blocks with $\mu_i^{(L)} = \mu_i^{(U)} = \phi_i$ and $\kappa_i^{(L)} = \kappa_i^{(L)} = \psi_i$.*
   - *(b) For $0 \le r \le d-\rho-1$, in each clock cycle, set $\delta^{(r)} = \hat{\delta}_0^{(r)}$ and generate $\gamma^{(r)}$ and $MC^{(r)}$ according to Algorithm 4.1.*

   Finally, we obtain the polynomial $\Lambda(z)$ and $\hat{\Omega}(z)$ as illustrated in Fig. 1.

5. CONCLUSION

Applying the methodology behind the algorithm transformation proposed in [7], in this paper, we reformulated the BM algorithm for errors-and-erasures RS decoding and correspondingly present the semi-systolic architecture which has a critical path of only $T_{mult} + Tadd$. Moreover, an operation scheduling scheme has been proposed for the key equation solver (KES) block so that the overall RS decoder hardware complexity can be reduced without loss of throughput. Compared with the errors-alone RS decoder, the presented errors-and-erasures RS decoder may achieve the same throughput but requires extra $d + 2 \cdot \lfloor \frac{d-1}{2} \rfloor - 3$ multipliers and $d + \lfloor \frac{d-1}{2} \rfloor - 3$ adders due to the implementation of erasure locator (EL) block and higher possible degree of errata locator polynomial $\Lambda(z)$ in the KES block.

6. REFERENCES

[1] R. E. Blahut. *Theory and Practice of Error Control Codes*. Addison Wesley, 1984.

[2] H.-C. Chang, C. B. Shung, and C.-Y. Lee. A Reed-Solomon product-code (RS-PC) decoder chip for DVD applications. *IEEE Journal of Solid-State Circuits*, 36(2):229–238, Feb. 2001.

[3] C. B. *et al.* The U.S. HDTV standard the grand. *IEEE Spectrum*, 32:36–45, April 1995.

[4] J. Forney. On decoding BCH codes. *IEEE Trans. on Inform. Theory*, IT-11:549–557, Oct. 1965.

[5] 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. on Communications*, 47(10):1488–1494, Oct. 1999.

[6] V. Pless. *Introduction to the theory of error-correcting codes*. Wiley, 1998.

[7] D. V. Sarwate and N. R. Shanbhag. High-speed architecture for Reed-Solomon decoders. *IEEE Trans. on VLSI Systems*, 9(5):641–655, Oct. 2001, available at http://www.icims.csl.uiuc.edu/~shanbhag/myhome.

[8] K. Seki, K. Mikami, M. Baba, N. Shinohara, S. Suzuki, and H. Tezuka. Single-ship 10.7Gb/s FEC CODEC LSI using time-multiplexed RS decoder. In *IEEE Custom Integrated Circuits Conference*, pages 289–292, 2001.

[9] L.-L. Yang and L. Hanzo. Performance analysis of coded M-ary orthogonal signaling using errors-and-erasures decoding over frequency-selective fading channels. *IEEE Journal on Selected Areas in Communications*, 19(2):211 –221, Feb. 2001.