Improved High Code-Rate Soft BCH Decoder Architectures With One Extra Error Compensation

Yi-Min Lin, Hsie-Chia Chang, and Chen-Yi Lee

***Abstract*— Compared with traditonal hard Bose-Chaudhuri-Hochquenghem (BCH) decoders, soft BCH decoders provide better error-correcting performance but much higher hardware complexity. In this brief, an improved soft BCH decoding algorithm is presented to achieve both competitive hardware complexity and better error-correcting performance by dealing with least reliable bits and compensating one extra error outside the least reliable set. For BCH (255, 239; 2) and (255, 231; 3) codes, our proposed soft BCH decoders can achieve up to 0.75-dB coding gain with one extra error compensation and 5% less complexity than the traditional hard BCH decoders.**

***Index Terms*— Bose-Chaudhuri-Hochquenghem (BCH) codes, error-correction coding, soft decoding.**

I. Introduction

Bose-Chaudhuri-Hocquenghen (BCH) [1] codes are popular in storage devices and communication systems, such as Flash memories, DMB-T, DVB-T2, and DVB-S2 systems. While operating under GF(2^m), an $(N, K; t)$ BCH code has a error-correcting capability t with block length of N bits and information length of K bits, where $N - K \leq m \times t$.

In general, soft-decision maximum likelihood decoding (MLD) can provide around 3-dB coding gain for the same codes as compared with hard-decision decoding algorithms. However, MLD algorithm requires extreme computation complexity and is impractical for hardware implementation. Therefore, suboptimal soft decoding algorithms of error-control codes become popular and have aroused many research interests in BCH decoding [2]–[8]. Forney [2] developed the generalized-minimum-distance (GMD) algorithm, which uses soft information to generate test sequences for several hard BCH decoders to form a list of candidate code words and choose the most likely one from the candidate list. With a similar concept, the Chase, Modified GMD, and Chase–GMD algorithms [3]–[5] are also widely used to efficiently generate the candidate list and have been applied in many applications. In addition, a suboptimum maximum *a posteriori* (MAP) algorithm [6] with a Hamming SISO decoder, and the adaptive belief propagation algorithm were proposed for soft BCH decoding in 2005 and 2008, respectively [7], [8].

The hardware complexity and the storage requirement of a soft BCH decoder are generally much higher than that of a hard BCH decoder [2]–[8]. On the other hand, an error magnitude solver-based soft BCH decoding method that collects and deals with the least reliable bits instead of the entire code word was developed to achieve lower complexity decoders in 1997 [9]. There is a total of $2t$-chosen least reliable bits and their corresponding error magnitudes are calculated to determine the error locations in these $2t$ positions. Due to the limited possible error locations, it can provide lower complexity than other soft decoders, even lower complexity than the traditional hard decoder. However, this kind of soft decoder corrects the errors only when all actual error locations are collected in the limited possible locations. The decoder is unable to solve any error even though only one error occurred outside these locations. The hardware complexity is improved but the error-correcting performance highly depends on the reliability of the input signals. As a result, the trade-off between error-correcting performance and hardware complexity is a bottleneck for the soft BCH decoders.

In this brief, the proposed soft decoding algorithm has a concept similar to [9], but has one extra error compensation to enhance the correcting performance for the high code-rate BCH decoders. Consequently, in contrast to the conventional hard BCH decoders, the proposed soft BCH decoders achieve better performance by compensating one extra error outside the least reliable set and provide comparable hardware complexity by dealing with the least reliable bits. Moreover, the conventional BCH decoder contains three major blocks: syndrome calculator, key equation solver, and Chien search. For long-block-length BCH decoders, the decoding latency is dominated by the syndrome calculator and the Chien search. Unlike the conventional algorithms using parallel Chien search to enhance throughput, an error-locator evaluator is proposed to eliminate Chien search procedure for higher throughput [10].

This brief is organized as follows. Section II describes the proposed soft BCH decoding algorithm. The proposed architecture and comparison between hard and soft decoders are presented in Section III. Based on the proposed method, Section IV demonstrates the implementation results of the soft BCH decoders. Finally, we conclude this brief in Section V.

