Parallel Welch–Berlekamp Algorithm

Chao Chen, Yunghsiang S. Han, *Fellow, IEEE*, Nianqi Tang, Xiao Ma, and Baoming Bai

***Abstract*—This paper presents new variants of the Welch–Berlekamp algorithm that are favorable to hardware implementation. First, we derive the parallel Welch–Berlekamp (PWB) algorithm in a constructive manner based on the properties of solutions to the rational interpolation problem. The algorithm features the simultaneously performed discrepancy computation and polynomial update. Second, we explore the early-termination mechanism of the PWB algorithm for decoding of Reed–Solomon (RS) codes. By introducing the concept of incomplete error locator polynomial, we show that if $e \leq t$ (where e is the number of errors and t is the error correction capability), the PWB algorithm can be terminated at latest at the completion of the $(t+e)$-th iteration. This leads to the early-terminating PWB (EPWB) algorithm. Finally, we develop frequency-domain versions of the PWB and EPWB algorithms, namely, FPWB and FEPWB. The key point toward the two algorithms is to replace the update of polynomial coefficients with the update of polynomial evaluations. It is worth noting that the FEPWB algorithm applies only to shortened RS codes. Furthermore, an efficient systolic architecture for the FPWB algorithm is designed, which is easily adapted for the FEPWB algorithm.**

***Index Terms*—Reed–Solomon codes, parallel Welch–Berlekamp algorithm, incomplete error locator polynomial, early termination, systolic architecture.**

I. Introduction

RECENTLY, a new Fast Fourier Transform (FFT) over binary extension fields was proposed by Lin, Chung, and Han [5], which for the first time achieves the $O(n \log n)$ complexity over such fields (where n is the FFT length). Based on LCH-FFT, new low-complexity encoding and decoding algorithms have been developed for Reed–Solomon (RS) codes [5], [6], [7], [8], [9], [10].

The LCH-FFT-based RS decoding has the Welch–Berlekamp (WB) form of the key equation [7], which can be solved by the WB algorithm [2], [3], [4]. Due to the use of LCH-FFT, the computational complexity of both the syndrome computation and the Chien search is significantly reduced. Although the WB algorithm requires a little more computation, the overall computation complexity of the LCH-FFT-based RS decoding is considerably lower than that of the conventional RS decoding that uses the Berlekamp–Massey (BM) algorithm [1]. See [9] for a detailed complexity comparison.

This work was supported by the China National Key R&D Program under Grant 2021YFA1000500. Part of this work was presented at the 2023 IEEE International Symposium on Information Theory [20]. (*Corresponding author: Chao Chen.*)

Chao Chen and Baoming Bai are with the State Key Lab of ISN, Xidian University, Xi'an, China. (e-mail: cchen@xidian.edu.cn, bmbai@mail.xidian.edu.cn).

Yunghsiang S. Han and Nianqi Tang are with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen, China. (e-mail: yunghsiangh@gmail.com, 724973040@qq.com).

Xiao Ma is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. (e-mail: maxiao@mail.sysu.edu.cn).

For the purpose of hardware implementation, parallel algorithms other than the WB algorithm were derived to solve the WB equation based on the concepts of module and exact sequence [11]. Recently, the modular approach (MA) algorithm [9] was proposed as an improvement of [11]. Two variants, called frequency-domain modular approach (FDMA) and fast modular approach (FMA), were also developed [9], which are suitable for hardware and software implementations, respectively. In [19], it was shown that the MA algorithm is equivalent to a modified WB algorithm.

In this paper, we present new variants of the WB algorithm favorable to hardware implementation. The main contributions are as follows.

1) We present the parallel Welch–Berlekamp (PWB) algorithm. Different from the WB algorithm that conducts the discrepancy computation and the polynomial update in serial, the PWB algorithm conducts the two operations in parallel, hence its name. The derivation of the PWB algorithm is constructive from the properties of solutions to the rational interpolation problem, which is entirely different from that of the MA algorithm [9] that relies on complicated mathematical concepts including module and exact sequence.
2) We present an early-terminating PWB (EPWB) algorithm. It is known that the BM algorithm can be terminated early [13], [14], [15], [16], [17]. By introducing the concept of incomplete error locator polynomial, we show that the PWB algorithm can be terminated early as well. Specifically, if $e \leq t$ (where e is the number of errors and t is the error correction capability), the PWB algorithm can be terminated at latest at the completion of the $(t+e)$-th iteration. The early termination rule also applies to the WB algorithm. To the best of our knowledge, this is the first time that the early termination mechanism is explored for the WB algorithm.
3) We develop frequency-domain versions of the PWB and EPWB algorithms, dubbed FPWB and FEPWB, respectively. By the qualifier ‘frequency-domain’, we mean to use the update of polynomial evaluations to replace the update of polynomial coefficients. Moreover, we design an efficient architecture for the FPWB algorithm, which can be easily adapted for the FEPWB algorithm. The architecture has a regular systolic structure, a low implementation cost, and a small critical path.

The rest of the paper is organized as follows. Section II reviews the rational interpolation problem and characterizes the solution properties. Section III derives the PWB algorithm. Section IV presents the EPWB algorithm. Section V develops the FPWB and FEPWB algorithms and the architectures. Finally, Section VI concludes the paper.

II. The Rational Interpolation Problem

Let $\mathbb{F}$ be an arbitrary field. Let $(x_i, y_i) \in \mathbb{F} \times \mathbb{F}, 0 \le i \le \rho - 1$, be ρ points such that the x_i's are distinct. Consider the following rational interpolation problem.

Problem 1*:* Find a pair of polynomials $(W(x), N(x)) \in \mathbb{F}[x] \times \mathbb{F}[x]$ satisfying

$$N(x_i) = y_i W(x_i), \quad i = 0, 1, \dots, \rho - 1. \tag{1}$$

An equivalent formulation of Problem 1 is to find a pair of polynomials $(W(x), N(x))$ satisfying

$$N(x) = P(x)W(x) \mod \prod_{i=0}^{\rho-1}(x - x_i), \tag{2}$$

where $P(x)$ is an interpolating polynomial such that $P(x_i) = y_i$, $0 \le i \le \rho - 1$. For example, we may let $P(x)$ be the Lagrange interpolating polynomial

$$P(x) = \sum_{i=0}^{\rho-1} y_i \frac{\prod_{j \ne i}(x - x_j)}{\prod_{j \ne i}(x_i - x_j)}. \tag{3}$$

Clearly, Problem 1 has a trivial solution, $W(x) = N(x) = 0$. Here, we are only interested in the nontrivial solutions. Suppose that $(W(x), N(x))$ is a solution to Problem 1. The solution is said to be *reducible* if $W(x) = f(x)w(x)$ and $N(x) = f(x)n(x)$ such that $\deg(f(x)) > 0$ and $(w(x), n(x))$ is also a solution to Problem 1. The solution is said to be *irreducible* if it is not reducible.

Define the *rank* of a solution $(W(x), N(x))$ as

$$\operatorname{rank}(W(x), N(x)) \triangleq \max\{2\deg(W(x)), 2\deg(N(x)) + 1\}. \tag{4}$$

Suppose that $(W_0(x), N_0(x))$ and $(W_1(x), N_1(x))$ are two solutions of Problem 1. The two solutions are said to be *complementary* if

$$\operatorname{rank}(W_0(x), N_0(x)) + \operatorname{rank}(W_1(x), N_1(x)) = 2\rho + 1, \tag{5}$$

and

$$N_0(x)W_1(x) - N_1(x)W_0(x) = \gamma \prod_{i=0}^{\rho-1}(x - x_i), \tag{6}$$

for some $\gamma \ne 0 \in \mathbb{F}$. A solution is said to be a *complement* of another solution if the two solutions are complementary.

Lemma 1 ([18]): The solutions of Problem 1 have the following properties.

(i). There exists at least one irreducible solution with rank $\le \rho$.

(ii). If $(W_0(x), N_0(x))$ is an irreducible solution and $(W_1(x), N_1(x))$ is another solution such that

$$\operatorname{rank}(W_0(x), N_0(x)) + \operatorname{rank}(W_1(x), N_1(x)) \le 2\rho, \tag{7}$$

then $(W_1(x), N_1(x)) = (f(x)W_0(x), f(x)N_0(x))$ for some $f(x)$ with $\deg(f(x)) \ge 0$. This implies that the irreducible solutions with rank $\le \rho$ are unique (up to a scalar).

(iii). If $(W_0(x), N_0(x))$ and $(W_1(x), N_1(x))$ are two complementary solutions, then both of them are irreducible solutions and one of them is the (unique) irreducible solution with rank $\le \rho$.

(iv). If $(W_0(x), N_0(x))$ is the (unique) irreducible solution with rank $\le \rho$, then there exists at least one solution $(W_1(x), N_1(x))$ which is a complement of $(W_0(x), N_0(x))$.

(v). If $(W_0(x), N_0(x))$ is an irreducible solution and $(W_1(x), N_1(x))$ is one of its complements, then for any $a, b \in \mathbb{F}$ with $b \ne 0$, $(bW_1(x) - aW_0(x), bN_1(x) - aN_0(x))$ is also one of its complements.

It is worth pointing out that the properties as given in Lemma 1 were derived independently of any algorithm that solves Problem 1 [18]. Instead, the algorithm to be presented in the next section is constructed based on these properties.

The *minimal interpolation problem* related to Problem 1 can be described as follows [18].

Problem 2*:* Find a pair of polynomials $(W(x), N(x))$ satisfying

$$\begin{cases} N(x_i) = y_i W(x_i), \quad i = 0, 1, \dots, \rho - 1. \\ \operatorname{rank}(W(x), N(x)) \text{ is minimized.} \end{cases} \tag{8}$$

By Lemma 1, the solution of Problem 2 is unique, which is exactly the irreducible solution of Problem 1 with rank $\le \rho$. In the next section, we will derive a parallel WB algorithm to solve Problem 2.

III. Parallel Welch–Berlekamp Algorithm

A. Modified WB algorithm

Let $1 \le r \le \rho$, consider the following two problems.

Problem $1(r)$*:* Find a pair of polynomials $(W(x), N(x))$ satisfying

$$N(x_i) = y_i W(x_i), \quad i = 0, 1, \dots, r - 1. \tag{9}$$

Problem $2(r)$*:* Find a pair of polynomials $(W(x), N(x))$ satisfying

$$\begin{cases} N(x_i) = y_i W(x_i), \quad i = 0, 1, \dots, r - 1. \\ \operatorname{rank}(W(x), N(x)) \text{ is minimized.} \end{cases} \tag{10}$$

The basic idea for problem-solving is to construct two complementary solutions of Problem $1(r+1)$ from those of Problem $1(r)$. Suppose that $(W_0^{[r]}(x), N_0^{[r]}(x))$ and $(W_1^{[r]}(x), N_1^{[r]}(x))$ are two complementary solutions of Problem $1(r)$. By Lemma 1, one of the two solutions with a lower rank is the solution of Problem $2(r)$. By the definition of complementary solutions (refer to (5) and (6)), we have

$$\operatorname{rank}(W_0^{[r]}(x), N_0^{[r]}(x)) + \operatorname{rank}(W_1^{[r]}(x), N_1^{[r]}(x)) = 2r + 1, \tag{11}$$

and

$$N_0^{[r]}(x)W_1^{[r]}(x) - N_1^{[r]}(x)W_0^{[r]}(x) = \gamma \prod_{i=0}^{r-1}(x - x_i), \tag{12}$$

for some $\gamma \ne 0 \in \mathbb{F}$. To simplify notation, we denote $\operatorname{rank}(W_0^{[r]}(x), N_0^{[r]}(x))$ and $\operatorname{rank}(W_1^{[r]}(x), N_1^{[r]}(x))$ by $\operatorname{rank}_0^{[r]}$ and $\operatorname{rank}_1^{[r]}$, respectively. From (11), we see that one of $\operatorname{rank}_0^{[r]}$ and $\operatorname{rank}_1^{[r]}$ is even and the other is odd. Hence, $\operatorname{rank}_0^{[r]} \ne \operatorname{rank}_1^{[r]}$.

Define the *discrepancies* b_r and a_r as

$$\begin{pmatrix} b_r \\ a_r \end{pmatrix} \triangleq \begin{pmatrix} N_0^{[r]}(x_r) - y_r W_0^{[r]}(x_r) \\ N_1^{[r]}(x_r) - y_r W_1^{[r]}(x_r) \end{pmatrix}. \tag{13}$$

Then, we have the following result.

Lemma 2: a_r and b_r cannot both be zero.

Proof: Since $(x - x_r) \not| \prod_{i=0}^{r-1}(x - x_i)$, from (12), we have

$$(x - x_r) \not| (N_0^{[r]}(x)W_1^{[r]}(x) - N_1^{[r]}(x)W_0^{[r]}(x)). \tag{14}$$

Suppose on the contrary that $a_r = b_r = 0$. Based on (13), we have

$$N_0^{[r]}(x_r)W_1^{[r]}(x_r) = N_1^{[r]}(x_r)W_0^{[r]}(x_r), \tag{15}$$

which implies that $(x - x_r) \mid (N_0^{[r]}(x)W_1^{[r]}(x) - N_1^{[r]}(x)W_0^{[r]}(x))$, a contradiction to (14). ■

We introduce a quantity $\delta^{[r]}$, defined as

$$\delta^{[r]} \triangleq \begin{cases} 1, & \text{if } \big(\text{rank}_0^{[r]} < \text{rank}_1^{[r]} \text{ and } b_r = 0\big) \text{ or} \\ & \quad \big(\text{rank}_0^{[r]} > \text{rank}_1^{[r]} \text{ and } a_r \neq 0\big); \\ 0, & \text{if } \big(\text{rank}_0^{[r]} < \text{rank}_1^{[r]} \text{ and } b_r \neq 0\big) \text{ or} \\ & \quad \big(\text{rank}_0^{[r]} > \text{rank}_1^{[r]} \text{ and } a_r = 0\big). \end{cases} \tag{16}$$

We have the following result.

Theorem 1 (modified WB algorithm):

$$\begin{pmatrix} W_0^{[r+1]}(x) & N_0^{[r+1]}(x) \\ W_1^{[r+1]}(x) & N_1^{[r+1]}(x) \end{pmatrix} = \begin{pmatrix} -a_r & b_r \\ (x - x_r)(1 - \delta^{[r]}) & (x - x_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} W_0^{[r]}(x) & N_0^{[r]}(x) \\ W_1^{[r]}(x) & N_1^{[r]}(x) \end{pmatrix}. \tag{17}$$

Proof: According to the definition of $\delta^{[r]}$ given in (16), we need to consider four cases. We prove for the following two cases, one for $\delta^{[r]} = 1$ and one for $\delta^{[r]} = 0$. The other two cases can be proved similarly and are omitted.

