A 2.56 Gb/s Soft RS (255, 239) Decoder Chip for Optical Communication Systems

Yi-Min Lin, Chih-Hsiang Hsu, Hsie-Chia Chang, and Chen-Yi Lee, *Member, IEEE*

***Abstract*—Due to high transmission rate requirement for optical communication systems, the growing uncertainty of received signals results in the limited transmission distance. In this paper, a *decision-confined* soft RS decoder chip is proposed to enhance the error correcting performance with area-efficient architectures. Instead of generating numerous possible candidate codewords and determining the most likely one as output codeword, our approach produces only one codeword by confining the degree of error location polynomial. Therefore, hardware complexity is significantly reduced by eliminating decision making unit. Moreover, an iteration-reduced RiBM algorithm is provided to enlarge the coding gain by using more least reliable positions (LRPs) in the limited operation latency. According to simulation results, our proposed soft RS (255, 239; 8) decoder with 5 LRPs outperforms 0.4 dB at 10^{-4} codeword error rate (CER) as compared to hard RS decoders. Implemented in standard CMOS 90 nm technology, the soft decoder chip can achieve 2.56 Gb/s throughput with similar complexity as a hard decoder. It can fit well for 10–40 Gb/s with 16 RS decoders in optical fiber systems and 2.5 Gb/s GPON applications.**

***Index Terms*—Error correction code, optical communication, Reed-Solomon (RS) code.**

I. Introduction

REED-SOLOMON (RS) code, a famous linear block code, is widely utilized in various digital data storage devices and communication systems. Due to the advantage of overcoming the burst errors, International Telecommunication Union (ITU-T) standardizes RS (255, 239; 8) in the high speed optical fiber system [1] and Gigabit Passive Optical Network (GPON) [2], which demand 2.5 Gb/s throughput for achieving 10–40 Gb/s with 16 RS decoders and satisfying the maximum up and down link requirement respectively.

Fig. 1 shows the optical fiber system developed by ITU-T. RS decoders can provide significant gains to resist transmission degradation; however, the higher transmission throughput requirement in optical communication systems will damage the reliability of transmitted data. Forward error correction (FEC) devices need to enhance error correcting capability to overcome increasing noise for supporting longer transmission distance. Without decreasing data rate or changing original system architecture, soft RS decoding algorithms can achieve considerable performance gain by making use of soft information [3]–[7]. Nevertheless, the complex computations of soft decoding algorithms is the bottleneck for the hardware implementation. With VLSI architectures development in these years, the soft decoder still have several times hardware complexity in contrast to hard decoders [8]–[15].

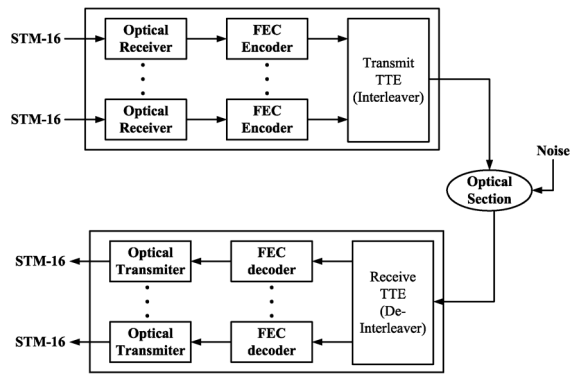

Fig. 1. Block diagram of submarine system, ITU-G.975.

In this paper, a decision-confined soft decoder chip is proposed to enhance error correcting capability while maintaining area efficiency. In Chase algorithm, the final step is exploiting a decision making unit to determine the decoded codeword from a list of candidate codewords. The Euclidean distance between the received data and each codeword is demanded to be calculated. For the long length codeword; however, the Euclidean distance calculation requires large number of multiplication calculations, which are too complex for practical implementation. Our design, instead, only decodes one candidate codeword with the degree of error location polynomial $\Lambda(x)$ less than error correcting capability t. Therefore, the decision making unit and the memories for storing each candidate codeword are no longer demanded, leading to significant hardware complexity reduction. In addition, with advantages of half computation latency, an iteration-reduced reformulated inversionless Berlekamp Massey (IR-RiBM) algorithm and its homogeneous architecture are provided to achieve better performance gain by using more LRPs in the same decoding period. With Gray code based bit-flipping method, the syndrome updater procedure becomes much simpler without using several suit hardware. Moreover, based on Björck-Pereyra (BP) method [16], error values can be computed according to the relation between the syndrome values and the error locators without calculating error evaluator polynomial $\Omega(x)$, resulting in further complexity reduction.

Manuscript received April 08, 2012; revised September 06, 2012 and October 18, 2013; accepted November 30, 2013. Date of publication February 04, 2014; date of current version June 24, 2014. This work was supported by National Science Council and Ministry of Economic Affairs of Taiwan, R.O.C., under grants NSC 101-2628-E-009-013-MY3 and NSC 102-2220-E-009-044, respectively. This paper was recommended by Associate Editor M. Alioto.

Y.-M. Lin is with the SK Hynix Memory Solutions, San Jose, CA, USA. He is also with the Department of Electronics Engineering & Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, R.O.C. (e-mail: ymlin.mail@gmail.com; ymlin@si2lab.org).

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

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2014.2298282

This paper is organized as follows. Section II describes the overview of conventional hard RS codes following with soft decoding methods. Section III and IV characterize decision-confined and iteration-reduced RiBM algorithms respectively. The proposed low complexity VLSI architectures as well as the implementation results of our proposed soft RS decoder are presented in Section V and VI. Finally, we conclude this paper in Section VII.

II. Background

RS code is a kind of cyclic codes and can be classified into non-binary BCH codes [17]. An $(N, K; t)$ RS code over $GF(2^m)$ has block length of N symbols and information length of K symbols, where a symbol consists of m-bit. It has the error correcting capability $t = \lfloor (N-K)/2 \rfloor$. A conventional decoding procedure of RS codes contains four major steps: *syndrome calculator*, *key equation solver*, *Chien search*, and *error value evaluator* as shown in Fig. 2(a). The received polynomial $R(x)$ is fed into syndrome calculator to generate syndrome polynomial $S(x) = S_1 + S_2 x^1 + \cdots + S_{2t} x^{2t-1}$, which is expressed as

$$S_i = R(\alpha^i) = \sum_{j=1}^{v} e_j (\alpha^i)^{l_j} = \sum_{j=1}^{v} e_j (X_j)^i \quad \text{for } i = 1 \sim 2\text{t}, \tag{1}$$

where α is the primitive element over $GF(2^m)$ and v is the number of actual errors. Notice that e_j and l_j are the j-th actual error value and error location respectively and $X_j = \alpha^{l_j}$ indicates the corresponding error locator. The key equation solver computes error location polynomial $\Lambda(x)$ and error evaluator polynomial $\Omega(x)$ according to the key equation

$$S(x)\Lambda(x) = \Omega(x) \bmod x^{2t}, \tag{2}$$

where $\Lambda(x)$ is defined as

$$\Lambda(x) = \prod_{j=1}^{v} (1 - X_j x) = \Lambda_0 + \Lambda_1 x + \cdots + \Lambda_v x^v. \tag{3}$$

The most popular methods for solving the key equation are Berlekamp-Massey (BM) [18] and modified Euclidean algorithms [19]. Once $\Lambda(x)$ and $\Omega(x)$ are evaluated, the Chien search is applied to find the roots of $\Lambda(x)$ [20]. If α^{-l_j}, the inverse of error locator X_j, is a root of $\Lambda(x)$, an error is assumed at the l_j-th position. The error value evaluator is utilized to calculate error values based on Forney's algorithm [21]:

$$e_j = \frac{\Omega\left(X_j^{-1}\right)}{\Lambda'\left(X_j^{-1}\right)}. \tag{4}$$

Notice that $\Lambda'(x) = \Lambda_1 + \Lambda_3 x^2 + \cdots$ denotes the formal derivative of $\Lambda(x)$. Finally, the estimated codeword polynomial $\hat{C}(x)$ is obtained by outputting $R(x)$ from the FIFO and adding those error values at error locations.