II. Proposed Soft BCH Decoding Algorithm

The proposed soft BCH decoder includes three major blocks: syndrome calculator, error-locator evaluator, and compensation error magnitude solver (CEMS) [11]. As compared with [9], the proposed soft BCH decoder enhances the error-correcting performance by compensating one extra error outside the least reliable locations while maintaining the low-complexity property.

A. Proposed Decoding Scheme

In the proposed decoding scheme, the reliability values are fed into the soft decoder and the received polynomial $R(x)$ are generated by inverting their sign bits in the BPSK modulation. The syndrome polynomial $S(x) = S_1 + S_2x^1 + \cdots + S_{2t}x^{2t-1}$ is expressed as

$$S_j = R(\alpha^j) = \sum_{i=1}^{\upsilon}(\alpha^j)^{e_i} = \sum_{i=1}^{\upsilon}(\beta_{e_i})^j \quad \text{for j} = 1 \sim 2\text{t} \qquad (1)$$

where α is the primitive element over GF(2^m) and υ is the number of actual errors. Notice that e_i is the ith actual error location and $\beta_{e_i} = \alpha^{e_i}$ indicates the corresponding error locator. With the soft inputs, the decoder chooses $2t$ least reliable inputs and evaluates their corresponding error locators to form the error-locator set $\mathcal{B} = [\beta_{l_1}, \beta_{l_2}, \ldots, \beta_{l_{2t}}]^T$. The error-location set, $L = [l_1, l_2, \ldots, l_{2t}]^T$, can be obtained in accordance with $\mathcal{B}$, because β_{l_i} is the error locator of the l_ith location and $\beta_{l_i} = \alpha^{l_i}$. In BCH codes, if the l_ith location is the exact error location, the error magnitude γ_i is equal to 1; otherwise, γ_i is equal to 0. The error-magnitude set $\Gamma = [\gamma_1, \gamma_2, \ldots, \gamma_{2t}]^T$ is defined as the error magnitude in accordance with L, and is valid only if it is a binary vector.

Manuscript received July 2, 2012; revised July 17, 2012; accepted October 18, 2012. Date of publication December 21, 2012; date of current version September 23, 2013. This work was supported in part by the National Science Council under Grant NSC 101-2628-E-009-013-MY3, and the Ministry of Economic Affairs of Taiwan under Grant 100-EC-17-A-01-S1-124.

Y.-M. Lin is with the Electrical Engineering Department, University of California, Los Angeles CA 90005 USA and also with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu 30050, Taiwan (e-mail: ymlin.mail@gmail.com).

H.-C. Chang and C.-Y. Lee are with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu 30050, Taiwan (e-mail: hcchang@mail.nctu.edu.tw; 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/TVLSI.2012.2227847