1) $\text{rank}_0^{[r]} < \text{rank}_1^{[r]}$ and $b_r = 0$.

In this case, $\delta^{[r]} = 1$. By Lemma 2, $a_r \neq 0$. It is easily verified that $(W_0^{[r+1]}(x), N_0^{[r+1]}(x)) = (-a_r W_0^{[r]}(x), -a_r N_0^{[r]}(x))$ and $(W_1^{[r+1]}(x), N_1^{[r+1]}(x)) = ((x - x_r)W_1^{[r]}(x), (x - x_r)N_1^{[r]}(x))$ are two solutions of Problem $1(r + 1)$. Based on (11) and (12), we have

$$\begin{aligned} &\text{rank}(-a_r W_0^{[r]}(x), -a_r N_0^{[r]}(x)) \\ &\quad + \text{rank}((x - x_r)W_1^{[r]}(x), (x - x_r)N_1^{[r]}(x)) \\ &= 2r + 3, \end{aligned} \tag{18}$$

and

$$\begin{aligned} &(-a_r N_0^{[r]}(x))((x - x_r)W_1^{[r]}(x)) \\ &\quad - ((x - x_r)N_1^{[r]}(x))(-a_r W_0^{[r]}(x)) \\ &= a_r \gamma \prod_{i=0}^{r}(x - x_i). \end{aligned} \tag{19}$$

Therefore, the two solutions $(W_0^{[r+1]}(x), N_0^{[r+1]}(x))$ and $(W_1^{[r+1]}(x), N_1^{[r+1]}(x))$ are complementary.

Algorithm 1: Modified WB algorithm

```
Input: $(x_i, y_i), 0 \le i \le \rho - 1$.
Output: $\big(W(x), N(x)\big)$ satisfying that $N(x_i) = y_i W(x_i)$
    for $0 \le i \le \rho - 1$, and $\text{rank}(W(x), N(x))$ is minimized.
1: Initialization:
   $\begin{pmatrix} W_0^{[0]}(x) & N_0^{[0]}(x) \\ W_1^{[0]}(x) & N_1^{[0]}(x) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$,
   $\begin{pmatrix} \text{rank}_0^{[0]} \\ \text{rank}_1^{[0]} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
2: for $r = 0, 1, \cdots, \rho - 1$ do
3:     $\begin{pmatrix} b_r \\ a_r \end{pmatrix} = \begin{pmatrix} N_0^{[r]}(x_r) - y_r W_0^{[r]}(x_r) \\ N_1^{[r]}(x_r) - y_r W_1^{[r]}(x_r) \end{pmatrix}$
4:     Let $\delta^{[r]} = \big((\text{rank}_0^{[r]} < \text{rank}_1^{[r]})\ \&\&\ (b_r = 0)\big)\ ||\ \big((\text{rank}_0^{[r]} > \text{rank}_1^{[r]})\ \&\&\ (a_r \neq 0)\big)$
5:     $\begin{pmatrix} W_0^{[r+1]}(x) & N_0^{[r+1]}(x) \\ W_1^{[r+1]}(x) & N_1^{[r+1]}(x) \end{pmatrix} =$
       $\begin{pmatrix} -a_r & b_r \\ (x - x_r)(1 - \delta^{[r]}) & (x - x_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} W_0^{[r]}(x) & N_0^{[r]}(x) \\ W_1^{[r]}(x) & N_1^{[r]}(x) \end{pmatrix}$
6:     if $\delta^{[r]} = 1$ then
7:         $\begin{pmatrix} \text{rank}_0^{[r+1]} \\ \text{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \text{rank}_0^{[r]} \\ \text{rank}_1^{[r]} + 2 \end{pmatrix}$
8:     else
9:         $\begin{pmatrix} \text{rank}_0^{[r+1]} \\ \text{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \text{rank}_1^{[r]} \\ \text{rank}_0^{[r]} + 2 \end{pmatrix}$
10:    end if
11: end for
12: if $\text{rank}_0^{[\rho]} < \text{rank}_1^{[\rho]}$ then
13:     return $\big(W_0^{[\rho]}(x), N_0^{[\rho]}(x)\big)$
14: else
15:     return $\big(W_1^{[\rho]}(x), N_1^{[\rho]}(x)\big)$
16: end if
```

2) $\text{rank}_0^{[r]} < \text{rank}_1^{[r]}$ and $b_r \neq 0$.

In this case, $\delta^{[r]} = 0$. It is easily verified that $(W_0^{[r+1]}(x), N_0^{[r+1]}(x)) = (-a_r W_0^{[r]}(x) + b_r W_1^{[r]}(x), -a_r N_0^{[r]}(x) + b_r N_1^{[r]}(x))$ and $(W_1^{[r+1]}(x), N_1^{[r+1]}(x)) = ((x - x_r)W_0^{[r]}(x), (x - x_r)N_0^{[r]}(x))$ are two solutions of Problem $1(r + 1)$. Based on (11) and (12), we have

$$\begin{aligned} &\text{rank}(-a_r W_0^{[r]}(x) + b_r W_1^{[r]}(x), -a_r N_0^{[r]}(x) + b_r N_1^{[r]}(x)) \\ &\quad + \text{rank}((x - x_r)W_0^{[r]}(x), (x - x_r)N_0^{[r]}(x)) \\ &= \text{rank}(W_1^{[r]}(x), N_1^{[r]}(x)) \\ &\quad + \text{rank}((x - x_r)W_0^{[r]}(x), (x - x_r)N_0^{[r]}(x)) \\ &= 2r + 3, \end{aligned} \tag{20}$$

and

$$\begin{aligned}&(-a_rN_0^{[r]}(x)+b_rN_1^{[r]}(x))((x-x_r)W_0^{[r]}(x))\\&\quad-((x-x_r)N_0^{[r]}(x))(-a_rW_0^{[r]}(x)+b_rW_1^{[r]}(x))\\&=-b_r(x-x_r)(N_0^{[r]}(x)W_1^{[r]}(x)-N_1^{[r]}(x)W_0^{[r]}(x))\\&=-b_r\gamma\prod_{i=0}^{r}(x-x_i).\end{aligned} \tag{21}$$

Therefore, the two solutions $(W_0^{[r+1]}(x), N_0^{[r+1]}(x))$ and $(W_1^{[r+1]}(x), N_1^{[r+1]}(x))$ are complementary. ■

Based on Theorem 1, we present a modified WB algorithm in Algorithm 1. The algorithm is iterative in nature, producing the solution to Problem 2 after ρ iterations.

Remark 1: The modified WB algorithm is different from the WB algorithm [3] in two aspects. First, Lemma 2 is not needed in deriving the WB algorithm in which a_r is not computed. Second, the WB algorithm keeps $\mathrm{rank}_0^{[r]} < \mathrm{rank}_1^{[r]}$ for all $0 \le r \le \rho$. If $\mathrm{rank}_0^{[r]} > \mathrm{rank}_1^{[r]}$, then a swap operation $(W_0^{[r]}(x), N_0^{[r]}(x)) \leftrightarrow (W_1^{[r]}(x), N_1^{[r]}(x))$ is performed to make $\mathrm{rank}_0^{[r]} < \mathrm{rank}_1^{[r]}$.

B. Parallel WB algorithm

For $0 \le r \le \rho$, let

$$\begin{pmatrix} b_i^{[r]} \\ a_i^{[r]} \end{pmatrix} \triangleq \begin{pmatrix} N_0^{[r]}(x_i) - y_iW_0^{[r]}(x_i) \\ N_1^{[r]}(x_i) - y_iW_1^{[r]}(x_i) \end{pmatrix}, \quad 0 \le i \le \rho-1. \tag{22}$$

Then based on (13) and (22), it is clear that

$$\begin{pmatrix} b_r^{[r]} \\ a_r^{[r]} \end{pmatrix} = \begin{pmatrix} b_r \\ a_r \end{pmatrix}. \tag{23}$$

We have the following result.

Theorem 2 (parallel WB algorithm): For any $0 \le r < \rho$, for all $0 \le i \le \rho - 1$,

$$\begin{aligned}&\begin{pmatrix} b_i^{[r+1]} \\ a_i^{[r+1]} \end{pmatrix}\\&=\begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (x_i-x_r)(1-\delta^{[r]}) & (x_i-x_r)\delta^{[r]} \end{pmatrix}\begin{pmatrix} b_i^{[r]} \\ a_i^{[r]} \end{pmatrix},\end{aligned} \tag{24}$$

with the initialization

$$\begin{pmatrix} b_i^{[0]} \\ a_i^{[0]} \end{pmatrix} = \begin{pmatrix} -y_i \\ 1 \end{pmatrix}. \tag{25}$$

Proof: According to Algorithm 1, $(W_0^{[0]}(x), N_0^{[0]}(x)) = (1, 0)$ and $(W_1^{[0]}(x), N_1^{[0]}(x)) = (0, 1)$. Then it follows from (22) that (25) holds. Based on (22), we have for all $0 \le i \le \rho - 1$,

$$\begin{aligned}&\begin{pmatrix} b_i^{[r+1]} \\ a_i^{[r+1]} \end{pmatrix}\\&=\begin{pmatrix} N_0^{[r+1]}(x_i) - y_iW_0^{[r+1]}(x_i) \\ N_1^{[r+1]}(x_i) - y_iW_1^{[r+1]}(x_i) \end{pmatrix}\\&=\begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (x_i-x_r)(1-\delta^{[r]}) & (x_i-x_r)\delta^{[r]} \end{pmatrix}\begin{pmatrix} N_0^{[r]}(x_i) - y_iW_0^{[r]}(x_i) \\ N_1^{[r]}(x_i) - y_iW_1^{[r]}(x_i) \end{pmatrix}\\&=\begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (x_i-x_r)(1-\delta^{[r]}) & (x_i-x_r)\delta^{[r]} \end{pmatrix}\begin{pmatrix} a_i^{[r]} \\ b_i^{[r]} \end{pmatrix},\end{aligned}$$

where the second equality follows from (17). ■

Combining Algorithm 1 and Theorem 2, we present the

Algorithm 2: Parallel WB algorithm

```
**Input:** $(x_i, y_i), 0 \le i \le \rho-1$.
**Output:** $\big(W(x), N(x)\big)$ satisfying that $N(x_i) = y_iW(x_i)$
    for $0 \le i \le \rho-1$, and $\mathrm{rank}(W(x), N(x))$ is minimized.
1:  Initialization:
    $\begin{pmatrix} W_0^{[0]}(x) & N_0^{[0]}(x) \\ W_1^{[0]}(x) & N_1^{[0]}(x) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} b_i^{[0]} \\ a_i^{[0]} \end{pmatrix} = \begin{pmatrix} -y_i \\ 1 \end{pmatrix}, 0 \le i \le \rho-1, \begin{pmatrix} \mathrm{rank}_0^{[0]} \\ \mathrm{rank}_1^{[0]} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
2:  for $r = 0, 1, \cdots, \rho-1$ do
3:      Let $\delta^{[r]} = \big((\mathrm{rank}_0^{[r]} < \mathrm{rank}_1^{[r]})\ \&\&\ (b_r^{[r]} = 0)\big)\ ||\ \big((\mathrm{rank}_0^{[r]} > \mathrm{rank}_1^{[r]})\ \&\&\ (a_r^{[r]} \ne 0)\big)$
4:      for $i = 0, 1, \cdots, \rho-1$ do
5:          $\begin{pmatrix} b_i^{[r+1]} \\ a_i^{[r+1]} \end{pmatrix} = \begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (x_i-x_r)(1-\delta^{[r]}) & (x_i-x_r)\delta^{[r]} \end{pmatrix}\begin{pmatrix} b_i^{[r]} \\ a_i^{[r]} \end{pmatrix}$
6:      end for
7:      $\begin{pmatrix} W_0^{[r+1]}(x) & N_0^{[r+1]}(x) \\ W_1^{[r+1]}(x) & N_1^{[r+1]}(x) \end{pmatrix} = \begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (x-x_r)(1-\delta^{[r]}) & (x-x_r)\delta^{[r]} \end{pmatrix}\begin{pmatrix} W_0^{[r]}(x) & N_0^{[r]}(x) \\ W_1^{[r]}(x) & N_1^{[r]}(x) \end{pmatrix}$
8:      if $\delta^{[r]} = 1$ then
9:          $\begin{pmatrix} \mathrm{rank}_0^{[r+1]} \\ \mathrm{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \mathrm{rank}_0^{[r]} \\ \mathrm{rank}_1^{[r]} + 2 \end{pmatrix}$
10:     else
11:         $\begin{pmatrix} \mathrm{rank}_0^{[r+1]} \\ \mathrm{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \mathrm{rank}_1^{[r]} \\ \mathrm{rank}_0^{[r]} + 2 \end{pmatrix}$
12:     end if
13: end for
14: if $\mathrm{rank}_0^{[\rho]} < \mathrm{rank}_1^{[\rho]}$ then
15:     return $\big(W_0^{[\rho]}(x),\ N_0^{[\rho]}(x)\big)$
16: else
17:     return $\big(W_1^{[\rho]}(x),\ N_1^{[\rho]}(x)\big)$
18: end if
```

parallel WB algorithm in Algorithm 2. Based on the update rule for $b_i^{[r+1]}$ and $a_i^{[r+1]}$ (refer to Lines 4-6), it is easily obtained that $a_i^{[r+1]} = b_i^{[r+1]} = 0$ for all $0 \le i \le r$. Therefore, Line 4 can actually be replaced by "**for** $i = r+1, \ldots, \rho-1$ **do**". It can be noticed that only $a_r^{[r]}$ and $b_r^{[r]}$ are used for polynomial update in the index-r iteration. The following explains why we need to compute $a_i^{[r]}$ and $b_i^{[r]}$ for all $0 \le i \le \rho - 1$ (or rather, $r \le i \le \rho - 1$) in the index-$(r-1)$ iteration. Taking $a_{r+1}^{[r]}$ and $b_{r+1}^{[r]}$ as an example, if they were not computed in the index-$(r-1)$ iteration, then we would not be able to compute $a_{r+1}^{[r+1]}$ and $b_{r+1}^{[r+1]}$ in the index-r iteration, which shall be used in the index-$(r+1)$ iteration. The rest may be deduced by analogy.