To provide higher error correcting performance with the same code rate, soft decoding algorithms of error control codes exploit reliability information from channel for correctting an error number larger than half the minimum distance of a code. Many researchers have been interested in soft decoding methods for BCH and RS codes [3]–[7]. In 1966, Forney developed generalized-minimum-distance (GMD) algorithm [3], which uses soft information and generates candidate sequences for several hard decoders to form a list of candidate codewords. Based on the block diagram of GMD decoding shown in Fig. 2(b), a GMD decoder uses a reliability evaluator to choose the LRPs and sends them into a candidate sequence generator to create several candidate sequences for hard decoders. Each hard decoder produces a candidate codeword to form a candidate codeword list, and a decision making unit determines the most probable one as the output codeword which has the smallest Euclidean distance away from soft received sequence $R(x)$. Since several hard decoders are utilized, a GMD decoder is many times the hardware complexity of a hard decoder. With similar concept, Chase and Chase-GMD algorithms [4], [5] are also widely used to efficiently generate the candidate list and have been applied in many applications.

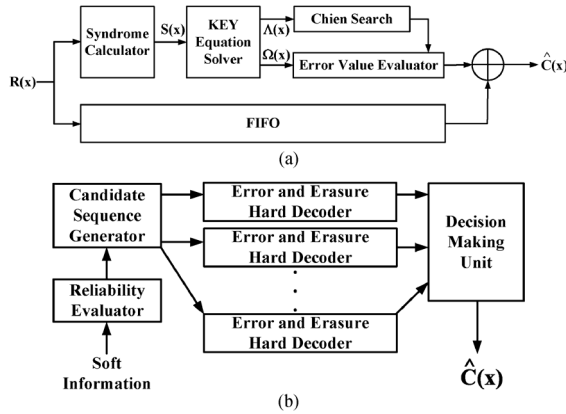

Fig. 2. Conventional hard and soft decoding processes. (a) Conventional RS decoding process. (b) GMD decoding process.

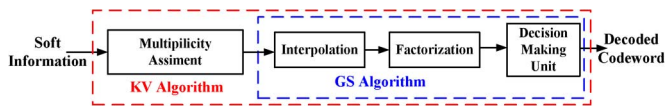

Fig. 3. Interpolation-based list decoding process.

On the other hand, Guruswami-Sudan (GS) algorithm [6], an interpolation-based list decoding method, merges probability information into algebraic interpolation process and finds out all probable codewords within its extended decoding sphere. An interpolation-based list decoding algorithm generally consists of three steps: *interpolation*, *factorization*, and *decision making* as shown in Fig. 3. The interpolation step builds a minimum degree non-zero bivariate polynomial, $Q(x, y)$, that passes each interpolation point with at least its associated multiplicity. Then the factorization step evaluates all factors of $Q(x, y)$ in form of $y - p(x)$, where $p(x)$ is one of the decoded results in the list and its degree is less than k. Finally, a decision making unit is applied to choose the output codeword from the list. Recently, based on GS algorithm, many researches, such as Koetter-Vardy (KV) algorithm [7], [11] and low-complexity Chase (LCC) method [8], utilize a multiplicity assignment unit to determine interpolation points and corresponding multiplicities according to the reliability information for reducing the computing complexity.

The low-complexity VLSI architectures for soft decoders have been developed in recent years [9], [11]–[14]. However, these soft decoders still require several times area than the hard decoders does and practical implementations have not been applied. Therefore, this paper will provide a decision-confined soft decoder chip to enhance error correcting capability while maintaining area efficiency.

III. Proposed Decision-Confined Decoding Algorithm

In Chase algorithm, several candidate sequences are generated and decoded as candidate codewords. Then a decision making unit is utilized to choose the most probable one as output codeword from the candidate codeword list. However, some of candidate sequences may not be successfully decoded due to having more than t errors, resulting in unnecessary computations. For a successful RS decoding procedure, the number of roots found by Chien search must be equal to the degree of $\Lambda(x)$. Checking the number of roots after Chien search for all candidates sequence costs a large amount of computation complexity and latency; therefore, it is necessary to find an efficient way for determining the most reliable candidate codeword. From our simulation results of RS (255, 239; 8) codes, there is over 99.5% probability that Chien search finds less than t roots for a degree-t $\Lambda(x)$ while more than t errors occur in the received data. According to this characteristic, a decision-confined algorithm is proposed for enhancing decoding efficiency with fewer number of decoded candidate sequences and without a decision making unit as shown in Algorithm-1.

Algorithm-1: Decision-Confined Algorithm

Input: *The received sequence* $R(x)$

The reliability of each bit

Initial: *η-bit integers*: $i = 1, \gamma = \gamma' = 0$.

Step 1.

Calculate syndrome polynomial $S(x)$.

Evaluate η LRPs: $\underline{L'} = [l'_1, l'_2, \ldots, l'_\eta]$, *and*

corresponding error values: $\underline{E'} = [e'_1, e'_2, \ldots, e'_\eta]$.

Step 2.

$\gamma' = \gamma, \gamma = i \oplus (i \gg 1)$ (*Gray code*)

Find the different bit between γ and γ': κ-th bit

Update syndrome polynomial:

$S^{[i]}(x) = \sum_{j=1}^{2t}(S_j^{[i-1]} + e'_\kappa(\alpha^{j \times l'_\kappa}))x^{j-1}$

Step 3.

Calculate $\Lambda^{[i]}(x)$ from key equation solver with $S^{[i]}(x)$

If $(deg(\Lambda^{[i]}(x)) < t)$

Go to Step 5

else if $(i = 2^\eta - 1$ *and* $deg(\Lambda^{[i]}(x)) = t)$

Go to Step 4

else

$i = i + 1$

Go to Step 2

Step 4.

Calculate $\Lambda(x)$ from key equal solver with $S(x)$

Step 5.

Evaluate error location and error value to obtain $e(x)$

Output: *The decoded codeword*: $\hat{C}(x) = R(x) \oplus e(x)$.

End

Based on the received soft information, η LRPs $\underline{L'} = [l'_1, l'_2, \ldots, l'_\eta]$ as well as corresponding error values: $\underline{E'} = [e'_1, e'_2, \ldots, e'_\eta]$ are determined in Step 1. In the meantime, the syndrome polynomial $S(x)$ is calculated. In Step 2, the overhead is too high for evaluating 2^η syndrome polynomials with (1) for 2^η candidate sequences. Instead, we apply a syndrome updater to obtain the i-th syndrome polynomial $S^{[i]}(x)$ from the syndrome polynomial $S(x)$. Notice that, there are only η uncommon points for all the candidate sequences. Therefore, the i-th candidate sequence can be constructed by adding the error pattern $e_f^{[i]}(x)$ induced by the bit flipping procedure to the received data, and the $e_f^{[i]}(x)$ is of the form:

$$e_f^{[i]}(x) = a_1^{[i]} e'_1 x^{l'_1} + a_2^{[i]} e'_2 x^{l'_2} + \cdots + a_\eta^{[i]} e'_\eta x^{l'_\eta}, \quad (5)$$

where $a_j^{[i]} \in GF(2)$ for $i = 1 \sim 2^\eta - 1$ and $j = 1 \sim \eta$. The i-th syndrome polynomial can be derived as

$$S^{[i]}(x) = \sum_{j=1}^{2t} \left(S_j + e_f^{[i]}(\alpha^j) \right) x^{j-1}. \quad (6)$$

To further improve the complexity, $S^{[i]}(x)$ can be obtained from the previous syndrome polynomial $S^{[i-1]}(x)$. The candidate sequences can be generated according to Gray code based bit flipping order, leading to only one different bit between successive two candidate sequences. Accordingly, $S^{[i]}(x)$ can be simplified as

$$S^{[i]}(x) = \sum_{j=1}^{2t} \left(S_j^{[i-1]} + e_f'^{[i]}(\alpha^j) \right) x^{j-1}, \quad (7)$$

where $e_f'^{[i]}(x) = e'_\kappa x^{l'_\kappa}$ is difference polynomial between $e_f^{[i-1]}$ and $e_f^{[i]}(x)$ and the only one different bit is at l'_κ-th position. For example, if the flipped LRP is the 25th symbol, 4th bit of the received data, the values of e'_κ and $\alpha^{l'_\kappa}$ will be α^4 and α^{25} respectively. The computations of the syndrome updater will be reduced since there is only one bit required to be flipped while the decoder generates one candidate codeword, leading to fewer multiplication computations.