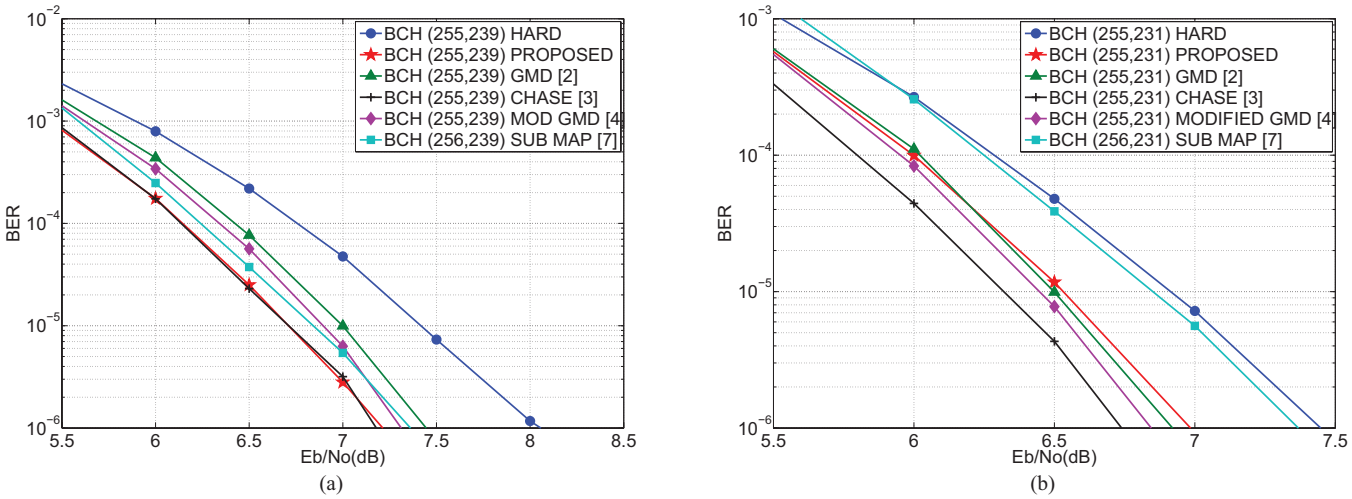

Fig. 1. Simulation results for the proposed soft BCH decoders with 255-b code word length. (a) BCH (255, 239; 2). (b) BCH (255, 231; 3).

The relation between $\mathcal{B}$, Γ, and the syndrome vector, $S = [S_1, S_2, \ldots, S_{2t}]^T$, can be formulated as

$$\begin{bmatrix} \beta_{l_1} & \beta_{l_2} & \cdots & \beta_{l_{2t}} \\ \beta_{l_1}^2 & \beta_{l_2}^2 & \cdots & \beta_{l_{2t}}^2 \\ \vdots & \vdots & \cdots & \vdots \\ \beta_{l_1}^{2t} & \beta_{l_2}^{2t} & \cdots & \beta_{l_{2t}}^{2t} \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{2t} \end{bmatrix} = \begin{bmatrix} \mathrm{S}_1 \\ \mathrm{S}_2 \\ \vdots \\ \mathrm{S}_{2t} \end{bmatrix} \tag{2}$$

where the $2t \times 2t$ matrix in (2) is defined as error-locator matrix $\mathbf{B}$. To represent the difference between S and the product of $\mathbf{B}$ and Γ, a discrepancy vector $\Delta = [\delta_1, \delta_2, \ldots, \delta_{2t}]^T$ is defined as

$$\Delta = \mathbf{B} \times \Gamma + S. \tag{3}$$

Notice that both the operations in (2) and (3) are under GF(2^m). It is evident that if all errors are located within the location set L, the valid Γ can be determined to make Δ be a zero vector. Otherwise, Γ is calculated as a nonbinary vector and this decoding approach fails to correct errors. If any error occurred outside L, the decoder is unable to solve any error, resulting in the lower correcting performance. However, the error-correcting ability can be enhanced by not only correcting errors located inside L but also correcting errors outside L. Under the analysis of Δ, an error located at l_{miss} and outside L induces $\Delta = [\beta_{l_{\text{miss}}}, \beta_{l_{\text{miss}}}^2, \ldots, \beta_{l_{\text{miss}}}^{2t}]$, where $\beta_{l_{\text{miss}}} = \alpha^{l_{\text{miss}}}$. Notice that a geometrical progression is a sequence of numbers where each term can be formulated by multiplying the previous one by a common ratio. To improve the error-correcting ability, we can additionally check whether Δ has the property of a geometrical progression and make a compensation for finding the missing location l_{miss} from Δ. Accordingly, the proposed compensation soft BCH decoder can correct up to $2t + 1$ errors. Except for the l_{miss}, other error locations are l_i whose corresponding γ_i equals 1. The estimated code word polynomial $\hat{C}(x)$ can be obtained by inverting values at these error locations in the received polynomial $R(x)$. Notice that, the proposed soft decoding algorithm can be applied to the RS codes as well but the computation of Γ becomes much more complex because of the nonbinary characteristic.