Remark 2: For both the modified WB algorithm and the parallel WB algorithm, the computed discrepancies are involved in the polynomial update. The difference is as follows. For the modified WB algorithm, since the discrepancies are computed temporarily in each iteration, the polynomial update is performed after the discrepancy computation. Whereas for the parallel WB algorithm, since the preceding iteration has worked out the discrepancies for the current iteration in advance, the discrepancy computation (computing the discrepancies for the next iteration) and the polynomial update are conducted in parallel, hence the name parallel WB algorithm. In this sense, the PWB algorithm is to the WB algorithm what the Reformulated inversionless BM (RiBM) algorithm [12] is to the BM algorithm. Our derivation of the PWB algorithm is in the same sprit as that of the RiBM algorithm, i.e., deserializing discrepancy computation and polynomial update. For hardware implementation, each iteration is usually accomplished in one clock cycle. As a consequence, the parallel WB algorithm is preferred in terms of reducing the critical path (which is dominated by the discrepancy computation and the polynomial update).

IV. Early-Terminating Parallel Welch–Berlekamp Algorithm

In this section, we show that the parallel WB algorithm can be terminated early when applied to the decoding of RS codes. We take the polynomial-evaluation-based RS codes [7] as an example to explore the early termination mechanism.

Let $\mathbb{F}_{2^m}$ be a binary extension field and let $\{v_j\}_{j=0}^{m-1}$ denote a basis of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$. Let $\{\omega_i\}_{i=0}^{2^m-1}$ be the elements of $\mathbb{F}_{2^m}$ represented as

$$\omega_i = \sum_{j=0}^{m-1} i_j v_j,\ i_j \in \mathbb{F}_2, \tag{26}$$

where $\{i_j\}_{j=0}^{m-1}$ is the binary representation of i, i.e., $i = \sum_{j=0}^{m-1} i_j 2^j$.

A full-length $(2^m, k)$ RS code over $\mathbb{F}_{2^m}$ can be defined as

$$\begin{aligned}\mathcal{C} \triangleq \big\{ (f(\omega_0), f(\omega_1), \cdots, f(\omega_{2^m-1})) :\\ f(x) \in \mathbb{F}_{2^m}[x], \deg(f(x)) < k \big\}.\end{aligned} \tag{27}$$

We see that each codeword is a multi-point polynomial evaluation and that the code symbols are indexed by the points $\{\omega_i\}_{i=0}^{2^m-1}$. The multi-point evaluation can be efficiently performed through LCH-FFT [5]. Let t denote the error correction capability of the code. By assuming that the number of parity symbols $2^m - k$ is even, we have $2t = 2^m - k$.

For systematic encoding, the reader can refer to [7], [8], and [10] for details. For a full-length $(2^m, k)$ systematic RS code, we consider two ways to place information and parity symbols, as illustrated in Fig. 1. As we shall see below, this may bring a slight difference in error evaluation. In Fig. 1-a), parity symbols and information symbols are placed on the left and the right of the block, respectively. In Fig. 1-b), the positions for information and parity symbols are exchanged as compared to Fig. 1-a). Also shown in Fig. 1 are code shortening methods. By code shortening it is meant that the symbols in a prescribed set of information positions are set to be zero and thus are not transmitted. These symbols are called shortened symbols. Let the number of shortened symbols be denoted by n_s. The shortened code has the parameter $(n = 2^m - n_s, k - n_s)$. In Fig. 1-a) and Fig. 1-b), the shortened symbols are positioned on the right side and the left side of the block, respectively.

Let $(c_0, c_1, \cdots, c_{2^m-1}) \in \mathcal{C}$ be the transmitted codeword. Assume that the received vector is given by

$$\begin{aligned}(r_0, r_1, \cdots, r_{2^m-1}) =& (c_0, c_1, \cdots, c_{2^m-1})\\ &+ (e_0, e_1, \cdots, e_{2^m-1}),\end{aligned} \tag{28}$$

where $(e_0, e_1, \cdots, e_{2^m-1})$ is the error vector over $\mathbb{F}_{2^m}$ and the operator '+' denotes the component-wise addition in $\mathbb{F}_{2^m}$. Define the error locator polynomial

$$\Lambda(x) = \prod_{\omega \in \mathcal{E}} (x - \omega), \tag{29}$$

where $\mathcal{E} = \{\omega_i : e_i \neq 0\}$ is the set of error locators. The number of errors is equal to $e = |\mathcal{E}|$.

Let $S(x)$ be the syndrome polynomial (refer to [7] for the definition and computation) corresponding to the received vector. Then the key equation is given by [7],

$$Z(x) = S(x)\Lambda(x) \mod \prod_{i=0}^{2^m-k-1} (x - \omega_i), \tag{30}$$

where $\deg(Z(x)) < \deg(\Lambda(x))$. For a full-length $(2^m, k)$ RS code, the error value in the error locator $\omega_i \in \mathcal{E}$ can be computed as [7]

$$e_i = \begin{cases} \frac{Z(\omega_i)}{\prod_{j=0}^{2^m-k-1}(\omega_i - \omega_j)\cdot\Lambda'(\omega_i)}, & \text{if } 2^m - k \le i \le 2^m - 1, \\ \frac{Z'(\omega_i) - S(\omega_i)\Lambda'(\omega_i)}{\prod_{j=0, j\neq i}^{2^m-k-1}(\omega_i - \omega_j)\cdot\Lambda'(\omega_i)}, & \text{if } 0 \le i < 2^m - k. \end{cases} \tag{31}$$

Note that (31) applies not only to systematic codes (regardless of how information and parity symbols are placed) but also to non-systematic codes. We see that $2^m - k$ is the dividing point for the two cases in (31). If we place the information and parity symbols as in Fig. 1-a), then for both full-length

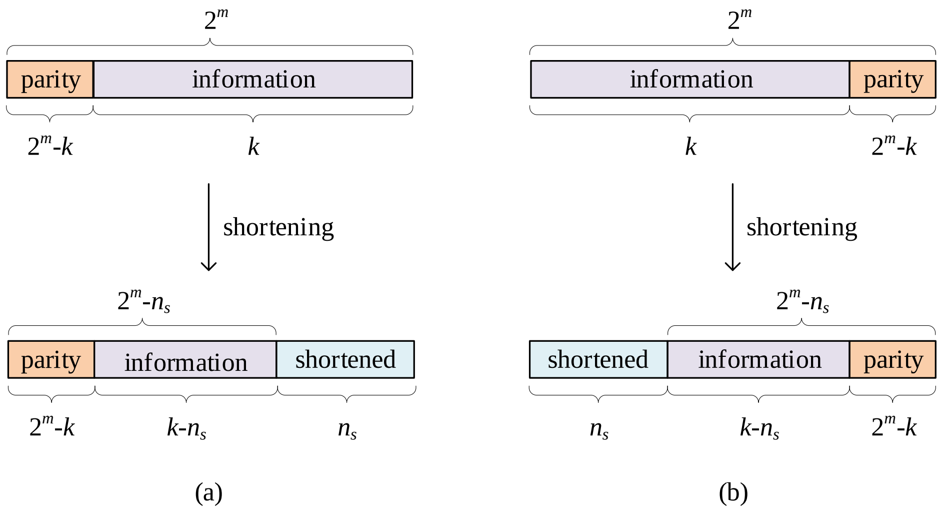


Fig. 1. Two ways to put information and parity symbols for full-length RS codes and the code shortening methods: (a) Information symbols on the right; (b) Information symbols on the left.

and shortened systematic codes, it follows from (31) that

$$e_i = \begin{cases} \frac{Z(\omega_i)}{\prod_{j=0}^{2^m-k-1}(\omega_i-\omega_j)\cdot\Lambda'(\omega_i)}, & \text{if } i \text{ is an information position (i.e., } i \geq 2^m-k), \\ \frac{Z'(\omega_i)-S(\omega_i)\Lambda'(\omega_i)}{\prod_{j=0,j\neq i}^{2^m-k-1}(\omega_i-\omega_j)\cdot\Lambda'(\omega_i)}, & \text{if } i \text{ is a parity position (i.e., } i < 2^m-k). \end{cases} \tag{32}$$

For a shortened systematic code as given in Fig. 1-b), by assuming $n_s \geq 2^m - k$, it follows from (31) that

$$e_i = \frac{Z(\omega_i)}{\prod_{j=0}^{2^m-k-1}(\omega_i-\omega_j)\cdot\Lambda'(\omega_i)}, \quad 2^m - n_s \leq i \leq 2^m - 1, \tag{33}$$

where i can be either an information position or a parity position. The error evaluation turns out to be uniform for both information and parity symbols. Here, we adopt the coordinate system of the original full-length code for the shortened code, i.e., the range of the symbol index i for the shortened code is $2^m - n_s \leq i \leq 2^m - 1$.

Now we introduce the concept of *incomplete error locator polynomial*. Given the error locator polynomial $\Lambda(x)$ in (29), a polynomial $\Lambda_1(x)$ is called an incomplete error locator polynomial if

$$\Lambda_1(x) \mid \Lambda(x). \tag{34}$$

and

$$\prod_{\omega_i\in\mathcal{E}, i\geq 2^m-k}(x-\omega_i) \mid \Lambda_1(x). \tag{35}$$

By defining

$$\Lambda_0(x) \triangleq \frac{\Lambda(x)}{\Lambda_1(x)}, \tag{36}$$

we have $\Lambda_0(x) \mid \prod_{\omega_i\in\mathcal{E}, i<2^m-k}(x-\omega_i)$. Based on (30), we have $\Lambda_0(x) \mid Z(x)$. We call

$$Z_1(x) \triangleq \frac{Z(x)}{\Lambda_0(x)} \tag{37}$$

the *incomplete error evaluator polynomial* associated with the incomplete error locator polynomial $\Lambda_1(x)$.

Note that for a shortened RS code as given in Fig. 1-b), the errors will not occur in the shortened positions $\{i : 0 \leq i < n_s\}$. Therefore, if the number of shortened symbols is such that $n_s \geq 2^m - k$, then $(x-\omega_i) \nmid \Lambda(x)$ for any $0 \leq i < 2^m - k$, which implies that $\Lambda_1(x) = \Lambda(x)$ and further $Z_1(x) = Z(x)$.

We have the following result.

Lemma 3: If $\Lambda_1(x)$ is an incomplete error locator polynomial and $Z_1(x)$ is the associated incomplete error evaluator polynomial, then the error values in the error locators $\omega_i \in \mathcal{E}$ with $i \geq 2^m - k$ can be alternatively computed as

$$e_i = \frac{Z_1(\omega_i)}{\prod_{j=0}^{2^m-k-1}(\omega_i-\omega_j)\cdot\Lambda_1'(\omega_i)}. \tag{38}$$

Proof: Based on (31), for $i \geq 2^m - k$, we have

$$\begin{aligned} e_i &= \frac{Z(\omega_i)}{\prod_{j=0}^{2^m-k-1}(\omega_i-\omega_j)\cdot\Lambda'(\omega_i)} \\ &= \frac{\Lambda_0(\omega_i)Z_1(\omega_i)}{\prod_{j=0}^{2^m-k-1}(\omega_i-\omega_j)\cdot(\Lambda_0'(\omega_i)\Lambda_1(\omega_i)+\Lambda_0(\omega_i)\Lambda_1'(\omega_i))} \\ &= \frac{\Lambda_0(\omega_i)Z_1(\omega_i)}{\prod_{j=0}^{2^m-k-1}(\omega_i-\omega_j)\cdot\Lambda_0(\omega_i)\Lambda_1'(\omega_i)} \\ &= \frac{Z_1(\omega_i)}{\prod_{j=0}^{2^m-k-1}(\omega_i-\omega_j)\cdot\Lambda_1'(\omega_i)}. \end{aligned} \tag{39}$$

■

To solve the key equation (30), we can set $\rho = 2t$, $x_i = \omega_i$ and $y_i = S(\omega_i)$ in Algorithm 2. If $e \leq t$, by virtue of the decoding problem itself, the key equation has a unique solution $(\Lambda(x), Z(x))$ such that $\deg(Z(x)) < \deg(\Lambda(x)) = e$, which is equal (up to a scalar) to the algorithm output [3], [7].

Theorem 3: In Algorithm 2, if $e \le t$, then there exists some $r \le t+e$ such that $\mathrm{rank}_1^{[r]} = 2t+1$. Furthermore, for any $0 < r \le 2t$, if $\mathrm{rank}_1^{[r]} = 2t+1$, then $\mathrm{rank}_0^{[r]} < \mathrm{rank}_1^{[r]}$.

Proof: Based on the initialization of $\mathrm{rank}_0^{[0]}$ and $\mathrm{rank}_1^{[1]}$ (refer to Line 1 of Algorithm 2) and the update of $\mathrm{rank}_0^{[r]}$ and $\mathrm{rank}_1^{[r]}$ (refer to Lines 8-11), it is easily verified by induction that

$$\mathrm{rank}_0^{[r]} + \mathrm{rank}_1^{[r]} = 2r+1, \quad 0 \le r \le 2t, \tag{40}$$

which is exactly (11) as required by the definition of complementary solutions. Since $\mathrm{rank}_0^{[2t]} + \mathrm{rank}_1^{[2t]} = 4t+1$, we have

$$\max\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\} \ge 2t+1. \tag{41}$$

Since $e \le t$, we have

$$\begin{aligned}\min\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\} &= \mathrm{rank}\{\Lambda(x), Z(x)\} \\ &= \max\{2\deg(\Lambda(x)), 1 + 2\deg(Z(x))\} \\ &= 2\deg(\Lambda(x)), \end{aligned} \tag{42}$$

where the first equality is due to the fact that the algorithm output $(\Lambda(x), Z(x))$ has a lower rank (note that the algorithm solves Problem 2(r)). Therefore, $\min\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\}$ is even.

First, we show that there exists some r, $0 \le r \le 2t$, such that $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} = 2t+1$. Suppose, on the contrary, that no such r exists. Then from (41), $\max\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\} \ge 2t+2$. It follows from Lines 8-11 of Algorithm 2 that when r is increased by 1, $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\}$ is increased at most by 2. This implies that as r increases, $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\}$ is guaranteed to take at least one of any two consecutive integers which are less than or equal to $2t+2$. Therefore, due to (41) and by assumption $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} \ne 2t+1$ for any r, there must exist some r', $0 \le r' \le 2t$, such that $\max\{\mathrm{rank}_0^{[r']}, \mathrm{rank}_1^{[r']}\} = 2t+2$. We now show that when $r \ge r'$, $\max\{\mathrm{rank}_0^{[r+1]}, \mathrm{rank}_1^{[r+1]}\}$ is either equal to $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\}$ or $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\}+2$. Consider the following two possible cases for $r \ge r'$. It is clear from (40) that $\mathrm{rank}_0^{[r]} \ne \mathrm{rank}_1^{[r]}$.