After updating the syndrome polynomial $S^{[i]}(x)$, the corresponding error location polynomial $\Lambda^{[i]}(x)$ is calculated by key equation solver in Step 3. An unique decoding condition is set to avoid unnecessary computations for those candidate sequences with more than t errors. Only the $\Lambda^{[i]}(x)$ with degree less than t will be sent to Chien search and error value evaluator for evaluating the error locations and error values. It is highly probable for these candidate sequences with less than t errors. Once the condition is met, this candidate sequence will be completely decoded as output codeword. Otherwise, the decoder will generate a new syndrome polynomial with another bit flipped sequence. Notice that, if all the generated error location polynomials have degree of t, the proposed decoder will decode received data as a hard RS decoder to guarantee error correcting capability. Therefore, the proposed decision-confined decoding algorithm will not cause an error floor at high SNR region.

Fig. 4 shows the RS (255, 239; 8) simulation results for our proposed decision-confined algorithm with $\eta = 3 \sim 5$ under BPSK modulation and AWGN channel. Our proposal with $\eta =$

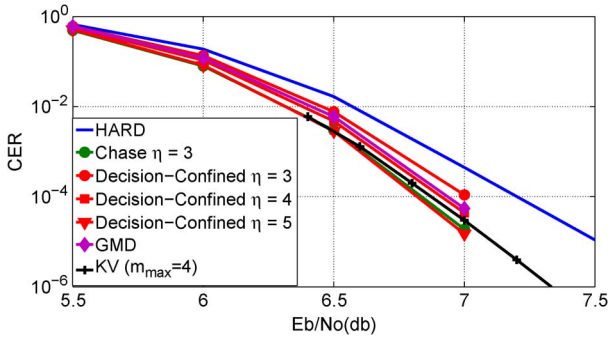

Fig. 4. Simulation results for RS (255, 239; 8) codes.

TABLE I
AVERAGE NUMBER OF EXECUTION FOR RS (255, 239; 8) CODES

	Proposed Algorithm $\eta = 5$			Chase Algorithm $\eta = 3$		
E_b/N_0	6.0	6.5	7.0	6.0	6.5	7.0
Syndrom Calculator	5.22	1.30	1.07	8	8	8
Key Equation Solver	5.22	1.30	1.07	8	8	8
Chien Search	1	1	1	8	8	8
Error Value Evaluator	1	1	1	8	8	8

5 can provide 0.4 dB coding gain at 10^{-4} codeword error rate (CER) over the hard RS decoder, and also outperforms GMD and KV algorithms. As compared with Chase algorithm with $\eta = 3$, our proposed method can achieve similar error correcting ability with $\eta = 5$. Notice that, the average computation complexity of our proposal is much less than Chase algorithm although it requires more LRPs to achieve similar error correcting performance. According to our simulation results of 10^6 RS (255, 239; 8) codewords, Table I demonstrates the average number of execution of each component for decoding a RS (255, 239; 8) codeword during different E_b/N_0 region. If our approach with $\eta = 5$ is applied at $E_b/N_0 = 7$, the average computation of syndrome calculator, key equation solver, Chien search and error value evaluator are 1.07, 1.07, 1 and 1 times respectively. However, Chase algorithm with $\eta = 3$ executes 2^3 calculation for all the decoding blocks.

IV. ITERATION-REDUCED RiBM ALGORITHM

According to our decision-confined algorithm, the coding gain can be further enhanced as the number of candidate sequences is increased, i.e. using more LRPs. However, the decoding period is limited and several parallel key equation solvers are required if more candidate sequences are generated. This will induces significant hardware increment. On the contrary, accelerating the decoding procedure can allow more candidate sequences decoded in the same decoding period without duplicating hardware. This section discusses the acceleration of key equation solver, which is the critical part of our design.

Berlekamp Massey (BM) algorithm is one of the most popular methods for solving the key equation, which evaluates $\Lambda(x)$ and $\omega(x)$ in $2t$ iterations. To improve the critical path delay, Sarwate and Shanbhag proposed a reformulated inversionless Berlekamp Massey (RiBM) algorithm [22] with a low-complexity homogeneous architecture in 2001. Based on RiBM algorithm, we proposed an iteration-reduced RiBM (IR-RiBM) algorithm to reduce the number of iterations from $2t$ to t by evaluating current discrepancy and predicting next discrepancy at the same iteration.

The discrepancy at each iteration k is defined as

$$\Phi^{(k)} = \sum_{j=0}^{L_{k-1}} \Lambda_j^{(k-1)} S_{k-j} \quad \text{for } k = 1 \sim 2t. \tag{8}$$

Notice that $\Lambda^{(k-1)}(x)$ represents the error location polynomial generated in k-th iteration, L_{k-1} can be viewed as its degree, and $\Lambda_j^{(k-1)}$ represents its j-th coefficient. RiBM algorithm, which only has one finite field multiplier (FFM) and one finite field adder (FFA) critical path delay, uses the characteristic that $\Phi^{(k)}$ is exactly the coefficients of x^k in the polynomial product of $S(x)$ and $\Lambda^{(k)}(x)$

$$\begin{aligned}\Delta^{(k)}(x) &= \Lambda^{(k)}(x) \cdot S(x) \\ &= \delta_0^{(k)} + \delta_1^{(k)} \cdot x + \cdots + \delta_k^{(k)} \cdot x^k + \cdots \\ &= \Omega^{(k)}(x) + x^{2t} \cdot \omega^{(k)}(x),\end{aligned} \tag{9}$$

where $\omega^{(k)}(x)$ of degree at most $t-1$ contains the high-order terms and can be utilized for error value evaluation. Accordingly, operations for updating $\Lambda^{(k)}(x)$ can be merged in those for calculating $\Delta^{(k)}(x)$. The $\delta_i^{(k)}$ is iteratively calculated as

$$\delta_i^{(k)} = q_0 \delta_i^{(k-1)} + q_1 \theta_{i-1}^{(k-1)}. \tag{10}$$

Algorithm-2: Iteration-Reduced RiBM Algorithm w/o $\omega(x)$

Initialization: $L_0 = 0, \alpha_0 = S_1, c_0 = 1, \beta_0 = S_2 - S_1^2$
$\hat{\delta}_i^{(0)} = \hat{\theta}_i^{(0)} = S_{i+1}, (i = 0, \ldots, 2t-1)$
$\hat{\delta}_{2t}^{(0)} = \hat{\theta}_{2t}^{(0)} = 1$

for $\tau = 1$ **to** t **do**

begin

Step 1.