B. CEMS Algorithm

The CEMS algorithm is applied to calculate Γ and Δ according to (2) and (3). The Gauss Elimination method is the most intuitive way to solve (2); however, it may provide invalid (nonbinary) error magnitude γ_i and its computation complexity is $O(n^3)$. In BCH codes, the valid error magnitude in Γ is a binary value. Accordingly, solving (2) and (3) can be formulated as into checking all combinations of γ_i over GF(2) instead of calculating exact error magnitudes. A $2t$-b counter is employed to execute a heuristic search for all binary combinations. Notice that $S_1^2 = S_2, S_2^2 = S_4, \ldots, S_t^2 = S_{2t}$ in BCH codes, the computation of the even part syndromes $(S_2, S_4, \ldots, S_{2t})$ can be eliminated. The odd syndromes vector $S_{\text{odd}} = [S_1, S_3, \ldots, S_{2t-1}]^T$ and the error-locator matrix with half rows, $\mathbf{B}_{\text{odd}}$, are defined to simplify (2) as

$$\begin{bmatrix} \beta_{l_1} & \beta_{l_2} & \cdots & \beta_{l_{2t}} \\ \beta_{l_1}^3 & \beta_{l_2}^3 & \cdots & \beta_{l_{2t}}^3 \\ \vdots & \vdots & \cdots & \vdots \\ \beta_{l_1}^{2t-1} & \beta_{l_2}^{2t-1} & \cdots & \beta_{l_{2t}}^{2t-1} \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{2t-1} \\ \gamma_{2t} \end{bmatrix} = \begin{bmatrix} \mathrm{S}_1 \\ \mathrm{S}_3 \\ \vdots \\ \mathrm{S}_{2t-1} \end{bmatrix}. \tag{4}$$

The modified discrepancy vector from (4) is

$$\begin{aligned} \Delta_{\text{odd}} &\triangleq [\delta_1, \delta_3, \ldots, \delta_{2t-1}]^T \\ &= \mathbf{B}_{\text{odd}} \times \Gamma + S_{\text{odd}}. \end{aligned} \tag{5}$$

Notice that only t rows in the $\mathbf{B}_{\text{odd}}$ and S_{odd} means only half the computations for calculating Δ_{odd} in contrast to calculating Δ, leading to significant computation reduction. The following steps illustrate the details of the proposed algorithm for CEMS.

A heuristic search for all binary combinations is completed by iteratively counting Γ value from 0 to 2^{2t-1}. At each iteration, the solver verifies whether or not Δ_{odd} becomes a geometrical progression. Once the geometrical progression check passes at certain Γ value, the corresponding error locations in L and l_{miss} can be found with Γ and Δ_{odd}.

C. Simulation Results

For the purpose of comparing with existing methods, our proposed designs are compared with the traditional hard decision, GMD [2], the 2-b flipping Chase [3], the modified GMD [4], and the two iterations suboptimum MAP [7] decoding algorithms. In all cases, the BPSK modulation and the AWGN channel are used and all the performances are compared at 10^{-5} BER.

The simulation results of 2-error-correcting and 3-error-correcting BCH codes with 255-b code word length are presented in Fig. 1(a) and (b), respectively. The proposed soft decoder can correct at least one random error and as many as $2t + 1$ errors if there are $2t$ errors

Algorithm 1 Compensation Error Magnitude Solver

Input : $\mathcal{B}$, S_{odd}, and $\Gamma = 0$

1) Construct $\mathbf{B}_{\text{odd}}$ based on $\mathcal{B}$
2) $\Delta_{\text{odd}} = \mathbf{B}_{\text{odd}} \times \Gamma + S_{\text{odd}}$
3) if Δ_{odd} is a zero vector
   l_{miss} = NULL
   Go to OUTPUT
   else if Δ_{odd} is a geometrical progression
   Go to Step 4)
   else
   if $\Gamma == 2^{2t} - 1$
   Failed Decoding !!!
   else
   $\Gamma = \Gamma + 1$
   Go to Step 2)
4) Find l_{miss} according to the relation: $\delta_1 = \alpha^{l_{\text{miss}}}$