1) $\delta^{[r]} = 1$.
   Then $\mathrm{rank}_0^{[r+1]} = \mathrm{rank}_0^{[r]}$ and $\mathrm{rank}_1^{[r+1]} = \mathrm{rank}_1^{[r]} + 2$.
   - If $\mathrm{rank}_0^{[r]} < \mathrm{rank}_1^{[r]}$, then $\max\{\mathrm{rank}_0^{[r+1]}, \mathrm{rank}_1^{[r+1]}\} = \max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} + 2$.
   - If $\mathrm{rank}_0^{[r]} > \mathrm{rank}_1^{[r]}$, then $\max\{\mathrm{rank}_0^{[r+1]}, \mathrm{rank}_1^{[r+1]}\} = \max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\}$, which can be seen as follows. Based on (40), $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} + \min\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} \le 4t+1$ for $r \le 2t$. Then it follows from $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} \ge 2t+2$ that $\min\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} \le 2t-1$ for $r \ge r'$. Therefore, $\mathrm{rank}_0^{[r]} - (\mathrm{rank}_1^{[r]} + 2) = \max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} - \min\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} - 2 \ge (2t+2) - (2t-1) - 2 > 0$, which implies $\max\{\mathrm{rank}_0^{[r+1]}, \mathrm{rank}_1^{[r+1]}\} = \max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]} + 2\} = \mathrm{rank}_0^{[r]} = \max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\}$.
2) $\delta^{[r]} = 0$.
   Then $\mathrm{rank}_0^{[r+1]} = \mathrm{rank}_1^{[r]}$ and $\mathrm{rank}_1^{[r+1]} = \mathrm{rank}_0^{[r]} + 2$.
   - If $\mathrm{rank}_0^{[r]} > \mathrm{rank}_1^{[r]}$, then $\max\{\mathrm{rank}_0^{[r+1]}, \mathrm{rank}_1^{[r+1]}\} = \max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} + 2$.
   - If $\mathrm{rank}_0^{[r]} < \mathrm{rank}_1^{[r]}$, then $\max\{\mathrm{rank}_0^{[r+1]}, \mathrm{rank}_1^{[r+1]}\} = \max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\}$, which can be seen as follows. Based on (40), $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} + \min\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} \le 4t+1$ for $r \le 2t$. Then it follows from $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} \ge 2t+2$ that $\min\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} \le 2t-1$ for $r \ge r'$. Therefore, $\mathrm{rank}_1^{[r]} - (\mathrm{rank}_0^{[r]} + 2) = \max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} - \min\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} - 2 \ge (2t+2) - (2t-1) - 2 > 0$, which implies $\max\{\mathrm{rank}_0^{[r+1]}, \mathrm{rank}_1^{[r+1]}\} = \max\{\mathrm{rank}_1^{[r]}, \mathrm{rank}_0^{[r]} + 2\} = \mathrm{rank}_1^{[r]} = \max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\}$.

Since $\max\{\mathrm{rank}_0^{[r']}, \mathrm{rank}_1^{[r']}\} = 2t+2$, then $\max\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\}$ must be even. This is in contradiction to that $\min\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\}$ is even because $\max\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\} + \min\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\} = 4t+1$ is odd. This proves that there exists some r, $0 \le r \le 2t$, such that $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} = 2t+1$.

Let r_1 be the smallest integer such that $\max\{\mathrm{rank}_0^{[r_1]}, \mathrm{rank}_1^{[r_1]}\} = 2t+1$. Next, we show that $r_1 \le t+e$. Since $e \le t$ and $\mathrm{rank}_0^{[t+e]} + \mathrm{rank}_1^{[t+e]} = 2t+2e+1$, we have $\max\{\mathrm{rank}_0^{[t+e]}, \mathrm{rank}_1^{[t+e]}\} \ge 2t+1$ because $\min\{\mathrm{rank}_0^{[t+e]}, \mathrm{rank}_1^{[t+e]}\} \le \min\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\} = 2\deg(\Lambda(x)) = 2e$ (note that the inequality here is due to that $\min\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\}$ is non-decreasing with r, refer to Lines 8-11 of Algorithm 2). Therefore, given that there exists some r, $0 \le r \le 2t$, such that $\max\{\mathrm{rank}_0^{[r]}, \mathrm{rank}_1^{[r]}\} = 2t+1$, the smallest r_1 such that $\max\{\mathrm{rank}_0^{[r_1]}, \mathrm{rank}_1^{[r_1]}\} = 2t+1$ must satisfy $r_1 \le t+e$.

Finally, we show that $\mathrm{rank}_0^{[r_1]} < \mathrm{rank}_1^{[r_1]}$. Suppose on the contrary that $\mathrm{rank}_0^{[r_1]} > \mathrm{rank}_1^{[r_1]}$, which implies $\mathrm{rank}_0^{[r_1]} = 2t+1$. Let $r_0 < r_1$ be the largest integer such that $\mathrm{rank}_0^{[r_0]} < \mathrm{rank}_0^{[r_0+1]} = \mathrm{rank}_0^{[r_0+2]} = \cdots = \mathrm{rank}_0^{[r_1]}$. Since $\mathrm{rank}_0^{[r_0+1]}$ is equal to either $\mathrm{rank}_0^{[r_0]}$ or $\mathrm{rank}_1^{[r_0]}$, we have $\mathrm{rank}_0^{[r_0+1]} = \mathrm{rank}_1^{[r_0]}$, which implies that $\mathrm{rank}_1^{[r_0]} = \mathrm{rank}_0^{[r_1]} = 2t+1$. Therefore, we have $\max\{\mathrm{rank}_0^{[r_0]}, \mathrm{rank}_1^{[r_0]}\} = 2t+1$, in contradiction to that r_1 is the smallest integer such that $\max\{\mathrm{rank}_0^{[r_1]}, \mathrm{rank}_1^{[r_1]}\} = 2t+1$. Therefore, $\mathrm{rank}_0^{[r_1]} < \mathrm{rank}_1^{[r_1]}$. This completes the proof of the first half of the theorem that if $e \le t$, there exists some $r \le t+e$ such that $\mathrm{rank}_1^{[r]} = 2t+1$.

For any $0 < r \le 2t$ such that $\mathrm{rank}_1^{[r]} = 2t+1$, it follows from (40) that $\mathrm{rank}_0^{[r]} = 2r+1-\mathrm{rank}_1^{[r]} = 2r-2t \le 2t < \mathrm{rank}_1^{[r]}$. This completes the proof of the second half of the theorem. ■

The following result shows that the PWB algorithm can be terminated early.

Theorem 4: Assuming $e \leq t$, Algorithm 2 has the following properties.

(i). If $\text{rank}_0^{[2e]} < \text{rank}_1^{[2e]}$, then $W_0^{[2e]}(x)$ is an incomplete error locator polynomial and $N_0^{[2e]}(x)$ is the associated incomplete error evaluator polynomial, otherwise, $W_1^{[2e]}(x)$ is an incomplete error locator polynomial and $N_1^{[2e]}(x)$ is the associated incomplete error evaluator polynomial.

(ii). Let r_1 be the smallest integer such that $\text{rank}_1^{[r_1]} = 2t+1$ (by Theorem 3, $r_1 \leq t+e$), then $W_0^{[r_1]}(x)$ is an incomplete error locator polynomial and $N_0^{[r_1]}(x)$ is the associated incomplete error evaluator polynomial.

(iii). The error values in the error locators $\omega_i \in \mathcal{E}$ with $i \geq 2^m - k$ are given by

$$e_i = \frac{N_0^{[r_1]}(\omega_i)}{\prod_{j=0}^{2^m-k-1}(\omega_i - \omega_j) \cdot W_0^{[r_1]\prime}(\omega_i)}, \tag{43}$$

where r_1 is the smallest integer such that $\text{rank}_1^{[r_1]} = 2t+1$.

(iv). If there are no errors in the positions $\{0 \leq i < 2^m - k\}$, as is always the case for a shortened RS code in Fig. 1-b), then the smallest integer r_1 such that $\text{rank}_1^{[r_1]} = 2t+1$ is equal to $t+e$ and $W_0^{[t+e]}(x) = \gamma\Lambda(x)$ for some nonzero $\gamma \in \mathbb{F}_{2^m}$.

Proof: Let $(W^{[r]}(x), N^{[r]}(x))$ be a polynomial pair defined by

$$(W^{[r]}(x), N^{[r]}(x)) \triangleq \begin{cases} (W_0^{[r]}(x), N_0^{[r]}(x)), & \text{if } \text{rank}_0^{[r]} < \text{rank}_1^{[r]}; \\ (W_1^{[r]}(x), N_1^{[r]}(x)), & \text{if } \text{rank}_0^{[r]} > \text{rank}_1^{[r]}. \end{cases} \tag{44}$$

According to the description of Algorithm 2 (refer to Lines 14-18), we see that the output is given by $(W^{[2t]}(x), N^{[2t]}(x))$, as defined by (44), which is equal (up to a scalar) to $(\Lambda(x), Z(x))$ if $e \leq t$. Under the assumption that $e \leq t$, we have $\min\{\text{rank}_0^{[r]}, \text{rank}_1^{[r]}\} \leq 2e < 2t+1$ for all $0 \leq r \leq 2t$. Based on (40), we have $\max\{\text{rank}_0^{[2e]}, \text{rank}_1^{[2e]}\} \geq 2e+1$. Note that $\max\{\text{rank}_0^{[r]}, \text{rank}_1^{[r]}\}$ is non-decreasing as r increases. Therefore, $\max\{\text{rank}_0^{[r]}, \text{rank}_1^{[r]}\} \geq \max\{\text{rank}_0^{[2e]}, \text{rank}_1^{[2e]}\} \geq 2e+1$ for $2e \leq r \leq 2t$.

(i). Based on (44), we shall prove that $W^{[2e]}(x)$ is an incomplete error locator polynomial and $N^{[2e]}(x)$ is the associated incomplete error evaluator polynomial. When $2e \leq r < 2t$, $(W^{[r+1]}(x), N^{[r+1]}(x))$ is updated based on the following four cases.

1) $\text{rank}_0^{[r]} < \text{rank}_1^{[r]}$ and $b_r^{[r]} \neq 0$.
Then $\delta^{[r]} = 0$. Since $\max\{\text{rank}_0^{[r]}, \text{rank}_1^{[r]}\} = \text{rank}_1^{[r]} \geq 2e+1$ and $\min\{\text{rank}_0^{[r+1]}, \text{rank}_1^{[r+1]}\} \leq 2e$, we have $\min\{\text{rank}_0^{[r+1]}, \text{rank}_1^{[r+1]}\} = \text{rank}_1^{[r+1]} = \text{rank}_0^{[r]} + 2 < \text{rank}_1^{[r]} = \text{rank}_0^{[r+1]}$. Therefore,
$$\begin{aligned}(W^{[r+1]}(x), N^{[r+1]}(x)) &= (W_1^{[r+1]}(x), N_1^{[r+1]}(x)) \\ &= (x-\omega_r)(W_0^{[r]}(x), N_0^{[r]}(x)) \\ &= (x-\omega_r)(W^{[r]}(x), N^{[r]}(x)).\end{aligned} \tag{45}$$

2) $\text{rank}_0^{[r]} < \text{rank}_1^{[r]}$ and $b_r^{[r]} = 0$.
Then $\delta^{[r]} = 1$. Since $\text{rank}_0^{[r+1]} = \text{rank}_0^{[r]} < \text{rank}_1^{[r]} + 2 = \text{rank}_1^{[r+1]}$, we have
$$\begin{aligned}(W^{[r+1]}(x), N^{[r+1]}(x)) &= (W_0^{[r+1]}(x), N_0^{[r+1]}(x)) \\ &= -a_r^{[r]}(W_0^{[r]}(x), N_0^{[r]}(x)) \\ &= -a_r^{[r]}(W^{[r]}(x), N^{[r]}(x)).\end{aligned} \tag{46}$$
Here, by Lemma 2, $a_r^{[r]} \neq 0$.

3) $\text{rank}_0^{[r]} > \text{rank}_1^{[r]}$ and $a_r^{[r]} \neq 0$.
Then $\delta^{[r]} = 1$. Since $\max\{\text{rank}_0^{[r]}, \text{rank}_1^{[r]}\} = \text{rank}_0^{[r]} \geq 2e+1$ and $\min\{\text{rank}_0^{[r+1]}, \text{rank}_1^{[r+1]}\} \leq 2e$, we have $\min\{\text{rank}_0^{[r+1]}, \text{rank}_1^{[r+1]}\} = \text{rank}_1^{[r+1]} = \text{rank}_1^{[r]} + 2 < \text{rank}_0^{[r]} = \text{rank}_0^{[r+1]}$. Therefore,
$$\begin{aligned}(W^{[r+1]}(x), N^{[r+1]}(x)) &= (W_1^{[r+1]}(x), N_1^{[r+1]}(x)) \\ &= (x-\omega_r)(W_1^{[r]}(x), N_1^{[r]}(x)) \\ &= (x-\omega_r)(W^{[r]}(x), N^{[r]}(x)).\end{aligned} \tag{47}$$

4) $\text{rank}_0^{[r]} > \text{rank}_1^{[r]}$ and $a_r^{[r]} = 0$.
Then $\delta^{[r]} = 0$. Since $\text{rank}_0^{[r+1]} = \text{rank}_1^{[r]} < \text{rank}_0^{[r]} + 2 = \text{rank}_1^{[r+1]}$, we have
$$\begin{aligned}(W^{[r+1]}(x), N^{[r+1]}(x)) &= (W_0^{[r+1]}(x), N_0^{[r+1]}(x)) \\ &= b_r^{[r]}(W_1^{[r]}(x), V_1^{[r]}(x)) \\ &= b_r^{[r]}(W^{[r]}(x), N^{[r]}(x)).\end{aligned} \tag{48}$$
Here, by Lemma 2, $b_r^{[r]} \neq 0$.