Case 1: $\hat{\delta}_0^{(\tau-1)} = 0$ & $\hat{\delta}_1^{(\tau-1)} = 0$

$g_0 = c_{\tau-1}, g_1 = g_2 = g_3 = 0$
$\hat{\theta}_i^{(\tau)} = \hat{\theta}_i^{(\tau-1)}, \hat{\theta}_{2t-2\tau}^{(\tau)} = \hat{\theta}_{2t-2\tau+1}^{(\tau)} = 0$
$L_\tau = L_{\tau-1}, \alpha_\tau = \alpha_{\tau-1}, c_\tau = c_{\tau-1}$

Case 2: $\hat{\delta}_0^{(\tau-1)} = 0$ & $\hat{\delta}_1^{(\tau-1)} \neq 0$ & $L_{\tau-1} > \tau - 1$

$g_0 = c_{\tau-1}, g_1 = g_2 = 0, g_3 = \hat{\delta}_1^{(\tau-1)}$
$\hat{\theta}_i^{(\tau)} = \hat{\theta}_i^{(\tau-1)}, \hat{\theta}_{2t-2\tau}^{(\tau)} = \hat{\theta}_{2t-2\tau+1}^{(\tau)} = 0$
$L_\tau = L_{\tau-1}, \alpha_\tau = \alpha_{\tau-1}, c_\tau = c_{\tau-1}$

Case 3: $\hat{\delta}_0^{(\tau-1)} = 0$ & $\hat{\delta}_1^{(\tau-1)} \neq 0$ & $L_{\tau-1} \leq \tau - 1$

$g_0 = c_{\tau-1}, g_1 = g_2 = 0, g_3 = \hat{\delta}_1^{(\tau-1)}$
$\hat{\theta}_i^{(\tau)} = \hat{\delta}_{i+2}^{(\tau-1)}$
$L_\tau = 2\tau - L_{\tau-1}, \alpha_\tau = \hat{\delta}_2^{(\tau-1)}, c_\tau = \hat{\delta}_1^{(\tau-1)}$

Case 4: $\hat{\delta}_0^{(\tau-1)} \neq 0$ & $L_{\tau-1} \leq \tau - 1$

$g_0 = c_{\tau-1} \cdot \hat{\delta}_0^{(\tau-1)}, g_1 = \beta_{\tau-1}, g_2 = \hat{\delta}_0^{(\tau-1)} \cdot \hat{\delta}_0^{(\tau-1)}$
$g_3 = 0, \hat{\theta}_i^{(\tau)} = \hat{\delta}_{i+1}^{(\tau-1)}, \hat{\theta}_{2t-2\tau}^{(\tau)} = 0$
$L_\tau = 2\tau - 1 - L_{\tau-1}, \alpha_\tau = \hat{\delta}_1^{(\tau-1)}, c_\tau = \hat{\delta}_0^{(\tau-1)}$

Case 5: $\hat{\delta}_0^{(\tau-1)} \neq 0$ & $L_{\tau-1} > \tau - 1$

$g_0 = c_{\tau-1}^2, g_1 = 0, g_2 = c_{\tau-1} \cdot \hat{\delta}_0^{(\tau-1)}, g_3 = \beta_{\tau-1}$

$\hat{\theta}_i^{(\tau)} = \hat{\theta}_i^{(\tau-1)}, \hat{\theta}_{2t-2\tau}^{(\tau)} = \hat{\theta}_{2t-2\tau+1}^{(\tau)} = 0$

$L_\tau = L_{\tau-1}, \alpha_\tau = \alpha_{\tau-1}, c_\tau = c_{k-1}$

Step 2.

$$\hat{\delta}_i^{(\tau)} = g_0\hat{\delta}_{i+2}^{(\tau-1)} + g_1\hat{\delta}_{i+1}^{(\tau-1)} + g_2\hat{\theta}_{i+1}^{(\tau-1)} + g_3\hat{\theta}_i^{(\tau-1)}$$

$$\beta_\tau = g_0(\alpha_\tau\hat{\delta}_2^{(\tau-1)} + c_\tau\hat{\delta}_3^{(\tau-1)}) + g_1(\alpha_\tau\hat{\delta}_1^{(\tau-1)} + c_\tau\hat{\delta}_2^{(\tau-1)}) + g_2(\alpha_\tau\hat{\theta}_1^{(\tau-1)} + c_\tau\hat{\theta}_2^{(\tau-1)}) + g_3(\alpha_\tau\hat{\theta}_0^{(\tau-1)} + c_\tau\hat{\theta}_1^{(\tau-1)})$$

end

Output: $\Lambda_i^{(t)} = \hat{\delta}_i^{(t)}, (i = 0, 1, \ldots, t)$.

It can be found that $L_k > L_{k-1}$ only if $\Phi^{(k)} \neq 0$ and $2L_{k-1} \leq k - 1$, implying that L can be increased only once within any two successive iterations [23]. Therefore, we can reduce the number of iterations of RiBM algorithm from $2t$ to t by calculating the discrepancy of the odd iteration $\Phi^{(2\tau-1)}$ as well as predicting the discrepancy of the even iteration $\varepsilon^{(2\tau)}$ in the meantime for $\tau = 1 \sim t$, where

$$\Phi^{(2\tau-1)} = \sum_{j=0}^{L_{\tau-1}} \Lambda_j^{(2\tau-2)} S_{2\tau-j-1}. \quad (11)$$

$$\varepsilon^{(2\tau)} = \sum_{j=0}^{L_{\tau-1}} \Lambda_j^{(2\tau-2)} S_{2\tau-j}. \quad (12)$$

Notice that $\Phi^{(2\tau-1)}$ and $\varepsilon^{(2\tau)}$ are exactly the coefficients of $x^{2\tau-2}$ and $x^{2\tau-1}$ in (9). Then the updating criterion can be modified into five cases as shown in Fig. 5 and $\delta_i^{(2\tau)}$ is iteratively calculated as

$$\delta_i^{(2\tau)} = \delta_i^{(2\tau-2)} + g_1\delta_{i-1}^{(2\tau-2)} + g_2\theta_{i-1}^{(2\tau-2)} + g_3\theta_{i-2}^{(2\tau-2)}, \quad (13)$$

where $g_1 \sim g_3$ will be generated based on different cases. Moreover, according to (9), $\omega^{(k)}(x)$ will not affect discrepancy calculation since it consists of coefficients of x^i in $\Delta^{(k)}(x)$ for $i > 2t$. Therefore, the calculation for $\omega(x)$ in RiBM algorithm can be eliminated if only $\Lambda(x)$ is required.

Algorithm-2 describes the iteration-reduced RiBM algorithm for evaluating $\Lambda(x)$ only. The initialization step utilizes either syndrome values or 1 for each storage element. In Step 1, each signal used for $\delta_i^{(\tau)}$ calculation is computed under five different cases according to Fig. 5. Based on these signals, $\delta_i^{(\tau)}$ can be updated in the step 2. Notice that, $\delta_{i+\tau}^{(i+\tau)}$ cannot be changed by $\delta_i^{(\tau)}$ for any $i < k$ [22]. Hence, we define $\hat{\delta}_i^{(\tau)} = \delta_{i+2\tau-2}^{(\tau)}$, implying that $\hat{\delta}_0^{(\tau)}$ and $\hat{\delta}_1^{(\tau)}$ become discrepancies of odd and even iterations respectively. As a result, the equation for updating $\hat{\delta}_i^{(\tau+1)}$ becomes

$$\begin{aligned}\hat{\delta}_i^{(\tau)} &= \delta_{i+2\tau-2}^{(\tau)} \\ &= g_0\delta_{i+2\tau-2}^{(\tau-1)} + g_1\delta_{i+2\tau-3}^{(\tau-1)} + g_2\theta_{i+2\tau-3}^{(\tau-1)} + g_3\theta_{i+2\tau-4}^{(\tau-1)} \\ &= g_0\hat{\delta}_{i+2}^{(\tau-1)} + g_1\hat{\delta}_{i+1}^{(\tau-1)} + g_2\hat{\theta}_{i+1}^{(\tau-1)} + g_3\hat{\theta}_i^{(\tau-1)}. \end{aligned} \quad (14)$$

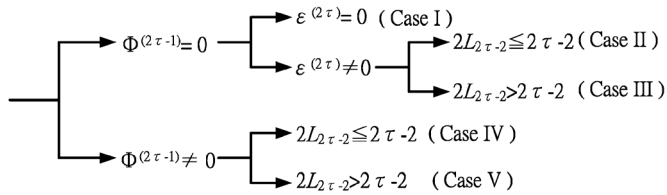

Fig. 5. The updating criterion for iteration-reduced RiBM algorithm.