Output : Γ and l_{miss}

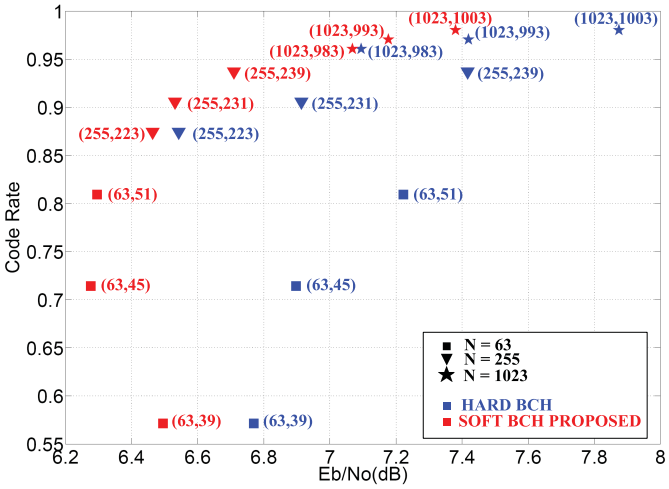

Fig. 2. Simulation results of hard and soft BCH decoders.

located at the least reliable positions (LRPs). For the GMD/modified GMD decoder, which can correct at least t random errors, at most $2t/2t+1$ errors can be corrected, if all of them are located at the $2t/2t+1$ LRPs. The 2-b flipping Chase decoder can correct at least t random errors and as many as $t+2$ errors, if there are two errors located at the two LRPs. The proposed soft BCH (255, 239; 2) decoder outperforms hard BCH (255, 239; 2) decoder by 0.75 dB. In addition, our proposed decoder is comparable with the Chase decoder while providing an improvement of 0.13–0.35 dB over other soft decoders. For BCH (255, 231; 3) codes, a coding gain of 0.4 dB can be achieved by our soft BCH decoder when compared with the hard decoder. In contrast to the GMD, the modified GMD, and the Chase decoders, our soft decoder has 0.03–0.22 dB performance loss. Notice that the two errors outside the limited possible locations are sufficient to make the proposed algorithm fail, whereas such error patterns will be corrected by other soft decoders. Therefore, the proposed soft decoder cannot offer a better performance than other soft decoders for a high SNR region (BER $< 10^{-8}$). However, these soft decoders demand several times the hardware complexity of our proposed soft decoder.

Fig. 2 demonstrates the performances of the hard and soft BCH decoders with a code word length of 63, 255, and 1023 b, respectively.

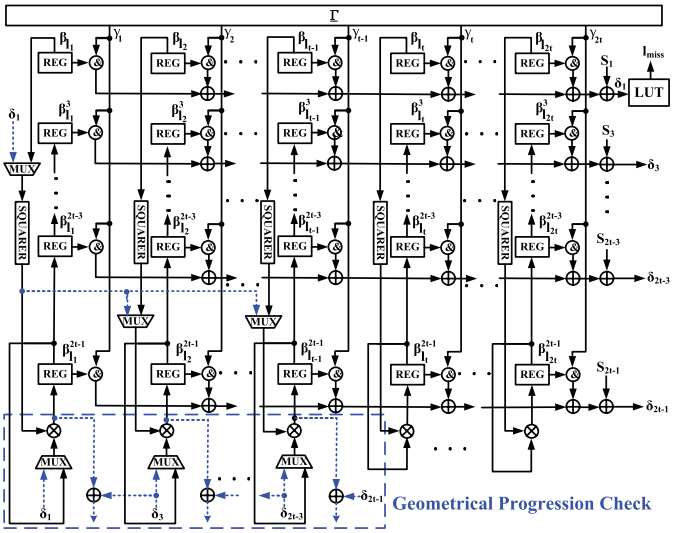

Fig. 3. Compensation error-magnitude solver.

Our proposed soft decoders can outperform hard decoders for error-correcting capability $t = 2-4$. Notice that, the achieved coding gain decreases in accordance with increasing t.