Based on the above update rule, $(W^{[2t]}(x), N^{[2t]}(x))$ can be written as

$$(W^{[2t]}(x), N^{[2t]}(x)) = \beta\Big(\prod_{i\in A}(x-\omega_i)\Big)(W^{[2e]}(x), N^{[2e]}(x)), \tag{49}$$

for some $\beta \neq 0 \in \mathbb{F}_{2^m}$ and some index set $A \subset \{2e, 2e+1, \ldots, 2t-1\}$. Since $(W^{[2t]}(x), N^{[2t]}(x))$ is equal (up to a scalar) to $(\Lambda(x), Z(x))$, we know from (49) that $W^{[2e]}(x)$ is an incomplete error locator polynomial and $N^{[2e]}(x)$ is the associated incomplete error evaluator polynomial. The proof is complete by noting (44).

(ii). The existence of such an r_1 is guaranteed by Theorem 3 which also indicates that $r_1 \leq t+e$. Similar to the proof of *(i)*, it can be shown that when $r_1 \leq r < 2t$, $(W^{[r+1]}(x), N^{[r+1]}(x))$ has the same update rule as in *(i)*. Here, we only give the proof for the first case: $\text{rank}_0^{[r]} < \text{rank}_1^{[r]}$ and $b_r^{[r]} \neq 0$. In this case, $\delta^{[r]} = 0$. Since $\text{rank}_1^{[r]} = \max\{\text{rank}_0^{[r]}, \text{rank}_1^{[r]}\} \geq \max\{\text{rank}_0^{[r_1]}, \text{rank}_1^{[r_1]}\} = 2t+1$

and $\min\{\mathrm{rank}_0^{[r+1]}, \mathrm{rank}_1^{[r+1]}\} \le \min\{\mathrm{rank}_0^{[2t]}, \mathrm{rank}_1^{[2t]}\} = 2e < 2t+1$, we have $\min\{\mathrm{rank}_0^{[r+1]}, \mathrm{rank}_1^{[r+1]}\} = \mathrm{rank}_1^{[r+1]} = \mathrm{rank}_0^{[r]} + 2 < \mathrm{rank}_1^{[r]} = \mathrm{rank}_0^{[r+1]}$. Therefore, $(W^{[r+1]}(x), N^{[r+1]}(x)) = (W_1^{[r+1]}(x), N_1^{[r+1]}(x)) = (x-\omega_r)(W_0^{[r]}(x), N_0^{[r]}(x)) = (x-\omega_r)(W^{[r]}(x), N^{[r]}(x))$. Consequently, $(W^{[2t]}(x), N^{[2t]}(x))$ can be written as

$$\begin{aligned}(W^{[2t]}(x), N^{[2t]}(x)) &= \beta\Big(\prod_{i\in A}(x-\omega_i)\Big)(W^{[r_1]}(x), N^{[r_1]}(x)) \\ &= \beta\Big(\prod_{i\in A}(x-\omega_i)\Big)(W_0^{[r_1]}(x), N_0^{[r_1]}(x)),\end{aligned} \tag{50}$$

for some $\beta \neq 0 \in \mathbb{F}_{2^m}$ and some index set $A \subset \{2e, 2e+1, \ldots, 2t-1\}$. The second equality in (50) is due to that $\mathrm{rank}_0^{[r_1]} < \mathrm{rank}_1^{[r_1]}$ by Theorem 3. Therefore, $W_0^{[r_1]}(x)$ is an incomplete error locator polynomial and $N_0^{[r_1]}(x)$ is the associated incomplete error evaluator polynomial.

(iii). This follows directly from *(ii)* and Theorem 3.

(iv). Due to (ii) and the assumption that there are no errors in the positions $\{i \ge 2^m - k\}$, we can deduce that

$$(W_0^{[r_1]}(x), N_0^{[r_1]}(x)) = (\gamma\Lambda(x), \gamma Z(x)), \tag{51}$$

for some scalar γ. This makes $b_i^{[r_1]} = 0$ for any $0 \le i \le 2t-1$, according to (22) and the definition of $\Lambda(x)$ and $Z(x)$. This, in turns, makes $\delta^{[r_1]} = 0$ and leads to

$$(W_0^{[r_1+1]}(x), N_0^{[r_1+1]}(x)) = (\gamma'\Lambda(x), \gamma' Z(x)), \tag{52}$$

for (possibly) another scalar γ', according to Line 7 of Algorithm 2. According to Line 9 of Algorithm 2, we have $\mathrm{rank}_0^{[r_1+1]} = \mathrm{rank}_0^{[r_1]}$ and $\mathrm{rank}_0^{[r_1+1]} < \mathrm{rank}_1^{[r_1+1]}$. By the same argument, we have $\mathrm{rank}_0^{[r_1]} = \mathrm{rank}_0^{[r_1+1]} = \cdots = \mathrm{rank}_0^{[2t]} = 2e$. Since $\mathrm{rank}_1^{[r_1]} = 2t+1$ and $\mathrm{rank}_0^{[r_1]} + \mathrm{rank}_1^{[r_1]} = 2r_1 + 1$ by (40), we have $r_1 = t+e$. Note that if all the e errors occur in the positions $\{i \ge 2^m - k\}$, an incomplete error locator polynomial is equal to the error locator polynomial multiplied by a nonzero field element. Then it follows from *(ii)* that $W_0^{[r_1]}(x) = \gamma\Lambda(x)$ for some nonzero $\gamma \in \mathbb{F}_{2^m}$, where $r_1 = t+e$. ■

Remark 3: Although derived in the context of the PWB algorithm, the early-terminating mechanism also applies to the WB algorithm. The algorithm difference, as stated in Remark 2, is not essential to the early-terminating rule.

Remark 4: It was previously known that the Berlekamp–Massey (BM) algorithm can be terminated early [13], [14], [15], [16], [17]. Therefore, it is interesting to make a comparison between the two algorithms in terms of early termination. Assume $e \le t$. The similarity is that both algorithms can be terminated before or at the completion of the $(t+e)$-th iteration. There are also several differences. For the BM algorithm, the error locator polynomial is obtained after $2e$ iterations. Since e is unknown, the early termination is based on a detection method to identify the $(t+e)$-th iteration. Whereas for the WB algorithm, the complete error locator polynomial may not be obtained until the $2t$-th iteration. The early termination is based on a detection method to identify the r_1-th iteration ($r_1 \le t+e$) so that an incomplete error locator polynomial can be obtained. As a consequence, the early-terminating BM algorithm can accomplish the error corrections in all positions while the early-terminating WB algorithm can only accomplish the error corrections in the positions $2^m - k \le i < 2^m$.

Algorithm 3: Early-terminating parallel WB (EPWB) algorithm

```
   Input: $(x_i, y_i), 0 \le i \le 2t-1$.
   Output: $(W_0^{[r_1]}(x), N_0^{[r_1]}(x))$.
 1: Initialization:
    $\begin{pmatrix} W_0^{[0]}(x) & N_0^{[0]}(x) \\ W_1^{[0]}(x) & N_1^{[0]}(x) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} b_i^{[0]} \\ a_i^{[0]} \end{pmatrix} = \begin{pmatrix} -y_i \\ 1 \end{pmatrix}, 0 \le i \le 2t-1, \begin{pmatrix} \mathrm{rank}_0^{[0]} \\ \mathrm{rank}_1^{[0]} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
 2: for $r = 0, 1, \cdots, 2t-1$ do
 3:   Let $\delta^{[r]} = \big((\mathrm{rank}_0^{[r]} < \mathrm{rank}_1^{[r]})\ \&\&\ (b_r^{[r]} = 0)\big)\ ||\ \big((\mathrm{rank}_0^{[r]} > \mathrm{rank}_1^{[r]})\ \&\&\ (a_r^{[r]} \neq 0)\big)$
 4:   for $i = 0, 1, \cdots, 2t-1$ do
 5:     $\begin{pmatrix} b_i^{[r+1]} \\ a_i^{[r+1]} \end{pmatrix} = \begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (x_i - x_r)(1-\delta^{[r]}) & (x_i - x_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} b_i^{[r]} \\ a_i^{[r]} \end{pmatrix}$
 6:   end for
 7:   $\begin{pmatrix} W_0^{[r+1]}(x) & N_0^{[r+1]}(x) \\ W_1^{[r+1]}(x) & N_1^{[r+1]}(x) \end{pmatrix} = \begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (x - x_r)(1-\delta^{[r]}) & (x - x_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} W_0^{[r]}(x) & N_0^{[r]}(x) \\ W_1^{[r]}(x) & N_1^{[r]}(x) \end{pmatrix}$
 8:   if $\delta^{[r]} = 1$ then
 9:     $\begin{pmatrix} \mathrm{rank}_0^{[r+1]} \\ \mathrm{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \mathrm{rank}_0^{[r]} \\ \mathrm{rank}_1^{[r]} + 2 \end{pmatrix}$
10:   else
11:     $\begin{pmatrix} \mathrm{rank}_0^{[r+1]} \\ \mathrm{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \mathrm{rank}_1^{[r]} \\ \mathrm{rank}_0^{[r]} + 2 \end{pmatrix}$
12:   end if
13:   if $((\mathrm{rank}_1^{[r+1]} = 2t+1)\ ||\ (r = 2t-1))$ then
14:     return $(W_0^{[r+1]}(x), N_0^{[r+1]}(x))$
15:   end if
16: end for
```

Based on Theorem 4, we present the early-terminating parallel WB (EPWB) algorithm in Algorithm 3. A difference between the PWB algorithm (Algorithm 2) and the EPWB algorithm is that the former selectively outputs $(W_0^{[2t]}(x), N_0^{[2t]}(x))$ or $(W_1^{[2t]}(x), N_1^{[2t]}(x))$ while the latter always outputs $(W_0^{[r_1]}(x), N_0^{[r_1]}(x))$. Note that in Line 13 of Algorithm 3, we add the subcase $r = 2t-1$. This is to ensure that the algorithm always has an output, considering that the other subcase $\mathrm{rank}_1^{[r+1]} = 2t+1$ may not be encountered if $e > t$. In fact, if $e > t$, it is entirely possible that $\mathrm{rank}_0^{[2t]} > \mathrm{rank}_1^{[2t]}$, in which case Algorithm 2

TABLE I
THE OPERATION DETAILS OF THE PWB ALGORITHM FOR EXAMPLE 1

r	$b_r^{[r]}$	$a_r^{[r]}$	$\text{rank}_0^{[r]}$	$\text{rank}_1^{[r]}$	$\delta^{[r]}$
0	α^{18}	1	0	1	0
1	α^4	α^{17}	1	2	0
2	0	α^{13}	2	3	1
3	0	α^{22}	2	5	1
4	0	α^3	2	7	1
5	α^{18}	α^{23}	2	9	0
6	α^{10}	α^{30}	9	4	1
7	α^{12}	0	9	6	0
8	–	–	6	11	–

r	$W_0^{[r]}(x)$	$N_0^{[r]}(x)$
0	1	0
1	1	α^{18}
2	$\alpha^4x+\alpha^{17}$	α^4
3	$\alpha^{17}x+\alpha^{30}$	α^{17}
4	$\alpha^8x+\alpha^{21}$	α^8
5	$\alpha^{11}x+\alpha^{24}$	α^{11}
6	$\alpha^{18}x^4+\alpha^{20}x^3+\alpha^{29}x^2+\alpha^{18}x+\alpha^{28}$	$\alpha^5x^4+\alpha^7x^3+\alpha^{16}x^2+\alpha^{14}x+\alpha^{15}$
7	$\alpha^{17}x^4+\alpha^{19}x^3+\alpha^{12}x^2+\alpha^{20}x+\alpha^{19}$	$\alpha^4x^4+\alpha^6x^3+\alpha^{15}x^2+\alpha^2x+\alpha^6$
8	$\alpha^{23}x^3+\alpha^7x^2+\alpha^7x+\alpha^{29}$	$\alpha^{23}x^2+\alpha^{10}x+\alpha^{16}$

r	$W_1^{[r]}(x)$	$N_1^{[r]}(x)$
0	0	1
1	x	0
2	$x+1$	$\alpha^{18}x+\alpha^{18}$
3	$x^2+\alpha^{18}x+\alpha$	$\alpha^{18}x^2+\alpha^5x+\alpha^{19}$
4	$x^3+\alpha^{11}x+\alpha^{19}$	$\alpha^{18}x^3+\alpha^{29}x+\alpha^6$
5	$x^4+\alpha^2x^3+\alpha^{11}x^2+\alpha^9x+\alpha^{21}$	$\alpha^{18}x^4+\alpha^{20}x^3+\alpha^{29}x^2+\alpha^{27}x+\alpha^8$
6	$\alpha^{11}x^2+\alpha^5x+\alpha^{29}$	$\alpha^{11}x+\alpha^{16}$
7	$\alpha^{11}x^3+\alpha^{26}x^2+\alpha^{26}x+\alpha^{17}$	$\alpha^{11}x^2+\alpha^{29}x+\alpha^4$
8	$\alpha^{17}x^5+\alpha^4x^4+\alpha^{13}x^3+\alpha^{18}x^2+\alpha^{11}x+\alpha^{30}$	$\alpha^4x^5+\alpha^{22}x^4+\alpha^{20}x^3+\alpha^{17}x^2+\alpha^{28}x+\alpha^{17}$

outputs $(W_1^{[2t]}(x), N_1^{[2t]}(x))$ whereas Algorithm 3 outputs $(W_0^{[2t]}(x), N_0^{[2t]}(x))$. For a bounded-distance decoding, however, such a kind of output diversity shall not be a concern.

To facilitate understanding the early termination mechanism, we now present several examples.

Example 1: Let $\mathbb{F}_{2^5}$ be generated by the primitive polynomial $p(x) = x^5 + x^2 + 1$. Let $\alpha \in \mathbb{F}_{2^5}$ be a primitive element and $\{v_0, v_1, v_2, v_3, v_4\} = \{1, \alpha, \alpha^2, \alpha^3, \alpha^4\}$ be a basis of $\mathbb{F}_{2^5}$ over $\mathbb{F}_2$. The elements of $\mathbb{F}_{2^5}$ can be represented as $\{\omega_i = \sum_{j=0}^{4} i_j v_j : i = \sum_{j=0}^{4} i_j 2^j, 0 \le i < 32\}$. Consider a $(32, 24)$ RS code over $\mathbb{F}_{2^5}$ as defined by (27). The error correction capability of the code is $t = 4$. Assume that the all-zero codeword is transmitted and the received word is given by $r(x) = \alpha^{19}x^5 + \alpha^{29}x^6 + \alpha^{18}x^{28}$. Therefore, $e = 3$ errors have occurred. The syndromes can be computed as $(S(\omega_0), S(\omega_1), \cdots, S(\omega_7)) = (\alpha^{18}, \alpha^{17}, \alpha^7, \alpha^{16}, \alpha^{10}, \alpha^{28}, \alpha^9, \alpha^{15})$. Thus, the algorithm input, $(x_i, y_i) = (\omega_i, S(\omega_i))$, $0 \le i < 8$, is given by

i	0	1	2	3	4	5	6	7
x_i	0	1	α	α^{18}	α^2	α^5	α^{19}	α^{11}
y_i	α^{18}	α^{17}	α^7	α^{16}	α^{10}	α^{28}	α^9	α^{15}

The operation details of the PWB algorithm are shown in Table I.