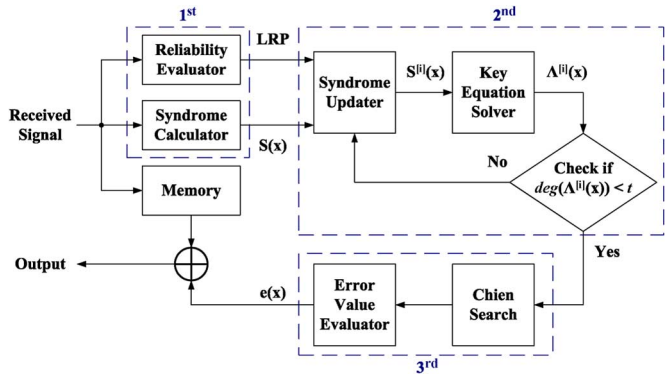

Fig. 6. Decision-confined soft RS decoding process.

In addition, the original definition of β_τ used for $g_1 \sim g_3$ calculation is

$$\beta_\tau = \hat{\delta}_1^{(\tau)}c_\tau - \hat{\delta}_0^{(\tau)}\alpha_\tau. \quad (15)$$

However, evaluating β_τ right after updating $\delta_0^{(\tau)}$ and $\delta_1^{(\tau)}$ leads to long critical path. To reduce the critical path, $\hat{\delta}_1^{(\tau-1)} \sim \hat{\delta}_3^{(\tau-1)}$ and $\hat{\theta}_1^{(\tau-1)} \sim \hat{\theta}_3^{(\tau-1)}$ can be applied to replace $\hat{\delta}_1^{(\tau)}$ and $\hat{\delta}_0^{(\tau)}$. And (15) becomes

$$\beta_\tau = g_0\left(\alpha_\tau\hat{\delta}_2^{(\tau-1)} + c_\tau\hat{\delta}_3^{(\tau-1)}\right) + g_1\left(\alpha_\tau\hat{\delta}_1^{(\tau-1)} + c_\tau\hat{\delta}_2^{(\tau-1)}\right) + g_2\left(\alpha_\tau\hat{\theta}_1^{(\tau-1)} + c_\tau\hat{\theta}_2^{(\tau-1)}\right) + g_3\left(\alpha_\tau\hat{\theta}_0^{(\tau-1)} + c_\tau\hat{\theta}_1^{(\tau-1)}\right) \quad (16)$$

After t iterations, $\Lambda^{(t)}(x)$ is stored in $\hat{\delta}_0^{(t)} \sim \hat{\delta}_t^{(t)}$.

Notice that, our proposed IR-RiBM algorithm reduces the required iterations by evaluating current and next discrepancies at the same time; therefore, it will not cause any loss in error correcting performance. In addition, as RiBM algorithm, it can provide extremely regular architecture, which will be discussed in Section V.

V. VLSI Architectures for Proposed Decoder

As shown in Fig. 6, a soft RS (255, 239; 8) decoder based on our decision-confined decoding algorithm is presented with 3-stage pipeline architecture for the 2.5 Gb/s throughput requirement of optical communication systems.

At the first stage, the reliability evaluator determines 5 LRPs and the syndrome calculator computes the syndrome polynomial $S(x)$ in the meanwhile. Then the syndrome updater iteratively modifies the syndrome polynomial $S^{[i]}(x)$ according to the LRPs with Gray code based bit-flipping method, and the iteration-reduced key equation solver computing the corresponding $\Lambda^{[i]}(x)$ at the second stage. Once the degree of $\Lambda^{[i]}(x)$ is less than t, the parallel-2 Chien search and the BP-based error value evaluator are applied to find error locations as well as error values at the last stage. According to our timing schedule shown

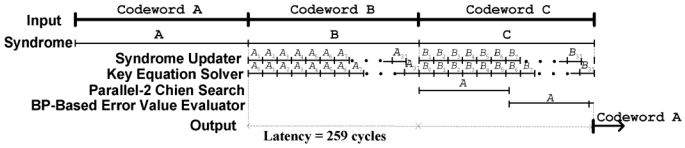

Fig. 7. Decoding scheme of the decision-confined soft RS decoder.

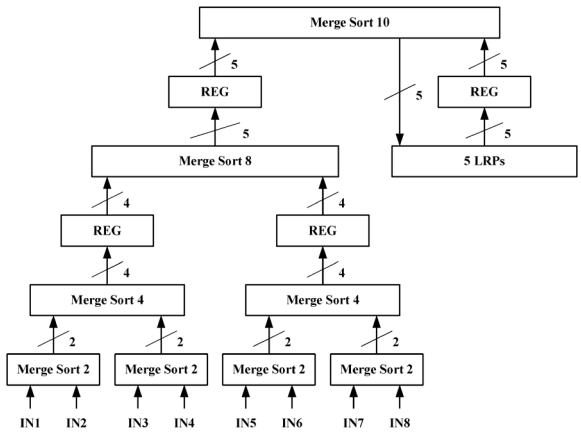

Fig. 8. Reliability evaluator.

in Fig. 7, the first stage requires 259 clock cycles due to the pipeline reliability evaluator architecture. At the second stage, each candidate sequence will consume 8 cycles for key equation solver, indicating that totally 256 cycles are demanded for 5 LRPs. Then the parallel-2 Chien search and BP-based error value evaluator spend 128 and 92 cycles respectively, resulting in 220 cycles at the last stage. The following subsections will show the unique blocks of our approach in contrast to conventional hard decoders.

A. Reliability Evaluator

For a RS (255, 239; 8) code over $GF(2^8)$, a symbol consisting of 8 bits is fed into the reliability evaluator each cycle for choosing 5 LRPs. Hence, the reliability evaluator has to compare $8 + 5 = 13$ values and determine 5 LRPs. To reduce the computation time as well as the critical path, the *merge sort concept* introduced in [24] is exploited for designing the reliability evaluator with pipeline architecture as shown in Fig. 8. The first and second stages calculate the 5 LRPs of the inputted 8 bits according to the *divide-and-conquer* concept. The third stage decides the new temporary LRPs among the output of the second stage and the temporary LRPs from previous calculation. Due to the delay of pipelined design of the reliability evaluator, the required latency is slight enlarged from 255 to 259 clock cycles.

B. Syndrome Updater

Instead of recalculating the i-th syndrome polynomial $S^{[i]}(x)$ with syndrome calculator, the syndrome updater calculates it by updating the $(i-1)$-th syndrome polynomial $S^{(i-1)}(x)$, leading to further hardware reduction. According to (7), $S^{[i]}(x)$ can be obtained with $e'_\kappa \times x^{l'_\kappa}$, where the only one different bit between successive two candidate sequences is at l'_κ-th position. The proposed syndrome updater shown in Fig. 9 utilizes look up Table I (LUT1) and LUT2 to find the e'_κ and $\alpha_{l'_\kappa}$ based on the results of reliability evaluator. Notice that, there are at most 2^5 candidates sequences for each received data and 259 computation cycles for each pipeline stage according to Fig. 7. The finite field

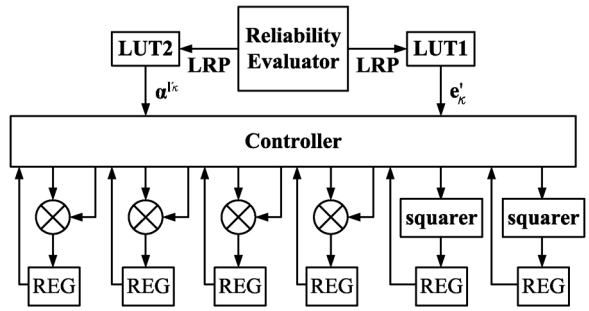

Fig. 9. Syndrome updater.

multipliers (FFMs) and squarers can be shared for computing 16 updated syndrome values. Our design only requires 4 FFMs and 2 squares for updating a syndrome polynomial within 8 cycles.