III. VLSI Architecture for Proposed Soft BCH Decoders

As mentioned in Section II, the proposed soft decoder includes three major blocks. In [11], we discussed an efficient architecture for each block. However, in contrast to the CEMS presented in [11], the CEMS proposed in this brief will provide a new sharing architecture, leading to $(t-1)$-multiplier hardware reduction. The architecture comparison between the hard and soft BCH decoders is demonstrated in the end of this section.

A. Compensation Error Magnitude Solver

Based on algorithm A, Fig. 3 illustrates the CEMS architecture to evaluate $\Delta_{\text{odd}} = \mathbf{B}_{\text{odd}} \times \Gamma + S_{\text{odd}}$ with S_{odd} and $\mathcal{B}$. The solid lines are the data flow of the $\mathbf{B}_{\text{odd}}$ matrix construction procedure while the dash lines are the data flow of the geometrical progression check procedure. There are $2t^2$ registers for storing all entries in $\mathbf{B}_{\text{odd}}$ matrix. In the ith column, the initial value of the first row register is set as β_{l_i} so that the output of the squarer will always be $\beta_{l_i}^2$. The tth row register is also initially set as β_{l_i} and iteratively multiplied by $\beta_{l_i}^2$ to generate $\beta_{l_i}^{2j+1}$ for operating cycles $j = 1 \sim (t-1)$. Consequently, the register values of the ith column form a geometric progression with the common ratio $\beta_{l_i}^2$ after $t-1$ cycles. The $\mathbf{B}_{\text{odd}}$ matrix is calculated with a total of only the $2t$ multipliers and the $2t$ squarers in $t-1$ cycles. These registers will hold their values in matrix multiplication procedure: $\Delta_{\text{odd}} = \mathbf{B}_{\text{odd}} \times \Gamma + S_{\text{odd}}$.

Both matrix multiplication and geometrical progression check are evaluated simultaneously in the following 2^{2t} cycles. A heuristic search for all binary combinations is completed iteratively to count Γ value from 0 to 2^{2t-1}. At each iteration, the $\beta_{l_i}^j$ value stored in the register will be operated with γ_i to generate the modified discrepancy vector Δ_{odd}. Then, the solver verifies whether Δ_{odd} is a geometrical progression or not. In the geometrical progression check procedure, δ_1 passes through a squarer to generate δ_1^2, which is multiplied with each δ_i value for being compared with δ_{i+2}. If Δ_{odd} is a geometrical progression, then $\delta_i \times \delta_1^2 = \delta_{i+2}$ for $i = 1, 3, \ldots, 2t-3$. The CEMS applies $t-1$ multipliers and one squarer to check this

TABLE I
COMPARISON TABLE FOR AN $(N, K; t)$ BCH CODE

	$(N, K; t)$ Hard BCH With iBM	$(N, K; t)$ Hard BCH With SiBM	$(N, K; t)$ Soft BCH With CEMS	(255, 239; 2) Hard BCH With iBM	(255, 239; 2) Hard BCH With SiBM	(255, 239; 2) Soft BCH With CEMS
Register	$5t+2$	$7t+2$	$2t^2+5t$	12	16	18
Mux	$t+1$	$2t$	$8t-2$	3	4	14
Mult	$3t+3$	$4t$	$2t$	9	8	4
Constant Mult	$3t$	$3t$	$t+1$	6	6	3
Squarer	0	0	$2t$	0	0	4
Lookup table (LUT)	0	0	1	0	0	1
RAM (bit)	N	N	N	255	255	255
Latency	$2N+2t$	$2N+t$	$N+2^{2t}+t-1$	514	512	272
Normalized* Complexity	$54t+42$	$72t+5$	$5t^2+49t+26.5$	150	149	144.5

* The normalized complexity is in terms of number of 8-b 2-to-1 multiplexer.