Since $\text{rank}_0^{[8]} < \text{rank}_1^{[8]}$, the PWB algorithm outputs $(W_0^{[8]}(x), N_0^{[8]}(x))$, which is equal (up to a scalar) to $(\Lambda(x), Z(x))$. The roots of $W_0^{[8]}(x)$ give the error locators $\alpha^5 = \omega_5$, $\alpha^{19} = \omega_6$, and $\alpha^{13} = \omega_{28}$. According to (31), the corresponding error values can be computed as $e_5 = \alpha^{19}$, $e_6 = \alpha^{29}$, and $e_{28} = \alpha^{18}$ based on $(W_0^{[8]}(x), N_0^{[8]}(x))$.

From Table I, the smallest integer r such that $\text{rank}_1^{[r]} = 2t + 1 = 9$ is equal to 5. Therefore, the EPWB algorithm outputs $(W_0^{[5]}(x), N_0^{[5]}(x))$. The roots of $W_0^{[5]}(x)$ are given by $\alpha^{13} = \omega_{28}$. According to (43), the error value at ω_{28} can be computed as $e_{28} = \alpha^{18}$ based on $(W_0^{[5]}(x), N_0^{[5]}(x))$.

According to Theorem 4-*(i)*, since $e = 3$ and $\text{rank}_0^{[2e]} > \text{rank}_1^{[2e]}$, $W_1^{[2e]}(x) = W_1^{[6]}(x)$ is also an incomplete error locator polynomial. The roots of $W_1^{[6]}(x)$ are given by $\alpha^5 = \omega_5$ and $\alpha^{13} = \omega_{28}$. According to Lemma 3, the error value at ω_{28} can be computed as $e_{28} = \alpha^{18}$ based on $(W_1^{[6]}(x), N_1^{[6]}(x))$.

For this example, the smallest integer r such that $\text{rank}_1^{[r]} = 2t+1 = 9$ is $r = 5 < 2e = 6$ and the incomplete error locator polynomial $W_0^{[5]}(x)$ contains no roots of $\Lambda(x)$ in $\{\omega_i : i < 2^m - k\}$. However, this is not always the case, as can be seen from the following two examples. □

Example 2: Consider the $(32, 24)$ RS code over $\mathbb{F}_{2^5}$ given in Example 1. Assume that the all-zero codeword is transmitted and the received word is given by $r(x) = \alpha^{18}x^4 + \alpha^2x^7 + \alpha^3x^{20}$. Therefore, $e = 3$ errors have occurred. The syndromes can be computed as $(S(\omega_0), S(\omega_1), \cdots, S(\omega_7)) = (\alpha^{16}, \alpha, \alpha^{26}, \alpha^{28}, \alpha^3, \alpha^{13}, \alpha^{24}, 1)$. Thus, the algorithm input, $(x_i, y_i) = (\omega_i, S(\omega_i))$, $0 \le i < 8$, is given by

i	0	1	2	3	4	5	6	7
x_i	0	1	α	α^{18}	α^2	α^5	α^{19}	α^{11}
y_i	α^{16}	α	α^{26}	α^{28}	α^3	α^{13}	α^{24}	1

TABLE II
THE OPERATION DETAILS OF THE PWB ALGORITHM FOR EXAMPLE 2

r	$b_r^{[r]}$	$a_r^{[r]}$	$\mathrm{rank}_0^{[r]}$	$\mathrm{rank}_1^{[r]}$	$\delta^{[r]}$
0	α^{16}	1	0	1	0
1	α^{25}	α	1	2	0
2	0	α^7	2	3	1
3	0	α^9	2	5	1
4	α^{26}	α^{21}	2	7	0
5	α^7	α^4	7	4	0
6	0	α^5	4	9	1
7	α^{22}	α^{13}	4	11	0
8	–	–	11	6	–

r	$W_0^{[r]}(x)$	$N_0^{[r]}(x)$
0	1	0
1	1	α^{16}
2	$\alpha^{25}x+\alpha$	α^{17}
3	$\alpha x+\alpha^8$	α^{24}
4	$\alpha^{10}x+\alpha^{17}$	α^2
5	$\alpha^{26}x^3+\alpha^{27}x+\alpha^{29}$	$\alpha^{11}x^3+\alpha^{22}x+\alpha^{14}$
6	$\alpha^{17}x^2+\alpha^{21}x+\alpha^{26}$	$\alpha^9x+\alpha^{11}$
7	$\alpha^{22}x^2+\alpha^{26}x+1$	$\alpha^{14}x+\alpha^{16}$
8	$\alpha^{17}x^5+\alpha^4x^4+\alpha^{30}x^3+\alpha^{25}x^2+\alpha^{24}x$	$\alpha^2x^5+\alpha^{20}x^4+\alpha^{27}x^3+\alpha^2x^2+\alpha^{12}x$

r	$W_1^{[r]}(x)$	$N_1^{[r]}(x)$
0	0	1
1	x	0
2	$x+1$	$\alpha^{16}x+\alpha^{16}$
3	$x^2+\alpha^{18}x+\alpha$	$\alpha^{16}x^2+\alpha^3x+\alpha^{17}$
4	$x^3+\alpha^{11}x+\alpha^{19}$	$\alpha^{16}x^3+\alpha^{27}x+\alpha^4$
5	$\alpha^{10}x^2+\alpha^{14}x+\alpha^{19}$	$\alpha^2x+\alpha^4$
6	$\alpha^{26}x^4+x^3+\alpha^{27}x^2+\alpha^{27}x+\alpha^3$	$\alpha^{11}x^4+\alpha^{16}x^3+\alpha^{22}x^2+\alpha^{28}x+\alpha^{19}$
7	$\alpha^{26}x^5+\alpha^{13}x^4+\alpha^8x^3+\alpha^7x^2+\alpha^{26}x+\alpha^{22}$	$\alpha^{11}x^5+\alpha^{29}x^4+\alpha^5x^3+\alpha^{11}x^2+\alpha^{14}x+\alpha^7$
8	$\alpha^{22}x^3+\alpha^{17}x^2+\alpha^{27}x+\alpha^{11}$	$\alpha^{14}x^2+\alpha x+\alpha^{27}$

The operation details of the PWB algorithm are shown in Table II.

Since $\mathrm{rank}_0^{[8]} > \mathrm{rank}_1^{[8]}$, the PWB algorithm outputs $(W_1^{[8]}(x), N_1^{[8]}(x))$, which is equal (up to a scalar) to $(\Lambda(x), Z(x))$. The roots of $W_1^{[8]}(x)$ give the error locators $\alpha^2 = \omega_4$, $\alpha^{11} = \omega_7$, and $\alpha^7 = \omega_{20}$. According to (31), the corresponding error values can be computed as $e_4 = \alpha^{18}$, $e_7 = \alpha^2$, and $e_{20} = \alpha^3$ based on $(W_1^{[8]}(x), N_1^{[8]}(x))$.

From Table II, the smallest integer r such that $\mathrm{rank}_1^{[r]} = 2t+1 = 9$ is equal to 6. Therefore, the EPWB algorithm outputs $(W_0^{[6]}(x), N_0^{[6]}(x))$. The roots of $W_0^{[6]}(x)$ are given by $\alpha^2 = \omega_4$ and $\alpha^7 = \omega_{20}$. According to (43), the error value at ω_{20} is $e_{20} = \alpha^3$ based on $(W_0^{[6]}(x), N_0^{[6]}(x))$.

For this example, the smallest integer r such that $\mathrm{rank}_1^{[r]} = 2t+1 = 9$ is $r = 6 = 2e$ and the incomplete error locator polynomial $W_0^{[6]}(x)$ contains partial roots of $\Lambda(x)$ in $\{\omega_i : i < 2^m - k\}$. □

Example 3: Consider the $(32, 24)$ RS code over $\mathbb{F}_{2^5}$ given in Example 1. Assume that the all-zero codeword is transmitted and the received word is given by $r(x) = \alpha^{23}x^{19} + \alpha^{27}x^{25} + \alpha^5x^{30}$. Therefore, $e = 3$ errors have occurred. The syndromes can be computed as $(S(\omega_0), S(\omega_1), \ldots, S(\omega_7)) = (\alpha^{10}, \alpha^{30}, \alpha^{15}, \alpha^{13}, \alpha^{24}, \alpha^4, \alpha^{26}, 0)$. Thus, the algorithm input, $(x_i, y_i) = (\omega_i, S(\omega_i))$, $0 \le i < 8$, is given by

i	0	1	2	3	4	5	6	7
x_i	0	1	α	α^{18}	α^2	α^5	α^{19}	α^{11}
y_i	α^{10}	α^{30}	α^{15}	α^{13}	α^{24}	α^4	α^{26}	0

The operation details of the PWB algorithm are shown in Table III.

Since $\mathrm{rank}_0^{[8]} < \mathrm{rank}_1^{[8]}$, the PWB algorithm outputs $(W_0^{[8]}(x), N_0^{[8]}(x))$, which is equal (up to a scalar) to $(\Lambda(x), Z(x))$. The roots of $W_0^{[8]}(x)$ give the error locators $\alpha^{17} = \omega_{19}$, $\alpha^{25} = \omega_{25}$, and $\alpha^{24} = \omega_{30}$. According to (31), the corresponding error values can be computed as $e_{19} = \alpha^{23}$, $e_{25} = \alpha^{27}$, and $e_{30} = \alpha^5$ based on $(W_0^{[8]}(x), N_0^{[8]}(x))$.

From Table III, the smallest integer r such that $\mathrm{rank}_1^{[r]} = 2t+1 = 9$ is equal to 7. Therefore, the EPWB algorithm outputs $(W_0^{[7]}(x), N_0^{[7]}(x))$. The roots of $W_0^{[7]}(x)$ are $\alpha^{17} = \omega_{19}$, $\alpha^{25} = \omega_{25}$, and $\alpha^{24} = \omega_{30}$. According to (43), the error values at these error locators can be computed as $e_{19} = \alpha^{23}$, $e_{25} = \alpha^{27}$, and $e_{30} = \alpha^5$ based on $(W_0^{[7]}(x), N_0^{[7]}(x))$, the same as computed based on the PWB algorithm.

In this example, all three errors occur in the positions $\{i \ge 2^m - k\}$. The smallest integer r such that $\mathrm{rank}_1^{[r]} = 2t+1$ is equal to $t + e = 7$ and $W_0^{[7]}(x) = \alpha^{30}\Lambda(x)$, which conforms to Theorem 4-*(iv)*. □

From the above examples, we have the following observations: (1) The incomplete error locator polynomial may or may not contain the error locators ω_i with $i < 2^m - k$ as its roots. (2) If some errors occur in the positions $\{i < 2^m - k\}$, then the smallest integer r such that $\mathrm{rank}_1^{[r]} = 2t+1$ may be less than $2e$.

TABLE III
THE OPERATION DETAILS OF THE PWB ALGORITHM FOR EXAMPLE 3

r	$b_r^{[r]}$	$a_r^{[r]}$	$\mathrm{rank}_0^{[r]}$	$\mathrm{rank}_1^{[r]}$	$\delta^{[r]}$
0	α^{10}	1	0	1	0
1	α^{18}	α^{30}	1	2	0
2	α^{23}	α^{30}	2	3	0
3	α^{16}	α^{26}	3	4	0
4	α^{7}	α^{20}	4	5	0
5	α^{20}	α^{30}	5	6	0
6	0	α^{7}	6	7	1
7	0	α^{9}	6	9	1
8	–	–	6	11	–

r	$W_0^{[r]}(x)$	$N_0^{[r]}(x)$
0	1	0
1	1	α^{10}
2	$\alpha^{18}x+\alpha^{30}$	α^{9}
3	$\alpha^{13}x+\alpha^{19}$	$\alpha^{2}x+\alpha^{29}$
4	$\alpha^{3}x^2+\alpha x+\alpha^{19}$	$\alpha^{23}x+\alpha^{29}$
5	$\alpha^{18}x^2+\alpha^{16}x+\alpha^{10}$	$\alpha^{9}x^2+\alpha^{20}$
6	$\alpha^{23}x^3+\alpha^{30}x^2+\alpha^{14}x+\alpha^{27}$	$\alpha^{18}x^2+\alpha^{24}x+\alpha^{6}$
7	$\alpha^{30}x^3+\alpha^{6}x^2+\alpha^{21}x+\alpha^{3}$	$\alpha^{25}x^2+x+\alpha^{13}$
8	$\alpha^{8}x^3+\alpha^{15}x^2+\alpha^{30}x+\alpha^{12}$	$\alpha^{3}x^2+\alpha^{9}x+\alpha^{22}$

r	$W_1^{[r]}(x)$	$N_1^{[r]}(x)$
0	0	1
1	x	0
2	$x+1$	$\alpha^{10}x+\alpha^{10}$
3	$\alpha^{18}x^2+\alpha^{7}x+1$	$\alpha^{9}x+\alpha^{10}$
4	$\alpha^{13}x^2+\alpha^{11}x+\alpha^{6}$	$\alpha^{2}x^2+\alpha^{5}x+\alpha^{16}$
5	$\alpha^{3}x^3+\alpha^{11}x^2+\alpha^{12}x+\alpha^{21}$	$\alpha^{23}x^2+\alpha^{4}x+1$
6	$\alpha^{18}x^3+\alpha^{7}x^2+\alpha^{29}x+\alpha^{15}$	$\alpha^{9}x^3+\alpha^{14}x^2+\alpha^{20}x+\alpha^{25}$
7	$\alpha^{18}x^4+\alpha^{24}x^3+\alpha^{24}x^2+\alpha^{20}x+\alpha^{3}$	$\alpha^{9}x^4+\alpha^{27}x^3+\alpha^{3}x^2+\alpha^{7}x+\alpha^{13}$
8	$\alpha^{18}x^5+\alpha^{26}x^4+\alpha^{12}x^3+\alpha^{13}x^2+\alpha^{29}x+\alpha^{14}$	$\alpha^{9}x^5+\alpha^{11}x^4+\alpha^{13}x^3+\alpha^{29}x^2+\alpha^{15}x+\alpha^{24}$