C. Iteration-Reduced RiBM Key Equation Solver

The key equation solver, which generally requires $2t$ iterations, is the most critical part of a RS decoder. For our soft RS (255, 239; 8) decoder, there are 259 cycles to deal with 32 candidate sequences in a pipeline stage, where solving the key equation for a candidate sequence generally needs $2 \times 8 = 16$ cycles. Using two key equation solvers in parallel is one way for meeting the timing constraint; however, it results in high hardware complexity and difficult signal controlling. Instead, the iteration-reduced RiBM algorithm is applied in our decoder to reduce the number of required iterations from $2t$ to t, i.e. from 16 to 8. Besides, only $\Lambda^{(i)}(x)$ is demanded to be verified whether the degree is less than t in the decoding constraint defined in Algorithm-1. The Chien search and error value evaluator do not need $\Lambda^{(i)}(x)$ and $\omega^{(i)}(x)$ to evaluate the error locations and error values if the degree of $\Lambda^{(i)}(x)$ is equal to t. Hence, $\omega^{(i)}(x)$ is not necessary to be computed in this complex part, leading to further hardware reduction.

According to Algorithm-2, the structure of the processing element of iteration-reduced RiBM (IR-PE) is depicted in Fig. 10. Each IR-PE contains 2 finite field adders (FFAs), 3 FFMs and 2 registers. Ignoring the control unit, the hardware requirement of this architecture is $2t + 1$ IR-PEs as illustrated in Fig. 11. The extremely regular structure offers some advantage in VLSI circuit layouts. At the beginning of the process, Λ_0 is initialized to 1 and stored into $\mathrm{IR-PE}_{2t}$. The initial loading of $\mathrm{IR-PE}_0 \sim \mathrm{IR-PE}_{2t-1}$ are $S_1 \sim S_{2t}$, respectively. In each iteration, Λ_0 is calculated and stored into $\mathrm{IR-PE}_{2t-2\tau}$. After t cycles, this iterative algorithm is completed and the coefficients of $\Lambda(x)$ are stored in the processors $\mathrm{IR-PE}_0 \sim \mathrm{IR-PE}_t$. Notice that the discrepancies $\Phi^{(2\tau-1)}$ and $\varepsilon^{(2\tau)}$ are always in the first and second IR-PEs ($\hat{\delta}_0^{(\tau)}$ and $\hat{\delta}_1^{(\tau)}$).

As compared with the architecture of [23], our method obviously improves the critical path and provides the advantage of regularity while offering the same number of iterations. The critical path in [23] is an adder tree discrepancy computation and longer than $3 \times (T_{mult} + T_{add})$, where T_{mult} and T_{add} are the delay time of FFM and FFA respectively. However, the critical path of our proposal passes through only 2 FFMs and 2 FFAs in the control signal β_τ updating procedure. In contrast to the RiBM architecture with $T_{mult} + T_{add}$ critical path delay, the proposed IR-RiBM needs only half number of iterations but has doubled critical path delay. However, for a 2.5 Gb/s soft RS Chip design, 320 MHz operating frequency is sufficient for the requirement. On the other hand, the improvement of the number of iterations in key equation solver procedure is more important

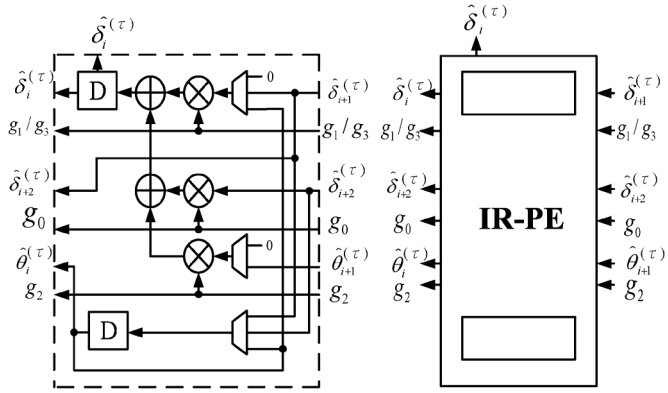

Fig. 10. Processing element (IR-PE) of iteration-reduced RiBM key equation solver.

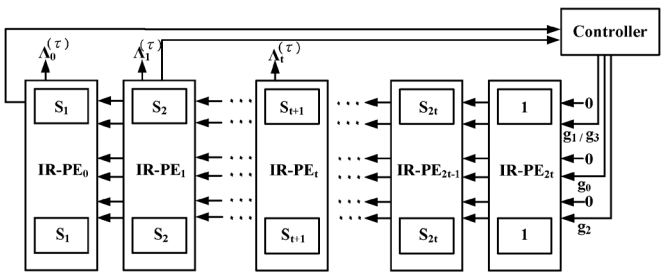

Fig. 11. Homogeneous architecture of iteration-reduced RiBM key equation solver.

for enhancing the error correcting performance with more LRPs, implying that our IR-RiBM is more suitable.

D. Error Value Evaluator

Conventionally, the corresponding error values can be calculated with $\Lambda(x)$ and $\Omega(x)$ based on the Forney's algorithm after Chien search evaluates the error locations. However, the parallel architecture for meeting timing constraint induces costly hardware complexity for error value calculation. On the other hand, the Björck-Pereyra (BP) [16] can compute the error values by solving the Vandermonde relation between the syndrome S_i and error locators X_i as:

$$\begin{bmatrix} X_1 & X_2 & \cdots & X_t \\ X_1^2 & X_2^2 & \cdots & X_t^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^t & X_2^t & \cdots & X_t^t \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_t \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_t \end{bmatrix}, \qquad (17)$$

.

The error value evaluator shown in Fig. 12 is implemented based on the following procedure:

• Step 1:

$$\text{for } (k = 1; k < t, k = k + 1)$$
$$\text{for } (i = t; i > k, i = i - 1)$$
$$S_i = S_i - X_k S_{i-1}$$

• Step 2:

$$\text{for } (k = t - 1; k > 0, k = k - 1)$$
$$\text{for } (i = k + 1; i \le t, i = i + 1)$$
$$S_i = S_i/(X_i - X_{i-k})$$
$$S_{i-1} = S_{i-1} - S_i$$

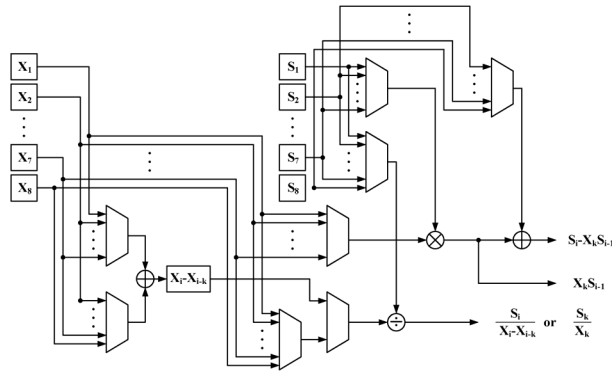

Fig. 12. BP-based error value evaluator.