V. Frequency-domain Algorithms and Architectures

A. FPWB algorithm and FEPWB algorithm

We first adapt the PWB algorithm to the decoding of RS codes by letting $x_i = \omega_i$ and $y_i = S(\omega_i)$. Unlike the PWB algorithm that updates the four polynomials $W_0^{[r+1]}(x)$, $N_0^{[r+1]}(x)$, $W_1^{[r+1]}(x)$, and $N_1^{[r+1]}(x)$, we consider to update two of them (refer to Line 7 of Algorithm 2),

$$\begin{pmatrix} W_0^{[r+1]}(x) \\ W_1^{[r+1]}(x) \end{pmatrix} = \begin{pmatrix} -a_r & b_r \\ (x-\omega_r)(1-\delta^{[r]}) & (x-\omega_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} W_0^{[r]}(x) \\ W_1^{[r]}(x) \end{pmatrix}. \tag{53}$$

Letting $x = \omega_i$ for $0 \le i \le t$, we obtain

$$\begin{pmatrix} W_0^{[r+1]}(\omega_i) \\ W_1^{[r+1]}(\omega_i) \end{pmatrix} = \begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (\omega_i-\omega_r)(1-\delta^{[r]}) & (\omega_i-\omega_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} W_0^{[r]}(\omega_i) \\ W_1^{[r]}(\omega_i) \end{pmatrix}. \tag{54}$$

Based on the above discussion, we now present the frequency-domain PWB (FPWB) algorithm in Algorithm 4. Unlike the PWB algorithm that returns a polynomial pair $(W(x), N(x))$, the FPWB algorithm returns the evaluation of $W(x)$ at $t+1$ points $\{\omega_i\}_{i=0}^{t}$, i.e., $(W(\omega_0), W(\omega_1), \dots, W(\omega_t))$. Note that if $e \le t$, the error locator polynomial $\Lambda(x)$ is such that $\deg(\Lambda(x)) \le t$. Consequently, we have

$$(\Lambda(\omega_0), \Lambda(\omega_1), \dots, \Lambda(\omega_t)) = (W(\omega_0), W(\omega_1), \dots, W(\omega_t)). \tag{55}$$

Once we have obtained $(\Lambda(\omega_0), \Lambda(\omega_1), \dots, \Lambda(\omega_t))$ by the FPWB algorithm, based on (30), we can compute

$$\begin{aligned}&(Z(\omega_0), Z(\omega_1), \dots, Z(\omega_{t-1})) \\ &= (\Lambda(\omega_0)S(\omega_0), \Lambda(\omega_1)S(\omega_1), \dots, \Lambda(\omega_{t-1})S(\omega_{t-1})).\end{aligned} \tag{56}$$

If $\deg(\Lambda(x)) \le t$ and $\deg(Z(x)) < t$, $\Lambda(x)$ and $Z(x)$ can be computed based on $(\Lambda(\omega_0), \Lambda(\omega_1), \dots, \Lambda(\omega_t))$ and $(Z(\omega_0), Z(\omega_1), \dots, Z(\omega_{t-1}))$ by polynomial interpolation or more efficiently by LCH-FFT [9].

We know from Section IV that for a shortened RS code as given in Fig. 1-b), we have $\Lambda_1(x) = \Lambda(x)$ and $Z_1(x) = Z(x)$, where $\Lambda_1(x)$ is an incomplete error locator polynomial and $Z_1(x)$ is the associated error evaluator polynomial. Based on the early-terminating PWB (EPWB) algorithm in Algorithm 3, we present the frequency-domain EPWB (FEPWB) algorithm in Algorithm 5. Note that the FEPWB algorithm applies only to the shortened RS code in Fig. 2-b) such that $n_s \ge 2^m - k$. For other RS codes in Fig. 2, we cannot compute $Z_1(x)$ as the FPWB algorithm that uses (56). If $e \le t$, the FEPWB

Algorithm 4: The frequency-domain PWB (FPWB) algorithm

```
Input: $(\omega_i, S(\omega_i)), 0 \le i \le 2t-1$.
Output: $(\Lambda(\omega_0), \Lambda(\omega_1), \dots, \Lambda(\omega_t))$
1: Initialization:
   $\begin{pmatrix} W_0^{[0]}(\omega_i) \\ W_1^{[0]}(\omega_i) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, 0 \le i \le t; \begin{pmatrix} b_i^{[0]} \\ a_i^{[0]} \end{pmatrix} = \begin{pmatrix} -y_i \\ 1 \end{pmatrix}, 0 \le i \le 2t-1; \begin{pmatrix} \mathrm{rank}_0^{[0]} \\ \mathrm{rank}_1^{[0]} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
2: for $r = 0, 1, \dots, 2t-1$ do
3:     Let $\delta^{[r]} = \big((\mathrm{rank}_0^{[r]} < \mathrm{rank}_1^{[r]}) \,\&\&\, (b_r^{[r]} = 0)\big) \,||\, \big((\mathrm{rank}_0^{[r]} > \mathrm{rank}_1^{[r]}) \,\&\&\, (a_r^{[r]} \ne 0)\big)$
4:     for  $i = 0, 1, \dots, 2t-1$ do
5:         $\begin{pmatrix} b_i^{[r+1]} \\ a_i^{[r+1]} \end{pmatrix} = \begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (\omega_i - \omega_r)(1-\delta^{[r]}) & (\omega_i - \omega_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} b_i^{[r]} \\ a_i^{[r]} \end{pmatrix}$
6:     end for;
7:     for $i = 0, 1, \dots, t$ do
8:         $\begin{pmatrix} W_0^{[r+1]}(\omega_i) \\ W_1^{[r+1]}(\omega_i) \end{pmatrix} = \begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (\omega_i - \omega_r)(1-\delta^{[r]}) & (\omega_i - \omega_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} W_0^{[r]}(\omega_i) \\ W_1^{[r]}(\omega_i) \end{pmatrix}$
9:     end for;
10:    if $\delta^{[r]} = 1$ then
11:        $\begin{pmatrix} \mathrm{rank}_0^{[r+1]} \\ \mathrm{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \mathrm{rank}_0^{[r]} \\ \mathrm{rank}_1^{[r]} + 2 \end{pmatrix}$
12:    else
13:        $\begin{pmatrix} \mathrm{rank}_0^{[r+1]} \\ \mathrm{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \mathrm{rank}_1^{[r]} \\ \mathrm{rank}_0^{[r]} + 2 \end{pmatrix}$
14:    end if
15:    if $\mathrm{rank}_0^{[2t]} < \mathrm{rank}_1^{[2t]}$ then
16:        return $\big(W_0^{[2t]}(\omega_0), W_0^{[2t]}(\omega_1), \dots, W_0^{[2t]}(\omega_t)\big)$
17:    else
18:        return $\big(W_1^{[2t]}(\omega_0), W_1^{[2t]}(\omega_1), \dots, W_1^{[2t]}(\omega_t)\big)$
19:    end if
20: end for
```

algorithm outputs

$$\begin{aligned}&(W_0^{[r_1]}(\omega_0), W_0^{[r_1]}(\omega_1), \dots, W_0^{[r_1]}(\omega_t)) \\ &= (\Lambda_1(\omega_0), \Lambda_1(\omega_1), \dots, \Lambda_1(\omega_t)).\end{aligned} \tag{57}$$

However, the errors may occur in the positions $\{i : 0 \le i < 2^m - k\}$, in which case we have $\Lambda_1(x) \ne \Lambda(x)$ and $Z_1(x) \ne Z(x)$, which implies that

$$\begin{aligned}&(Z_1(\omega_0), Z_1(\omega_1), \dots, Z_1(\omega_{t-1})) \\ &\ne (\Lambda_1(\omega_0)S(\omega_0), \Lambda_1(\omega_1)S(\omega_1), \dots, \Lambda_1(\omega_{t-1})S(\omega_{t-1})).\end{aligned} \tag{58}$$

Consequently, we have no way to compute $Z_1(x)$.

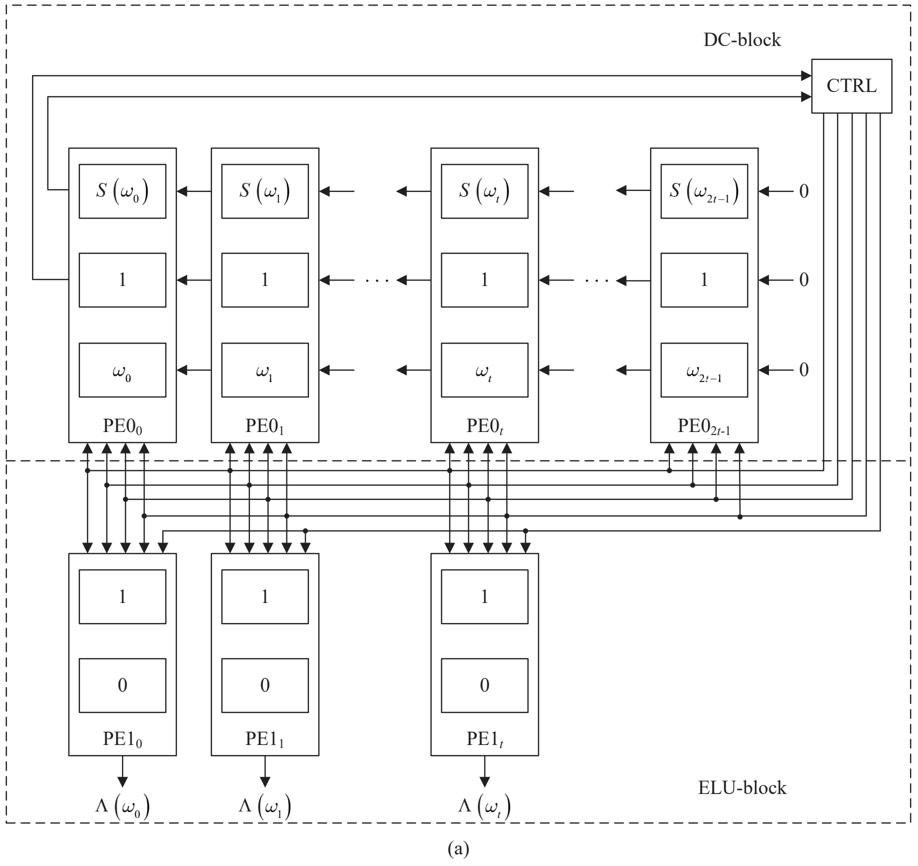
Algorithm 5: The frequency-domain EPWB (FEPWB) algorithm

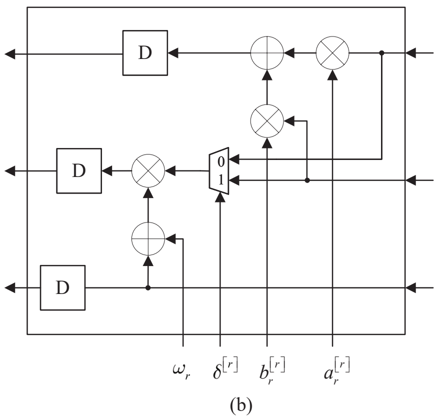
```
Input: $(\omega_i, S(\omega_i)), 0 \le i \le 2t-1$.
Output: $(\Lambda_1(\omega_0), \Lambda_1(\omega_1), \dots, \Lambda_1(\omega_t))$
1: Initialization:
   $\begin{pmatrix} W_0^{[0]}(\omega_i) \\ W_1^{[0]}(\omega_i) \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, 0 \le i \le t; \begin{pmatrix} b_i^{[0]} \\ a_i^{[0]} \end{pmatrix} = \begin{pmatrix} -y_i \\ 1 \end{pmatrix}, 0 \le i \le 2t-1; \begin{pmatrix} \mathrm{rank}_0^{[0]} \\ \mathrm{rank}_1^{[0]} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
2: for $r = 0, 1, \dots, 2t-1$ do
3:     Let $\delta^{[r]} = \big((\mathrm{rank}_0^{[r]} < \mathrm{rank}_1^{[r]}) \,\&\&\, (b_r^{[r]} = 0)\big) \,||\, \big((\mathrm{rank}_0^{[r]} > \mathrm{rank}_1^{[r]}) \,\&\&\, (a_r^{[r]} \ne 0)\big)$
4:     for $i = 0, 1, \dots, 2t-1$ do
5:         $\begin{pmatrix} b_i^{[r+1]} \\ a_i^{[r+1]} \end{pmatrix} = \begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (\omega_i - \omega_r)(1-\delta^{[r]}) & (\omega_i - \omega_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} b_i^{[r]} \\ a_i^{[r]} \end{pmatrix}$
6:     end for;
7:     for $i = 0, 1, \dots, t$ do
8:         $\begin{pmatrix} W_0^{[r+1]}(\omega_i) \\ W_1^{[r+1]}(\omega_i) \end{pmatrix} = \begin{pmatrix} -a_r^{[r]} & b_r^{[r]} \\ (\omega_i - \omega_r)(1-\delta^{[r]}) & (\omega_i - \omega_r)\delta^{[r]} \end{pmatrix} \begin{pmatrix} W_0^{[r]}(\omega_i) \\ W_1^{[r]}(\omega_i) \end{pmatrix}$
9:     end for;
10:    if $\delta^{[r]} = 1$ then
11:        $\begin{pmatrix} \mathrm{rank}_0^{[r+1]} \\ \mathrm{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \mathrm{rank}_0^{[r]} \\ \mathrm{rank}_1^{[r]} + 2 \end{pmatrix}$
12:    else
13:        $\begin{pmatrix} \mathrm{rank0}^{[r+1]} \\ \mathrm{rank}_1^{[r+1]} \end{pmatrix} = \begin{pmatrix} \mathrm{rank}_1^{[r]} \\ \mathrm{rank}_0^{[r]} + 2 \end{pmatrix}$
14:    end if
15:    if $((\mathrm{rank}_1^{[r+1]} = 2t+1) \,||\, (r = 2t-1))$ then
16:        return $\big(W_0^{[r+1]}(\omega_0), W_0^{[r+1]}(\omega_1), \dots, W_0^{[r+1]}(\omega_t)\big)$
17:    end if
18: end for
```

B. Architectures

In this paper, when we speak of an architecture designed based on a certain algorithm, we use the name of the algorithm to designate the architecture. Based on the description of the FPWB algorithm in Algorithm 4, we present a systolic architecture for the algorithm in Fig. 2. A systolic architecture refers to a network of processor elements (PEs) that rhythmically compute and pass data through the system [21]. It derived its name from drawing an analogy to how blood rhythmically flows through a biological heart as the data flows from memory in a rhythmic fashion passing through many elements before it returns to memory.

The FPWB architecture consists of two main blocks: discrepancy computation (DC) block and error locator update (ELU) block, as shown in the upper and lower part of Fig. 2-a),





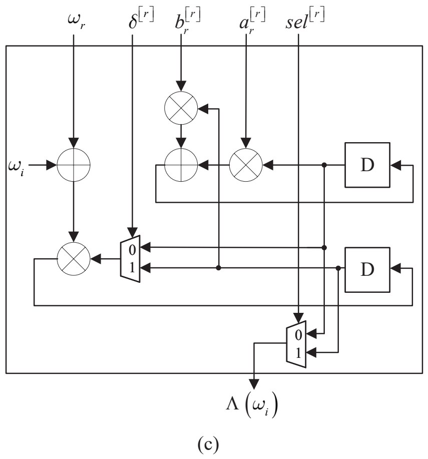


Fig. 2. The FPWB architecture. (a) The systolic architecture. (b) The structure for PE0. (c) The structure for PE1.

respectively. The DC block is responsible for updating $a_i^{[r]}$ and $b_i^{[r]}$ for $0 \le i \le 2t-1$. It is an array of $2t$ PE0s, arranged as $\text{PE}0_0, \text{PE}0_1, \dots, \text{PE}0_{2t-1}$ in sequence. The structure of PE0$_i$ is shown in Fig. 2-b), which accomplishes the following two assignment operations (refer to Line 5 of Algorithm 4),

$$b_i^{[r+1]} = b_r^{[r]} a_i^{[r]} - a_r^{[r]} b_i^{[r]}, \tag{59}$$

and

$$a_i^{[r+1]} = \begin{cases} (\omega_i - \omega_r) b_i^{[r]}, & \text{if } \delta^{[r]} = 0; \\ (\omega_i - \omega_r) a_i^{[r]}, & \text{if } \delta^{[r]} = 1. \end{cases} \tag{60}$$

Each PE0$_i$ consists of two adders, three multipliers, three latches, and one multiplexer. The ELU block is responsible for updating $W_0^{[r]}(\omega_i)$ and $W_1^{[r]}(\omega_i)$ for $0 \le i \le t$. It is an array of $t+1$ PE1s, arranged as $\text{PE}1_0, \text{PE}1_1, \dots, \text{PE}1_t$ in sequence. The structure of PE1$_i$ is shown in Fig. 2-c), which accomplishes the following two assignment operations (refer to Line 8 of Algorithm 4),

$$W_0^{[r+1]}(\omega_i) = b_r^{[r]} W_1^{[r]}(\omega_i) - a_r^{[r]} W_0^{[r]}(\omega_i), \tag{61}$$

and

$$W_1^{[r+1]}(\omega_i) = \begin{cases} (\omega_i - \omega_r) W_0^{[r]}(\omega_i), & \text{if } \delta^{[r]} = 0; \\ (\omega_i - \omega_r) W_1^{[r]}(\omega_i), & \text{if } \delta^{[r]} = 1. \end{cases} \tag{62}$$

Each PE1$_i$ consists of two adders, three multipliers, two latches, and two multiplexers.

TABLE IV
COMPARISONS OF VARIOUS ARCHITECTURES

Architectures	Adders	Multipliers	Latches	Muxes	Clocks	$T_{\text{crit-path}}$
RiBM [12]	$3t+1$	$6t+2$	$6t+2$	$3t+1$	$2t$	$T_{\text{mult}}+T_{\text{add}}$
ePIBMA [13]	$2t+1$	$4t+2$	$4t+2$	$6t+3$	$2t$	$T_{\text{mult}}+T_{\text{add}}$
FPWB	$6t+2$	$9t+3$	$8t+2$	$4t+2$	$2t$	$T_{\text{mult}}+T_{\text{add}}$
FEPWB	$6t+2$	$9t+3$	$8t+2$	$3t+1$	$t+e$	$T_{\text{mult}}+T_{\text{add}}$

It should be noted that $(b_i^{[r]}, a_i^{[r]})$ and $(W_0^{[r]}(\omega_i), W_1^{[r]}(\omega_i))$ have the same update rule. Here, we use a heterogeneous design, using two different PE structures separately updating them. This is based on the following considerations. To single out $b_r^{[r]}$ and $a_r^{[r]}$ as broadcast signals, the updated $b_i^{[r]}$ and $a_i^{[r]}$ are shifted leftwards so that $b_r^{[r]}$ and $a_r^{[r]}$ can always be fetched out from PE0_0 that is the leftmost PE0. An advantage of the design is that the critical path can be improved because the circuitry for selecting $b_r^{[r]}$ and $a_r^{[r]}$ can be eliminated in the critical path. On the other hand, the updated $W_0^{[r]}(\omega_i)$ and $W_1^{[r]}(\omega_i)$ are not shifted leftwards but left in the original place. This is to save power consumption. Moreover, if the early termination is adopted, as discussed below, the algorithm output can be fetched out from fixed positions.

The FEPWB algorithm differs from the FPWB algorithm in the early termination. The algorithm difference has a slight impact on the architecture design. The control unit needs to be modified to signal the early termination. Besides, one multiplexer in PE0_i in Fig. 2-c) shall be removed because the output of the FEPWB algorithm is not selective as compared with the FPWB algorithm. In light of the slight difference between the FPWB and the FEPWB architectures, it is easy to develop a scalable architecture that allows to switch between the functions of the two architectures.

Table IV compares various architectures, including RiBM [12], ePIBMA [13], FPWB, and FEPWB, in terms of resource, clock, and critical path. The RiBM and ePIBMA architectures were designed by reformulating the BM algorithm. Clearly, all the four architectures have the same critical path, i.e., one adder and one multiplier. The proposed FPWB and FEPWB architectures represent the most efficient architectures for the WB algorithm by far. For high-speed parallel implementation, although FPWB and FEPWB have higher complexity than RiBM and ePIBMA, the overall implementation complexity of the FFT-based decoding using FEPWB or FEPWB is expected to be considerably lower than that of the conventional decoding using RiBM or ePIBMA due to significant complexity advantage in the syndrome computation and the Chien search by using LCH-FFT. See [9] for detailed examples.

VI. CONCLUSION

In this paper, we present four new variants of the WB algorithm, namely, parallel WB (PWB) algorithm, early-terminating PWB (EPWB) algorithm, frequency-domain PWB (FPWB) algorithm, and frequency-domain EPWB (FEPWB) algorithm. The concept of incomplete error locator polynomial is introduced to show that the PWB algorithm can be terminated early. The similarities and differences between the WB algorithm and the BM algorithm are compared in terms of early-termination mechanism. A systolic architecture for the FPWB algorithm is developed, which can be easily adapted to the FEPWB algorithm. This provides an efficient key equation solver for the FFT-based RS decoding [7].

ACKNOWLEDGMENT

The authors would like to thank the associate editor and the anonymous reviewers for their excellent comments and suggestions that helped us improve this paper.

REFERENCES

[1] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[2] L. R. Welch and E. R. Berlekamp, "Error correction for algebraic block codes," U.S. Patent 4,633,470, Dec. 1986.

[3] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Hoboken, NJ: John Wiley & Sons, Inc., 2005.

[4] V. Guruswami, A. Rudra, and M. Sudan, *Essential Coding Theory*. Draft available at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/.

[5] S.-J. Lin, W.-H. Chung, and Y. S. Han, "Novel polynomial basis and its application to Reed–Solomon erasure codes," in *Proc. IEEE 55th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2014, pp. 316–325.

[6] S.-J. Lin, T. Y. Al-Naffouri, Y. S. Han, and W.-H. Chung, "Novel polynomial basis with fast Fourier transform and its application to Reed–Solomon erasure codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 11, pp. 6284–6299, Nov. 2016.

[7] S.-J. Lin, T. Y. Al-Naffouri, and Y. S. Han, "FFT algorithm for binary extension finite fields and its application to Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 62, no. 10, pp. 5343–5358, Oct. 2016.

[8] N. Tang and Y. Lin, "Fast encoding and decoding algorithms for arbitrary (n, k) Reed–Solomon codes over $\mathbb{F}_{2^m}$," *IEEE Commun. Lett.*, vol. 24, no. 4, pp. 716–719, Apr. 2020.

[9] N. Tang and Y. S. Han, "A new decoding method for Reed–Solomon codes based on FFT and modular approach," *IEEE Trans. Commun.*, vol. 70, no. 12, pp. 7790–7801, Oct. 2022.

[10] N. Tang and Y. S. Han, "New decoding of Reed–Solomon codes based on FFT and modular approach," [Online]. Available: https://arxiv.org/abs/2207.11079 (The arxiv version provides more details than [9])

[11] D. Dabiri and I. F. Blake, "Fast parallel algorithms for decoding Reed-Solomon codes based on remainder polynomials, " *IEEE Trans. Inf. Theory*, vol. 41, no. 4, pp. 873–885, Apr. 1995.

[12] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed–Solomon decoders," *IEEE Trans. VLSI Syst.*, vol. 9, no. 5, pp. 641–655, Oct. 2001.

[13] Y. Wu, "New scalable decoder architectures for Reed–Solomon codes," *IEEE Trans. Commun.*, vol. 63, no. 8, pp. 2741–2761, Aug. 2015.

[14] K.-K. Tzeng, C. R. P. Hartmann, and R.-T. Chien, "Some notes on iterative decoding," in *Proc. 9th Annu. Allerton Conf. Circuit and System Theory*, 1971, pp. 689–695.

[15] C.-L. Chen, "High-speed decoding of BCH codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 2, pp. 254–256, Mar. 1981.

[16] K. Imamura and W. Yoshida, "A simple derivation of the Berlekamp–Massey algorithm and some applications," *IEEE Trans. Inf. Theory*, vol. 33, no. 1, pp. 146–150, Jan. 1987.

[17] C.-W. Liu and C.-C. Lu, "A view of Gaussian elimination applied to early-stopped Berlekamp–Massey algorithm," *IEEE Trans. Commun.*, vol. 55, no. 6, pp. 1131–1143, Jun. 2007.

[18] X. Ma and X.-M. Wang, "On the minimal interpolation problem and decoding RS codes," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1573–1580, Jul. 2000.

[19] C. Chen, N. Tang, Y. S. Han, and B. Bai, “On the equivalence between the Welch–Berlekamp algorithm and the modular approach algorithm,” in *2023 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, Sept. 2023, pp. 1–5.

[20] C. Chen, Y. S. Han, N. Tang, S.-J. Lin, B. Bai, and X. Ma, “An early-termination method for the Welch–Berlekamp algorithm,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Taipei, Taiwan, Jun. 2023, pp. 815–819.

[21] H. T. Kung, “Why systolic architectures?”, *Computer*, vol. 15, no. 1, pp. 37–46, Jan. 1982.

Chao Chen received the Ph.D. degree in communication and information systems from Xidian University, China, in 2010. From 2010 to 2014, he was an Engineer with China Academy of Space Technology (CAST), Xi’an. From 2014 to 2015, he was a Postdoctoral Fellow with Institute of Network Coding (INC), The Chinese University of Hong Kong. He is currently an associate Professor with the State Key Laboratory of Integrated Services Networks (ISN), Xidian University, China. His research interests include channel coding and source coding.

Yunghsiang S. Han (Fellow, IEEE) was born in Taipei, Taiwan, in 1962. He received the B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respec tively, and the Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, USA, in 1993. From 1986 to 1988, he was a Lecturer at the Ming-Hsin Engineering College, Hsinchu. He was a Teaching Assistant from 1989 to 1992, and a Research Associate with the School of Computer and Information Science, Syracuse University, from 1992 to 1993. From 1993 to 1997, he was an Associate Professor with the Department of Electronic Engineering, Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering, National Chi Nan University, Nantou, Taiwan, from 1997 to 2004. He was promoted to a Professor in 1998. He was a Visiting Scholar with the Department of Electrical Engineering, University of Hawaii at Manoa, Honolulu, HI, USA, from June 2001 to October 2001; the SUPRIA Visiting Research Scholar with the Department of Electrical Engineering and Computer Science and the CASECenter, Syracuse University, from September 2002 to January 2004 and from July 2012 to June 2013; and a Visiting Scholar with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA, from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering, National Taipei University, Taipei, from August 2004 to July 2010. From August 2010 to January 2017, he was a Chair Professor with the Department of Electrical Engineering, National Taiwan University of Science and Technology. He has been a Chair Professor at the National Taipei University since February 2015. From February 2017 to February 2021, he was with the School of Electrical Engineering and Intelligentization, Dongguan University of Technology, China. He is currently with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China. His research interests include error-control coding, wireless networks, and security. He was a Winner of the 1994 Syracuse University Doctoral Prize. One of his papers won the prestigious 2013 ACM CCS Test-of-Time Award in Cybersecurity.

Nianqi Tang received the Ph.D. degree in communication and information systems from Xidian University, China, in 2019. From 2019 to 2023, he was a Senior Engineer with Huawei Technologies Co., Ltd. Since 2023, he has been an assistant professor with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China. His research interests include error control coding, network coding, and information theory.

Xiao Ma (Member, IEEE) received the Ph.D. degree in communication and information systems from Xidian University, China, in 2000. From 2000 to 2002, he was a Post-Doctoral Fellow with Harvard University, Cambridge, MA, USA. From 2002 to 2004, he was a Research Fellow with the City University of Hong Kong. He is currently a Professor with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. His research interests include information theory, channel coding theory and their applications to communication systems, and digital recording systems.

Baoming Bai (Senior Member, IEEE) received the B.S. degree from the Northwest Telecommunications Engineering Institute, China, in 1987, and the M.S. and Ph.D. degrees in communication engineering from Xidian University, China, in 1990 and 2000, respectively. From 2000 to 2003, he was a Senior Research Assistant at the Department of Electronic Engineering, City University of Hong Kong. Since April 2003, he has been with the State Key Laboratory of Integrated Services Networks (ISN), School of Telecom munication Engineering, Xidian University, China, where he is currently a Professor. In 2005, he was with the University of California, Davis, CA, USA, as a Visiting Scholar. In 2018, he spent one month as a Senior Visiting Fellow at McMaster University, Ontario, Canada. Dr. Bai co-authored the book Channel Coding for 5G (in Chinese, 2020). His research interests include information theory and channel coding, wireless communication, and quantum communication. He received the Best Paper Award from the CIC/IEEE China Communications, in 2018.