• Step 3:

$$\text{for } (k = 1; k \le t, k = k + 1)$$
$$S_i = S_k/X_k$$

The variable S_i, which initially represents the i-th syndrome value, is updated iteratively. Each calculation of the syndrome represents a row operation in (17). The control logic determines the computation order of the S_i and X_i, and the computation results will be used to update each S_i value. After all computations, S_i indicates the i-th error value. Although BP algorithm takes t^2 cycles time to work, it matches the timing schedule in our design.

VI. Implementation Result

As shown in Table II, the proposed decision-confined soft RS decoder is compared to the LCC soft RS decoder and the pRiBM hard RS decoder. Both our proposed decoder and the pRiBM decoder have 3-stage pipeline architecture while the LCC decoder has 4-stage pipeline architecture. The numbers of required clock cycles in each pipeline stage are 259, 528, 255 for the proposed decision-confined, the LCC and the pRiBM decoders, respectively. Our design and LCC decoder utilize 5 LRPs and 3 LRPs respectively for generating candidate sequences and have similar error correcting performance. If the complexity is normalized to XOR gate, our proposed decision-confined decoder is around 22,534 XOR gate while the LCC decoder and the pRiBM decoder are about 38,671 and 25,796 XOR gates, respectively. Due to fewer number of storage elements, our proposed decoder can save 42% hardware complexity as compared to the LCC decoder which requires to store all the candidate codewords. In addition, the complexity of the LCC decoder in Table II excludes the complexity of the decision making unit, which is required for deciding the output codeword from the list. The pRiBM decoder has the shortest critical path delay because of applying the pipeline multipliers. Notice that, for a 2.5 Gb/s soft RS chip design, 320 MHz operation frequency is sufficient for the requirement.

Fig. 13 shows the decision-confined soft RS (255, 239; 8) decoder die photo, which is implemented with cell-based design flow and fabricated in 90 nm 1P9M CMOS process. The chip is verified by Agilent 93000 SOC test system and the Shmoo plot in Fig. 14 indicates that our design can achieve 320 MHz operation frequency with 19.6 mW power at 0.98 V supply.

TABLE II
COMPARISON TABLE FOR RS (255, 239; 8) DECODER

	Decision-Confined with η = 5	LCC with η = 3 [9]	pRiBM [25]
Code Type	Soft RS (255, 239; 8)	Soft RS (255, 239; 8)	Hard RS (255, 239; 8)
$GF(2^8)$ Constant FFM	32	8	0
$GF(2^8)$ FFM	68	66	99
$GF(2^8)$ FFA	87	108	73
2-to1 MUX (Bit)	1592	1109	712
Register (Bit)	834	1606	2136
ROM (Byte)	520	1024	256
RAM (Byte)	0+256×3	68+256 × 8	0+256×3
Latency	259×3	520×4	255×3
Critical Path Delay	$T_{mux} + 2 \times T_{add} + 2 \times T_{mult}$	$T_{mux} + T_{add} + T_{mult}$	$T_{mux} + 3 \times T_{add}$

* The complexity ratio over $GF(2^8)$ among XOR, CFFM, FFM, FFA, MUX, Register, ROM (byte) and RAM (byte) is 1: 20: 100: 8: 1: 3: 8: 8.
** T_{mux}, T_{add} and T_{mult} are the delay time of MUX, FFA and FFM, respectively.

TABLE III
COMPARISON OF TIME AND HARDWARE COMPLEXITY WITH HARD RS DECODER

	Decision-Confined	Decision-Eased [12]	pRiBM [25]	pDCME [26]	DCME [27]
Code Type	Soft RS (255, 239; 8)	Soft RS (224, 216; 4)	Hard RS (255, 239; 8)	Hard RS (255, 239; 8)	Hard RS (255, 239; 8)
Technology	90 nm	90 nm	90 nm	0.13 μm	0.25 μm
Operation Frequency	320 MHz (Measurement)	312.5MHz (Post Layout)	690 MHz (Synthesis)	660 MHz (Synthesis)	200 MHz (Synthesis)
Core Area	216,225 μm^2	221,030 μm^2	-	-	-
Gate Count (Normalized) *	45.3 K (1.07)	30.0 K (1.42)	43.6 K (1.03)	53.2 K (1.26)	42.2 K (1)
Latency	259×3	224×2	255×3	255×3	255×3
Throughput	2.56 Gb/s	2.5 Gb/s	5.52 Gb/s	5.28 Gb/s	1.6 Gb/s
Power Consumption	19.6 mW @ 320 MHz	-	-	-	-
Coding Gain	0.4 dB @ 10^{-4} CER	0.5 dB @ 10^{-5} BER	-	-	-

* The normalized gate count is calculate as: $(gate\ count) \times (8/t)/(gate\ count\ of\ DCME)$.

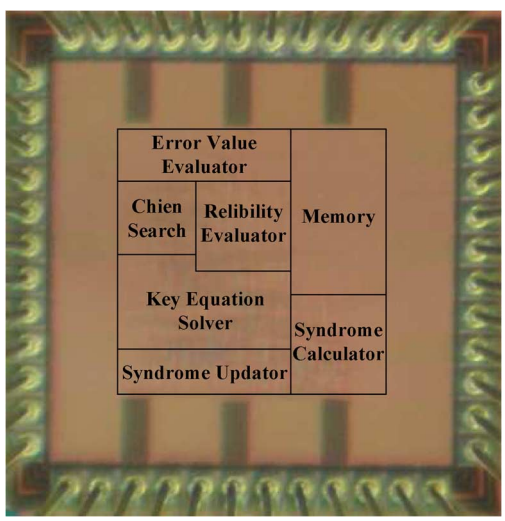

Fig. 13. Microphoto of decision-confined soft RS (255, 239; 8) chip.

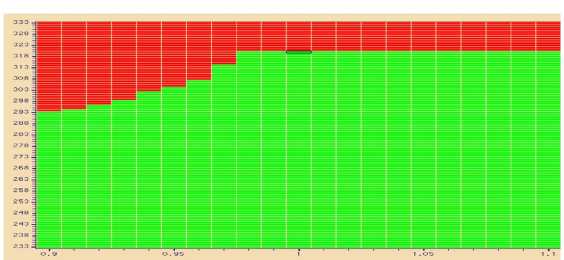

Fig. 14. Shmoo plot of decision-confined soft RS (255, 239; 8) chip.

Table III compares implementation results of our soft RS decoder with other soft and hard RS decoders. Our chip with 45.3 K gates is more area-efficient than the decision-eased soft RS (224, 216; 4) decoder and is comparable with a conventional hard decoder. All the hard decoders are designed with 3-stage pipeline architecture for the throughput. Due to applying serial syndrome calculation and Chien search, the latency of each pipeline stage is equal to the codeword length. Our design has slightly more latency in each pipeline stage because the reliability evaluator takes 259 cycles to determine 5 LRPs from 255 successive symbols. Moreover, it can fit well for 10–40 Gb/s with 16 RS decoders in optical fiber systems and 2.5 Gb/s GPON applications with 0.4 dB coding gain over hard decoders at 10^{-4} CER.

VII. CONCLUSION

This paper provides a novel decoding algorithm and its area-efficient architecture for soft RS decoders. Instead of applying several hard RS decoders and determining the most probable candidate codeword like Chase algorithm, our approach produces only one codeword by confining the degree of error location polynomial, leading to significant complexity reduction. Moreover, the IR-RiBM algorithm is proposed to allow more LRPs used in the limited operation latency for further error correcting performance enhancement. For RS (255, 239; 8) codes, our method can achieve 0.4 dB coding gain at 10^{-4} CER over hard decoders. According to the measurement results, the proposed soft RS decoder can achieve 2.56 Gb/s throughput with gate count of 45.3 K and power consumption of 19.6 mW. Consequently, our design can provide more powerful correcting ability with a high-speed and area-efficient solution for optical communications applications.

REFERENCES

[1] *Forward Error Correction for Submarine Systems*, ITU-T Std. G.975, 1996.

[2] *Gigabit-Capable Passive Optical Networks (G-PON): Transmission Convergence Layer Specification*, ITU-T Std. G.984.3, 2008.

[3] G. D. Forney, "Generalized minimum distance decoding," *IEEE Trans. Inform. Theory*, vol. 12, pp. 125–131, Apr. 1966.

[4] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 170–182, Jan. 1972.

[5] H. Tang, Y. Liu, M. Fossorier, and S. Lin, "On combining chase-2 and GMD decoding algorithms for nonbinary block codes," *IEEE Commun. Lett.*, vol. 5, no. 5, pp. 209–211, May 2001.

[6] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometry codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 1757–1767, Sep. 1999.

[7] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of *reed-solomon* codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.

[8] W. J. Gross, F. R. Kschischang, R. Koetter, and P. G. Gulak, "Towards a VLSI architecture for interpolation-based soft-decision Reed-Solomon decoders," *J. VLSI Signal Process*, vol. 39, pp. 93–111, 2005.

[9] X. Zhang, "High-speed VLSI architecture for low-complexity chase soft-decision Reed-Solomon decoding," in *IEEE Information Theory and Applications Workshop (ITA Workshop)*, Feb. 2009, pp. 422–430.

[10] X. Zhang and J. Zhu, "High-throughput interpolation architecture for algebraic soft-decision Reed-Solomon decoding," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 3, pp. 581–591, 2010.

[11] A. Ahmed, R. Koetter, and N. Shanbhag, "VLSI architectures for soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 648–667, Feb. 2011.

[12] Y.-M. Lin, Y.-C. Huang, C.-H. Yang, and H.-C. Chang, "A 30 K 2.5 Gb/s decision-eased soft RS (224, 216) decoder for wireless systems," in *Proc. IEEE ISIC*, Dec. 2011, pp. 164–167.

[13] X. Zhang, Y. Zheng, and Y. Wu, "A chase-type Koetter-Vardy algorithm for soft-decision Reed-Solomon decoding," in *Proc. IEEE Int. Computing, Networking and Communications Conf.*, Feb. 2012, pp. 466–470.

[14] Y. Wu, "Fast chase decoding algorithms and architectures for Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 58, no. 1, pp. 109–129, Jan. 2012.

[15] F. Garcia-Herrero, M. Canet, J. Valls, and P. Meher, "High-throughput interpolator architecture for low-complexity chase decoding of RS codes," *IEEE Trans. VLSI Syst.*, vol. 20, no. 3, pp. 568–573, 2012.

[16] A. Björck and V. Pereyra, "Solution of Vandermonde systems of equations," *Math. Computation*, vol. 24, pp. 893–903, Oct. 1970.

[17] D. C. Gorenstein and N. Zierler, "A class of cyclic linear error-correcting codes in p^m symbols," *J. Soc. Indust. Appl. Math.*, vol. 9, pp. 207–214, Jun. 1961.

[18] E. R. Berlekamp, *Algrbraic Coding Theory*. New York: McGraw-Hill, 1968.

[19] H. Shao, T. Truong, L. Deutsch, J. Yuen, and I. Reed, "A VLSI design of a pipeline Reed-Solomon decoder," *IEEE Trans. Comput.*, vol. C-34, no. 5, pp. 393–403, May 1985.

[20] R. Chien, "Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes," *IEEE Trans. Inform. Theory*, vol. IT-10, pp. 357–363, Oct. 1964.

[21] G. D. Forney, Jr., "On decoding BCH codes," *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 549–557, Oct. 1965.

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

[23] A. Raghupathy and K. J. R. Liu, "Algorithm-based low-power/high-speed Reed-Solomon decoder design," *IEEE Trans. Circuits Syst. II*, vol. 47, no. 11, pp. 1254–1270, Nov. 2000.

[24] D. Knuth, *The Art of Computer Programming, Vol. 3, Sorting and Searching*, 2nd ed. Boston, MA: Addison-Wesley, 1998.

[25] J.-I. Park, K. Lee, C.-S. Choi, and H. Lee, "High-speed low-complexity Reed-Solomon decoder using pipelined Berlekamp-Massey algorithm," in *Proc. IEEE ISOCC*, 2009, pp. 452–455.

[26] S. Lee, H. Lee, J. Shin, and J.-S. Ko, "A high-speed pipelined degree-computationless modified Euclidean algorithm architecture for Reed-Solomon decoders," in *Proc. IEEE ISCAS*, May 2007, pp. 901–904.

[27] J. Baek and M. Sunwoo, "New degree computationless modified Euclid algorithm and architecture for Reed-Solomon decoder," *IEEE Trans. VLSI Syst.*, vol. 14, no. 8, pp. 915–920, Aug. 2006.

Yi-Min Lin received the B.S. degree in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 2005. Under the advisement of Professor Chen-Yi Lee and Professor Hsie-Chia Chang, he received the Ph.D. degree from National Chiao Tung University, Hsinchu, Taiwan, in 2011.

From 2011 to 2012, he was a postdoctoral scholar in the Department of Electrical Engineering, University of California, Los Angeles, CA, USA, where he was working in the area of coding architectures for 60 GHz baseband. In November 2012, he joined SK Hynix Memory Solutions (SKHMS) as a systems architect for solid-state drives. His research interests include coding theory, VLSI architectures and integrated circuit design for communication and storage systems as well as signal processing.

Chih-Hsiang Hsu received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2009 and 2011, respectively.

From 2011 to 2012, he was in MediaTek Corporation. He is currently in ASUS, working in the area of protocol stack of wireless communication.

Hsie-Chia Chang received the B.S., M.S., and Ph.D. degrees from National Chiao Tung University, Hsinchu, Taiwan, in 1995, 1997, and 2002, respectively, all in electronic engineering.

From 2002 to 2003, he was with OSP/DE1 in MediaTek Corporation, working in the area of decoding architectures for Combo single chip. In February 2003, he joined the faculty of the Electronics Engineering Department, National Chiao-Tung University, where he has been a Professor since August 2010. His research interests include algorithms and VLSI architectures in signal processing, especially for error control codes and crypto-systems. Recently, he has also committed himself to designing high code-rate ECC schemes for flash memory and multi-Gb/s chip implementations for wireless communications

Dr. Chang served as the Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS since 2012. He also served as Technique Program Committee (TPC) member for IEEE A-SSCC 2011, 2012, and 2013. He was the recipient of the Outstanding Youth Electrical Engineer Award from Chinese Institute of Electrical Engineering in 2010, and the Outstanding Youth Researcher Award from Taiwan IC Design Society in 2011.

Chen-Yi Lee (M'01) received the B.S. degree from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1982, and the M.S. and Ph.D. degrees from Katholieke University Leuven (KUL), Belgium, in 1986 and 1990, respectively, all in electrical engineering.

From 1986 to 1990, he was with IMEC/VSDM, working in the area of architecture synthesis for DSP. In February 1991, he joined the faculty of the Electronics Engineering Department, National Chiao Tung University, Hsinchu, Taiwan, where he is currently a Professor and Dean of Research and Development Office. His research interests mainly include VLSI algorithms and architectures for high-throughput DSP applications. He is also active in various aspects of high-speed networking, system-on-chip design technology, very low power designs, and multimedia signal processing. In these areas, he has published more than 180 papers and holds decades of patents.

Dr. Lee served as the Director of Chip Implementation Center (CIC), an organization for IC design promotion in Taiwan from August, 2000 to December, 2003, and the microelectronics program coordinator of Engineering Division under National Science Council of Taiwan from January, 2003 to December, 2005. He was the former IEEE CAS Taipei Chapter Chair.