TABLE II
SUMMARY OF IMPLEMENTATION RESULTS

	BCH (255, 239) HARD	BCH (255, 239) HARD	BCH (255, 239) SOFT	BCH (255, 231) HARD	BCH (255, 231) HARD	BCH (255, 231) SOFT
Technology	90 nm	90 nm	90 nm	90 nm	90 nm	90 nm
Architecture	iBM [12]	SiBM [13]	CEMS	iBM [12]	SiBM [13]	CEMS
Operation Frequency	400 MHz (Post Layout)	500 MHz (Post Layout)	400 MHz (Post Layout)	360 MHz (Post Layout)	500 MHz (Post Layout)	360 MHz (Post Layout)
Core Area	14 400 μm^2	13 225 μm^2	13 225 μm^2	21 025 μm^2	21 025 μm^2	21 025μ m^2
Gate Count	4.4 K	4.3 K	4.2 K	6.8 K	6.6 K	6.7 K
Latency	514	512	272	516	513	321
Throughput	186 Mb/s	233.4 Mb/s	351.5 Mb/s	161.2 Mb/s	225 Mb/s	259.1 Mb/s
Coding Gain @10^{-5} BER	–	–	0.75 dB	–	–	0.4 dB

relation, and employs LUT to obtain l_{miss} according to $\delta_1 = \alpha^{l_{\text{miss}}}$. However, the geometrical progression check is processed after the $\mathbf{B}_{\text{odd}}$ matrix construction, the squarer and multipliers can be shared, leading to a total of only the $2t$ multipliers and the $2t$ squarers in the proposed CEMS architecture. The critical path of the matrix multiplication procedure is $(T_{\text{and}} + 2t \times T_{\text{xor}})$ for generating Δ_{odd} while that of the geometrical progression check procedure is $(T_{\text{xor}} + 2T_{\text{mux}} + T_{\text{sq}} + T_{\text{mult}})$ for using δ_1 to verify the relation $\delta_i \times \delta_1^2 = \delta_{i+2}$ with $i = 1, 3, \ldots, 2t-3$. Notice that $T_{\text{and}}, T_{\text{xor}}, T_{\text{mux}}, T_{\text{sq}}$, and T_{mult} represent the critical path of AND gate, XOR gate, multiplexer, squarer, and multiplier, respectively. Consequently, the critical path of CEMS is $(T_{\text{and}} + (2t+1) \times T_{\text{xor}} + 2 \times T_{\text{mux}} + T_{\text{sq}} + T_{\text{mult}})$.

B. Architecture Comparison

The architectures of the hard and soft BCH decoders are compared in Table I. The proposed soft BCH decoder is designed with the CEMS approach, whereas the hard BCH decoders are designed with inversionless Berlekamp-Massey (iBM) algorithm [12] and simplified iBM (SiBM) algorithm [13], respectively. A total of the $2t$ multipliers, the $2t$ squarers, and one LUT are utilized in the soft BCH decoder. The registers in the first row of $\mathbf{B}_{\text{odd}}$ matrix in the CMES is applied to store the error-locator set $\mathcal{B}$, which is also stored in the registers of the error-locator evaluator. Therefore, these registers can be shared, resulting in a total of $2t^2-2t$ registers used in the CEMS. In addition, the syndrome calculator and the error-locator evaluator take N clock cycles simultaneously in the decoding process and the CEMS takes $2^{2t}+t-1$ clock cycles.

In finite field operations, a multiplier is more complex than a register and a multiplexer. Due to fewer multipliers, the proposed soft BCH decoder, with more registers and multiplexers as well as an additional LUT, has similar hardware complexity when compared with the hard BCH decoders with iBM and SiBM algorithms. According to the synthesis results in CMOS 90-nm technology, the complexity ratio over GF(2^8) among each 8-b 2-to-1 multiplexer, squarer, constant multiplier, 8-b register, multiplier, and LUT is 1:1.5:1.5:2.5:12:27. The normalized complexity of the soft BCH decoder is around $(5t^2+49t+26.5)$ 8-b 2-to-1 multiplexers, whereas that of the hard BCH decoder with iBM/SiBM algorithms is $(54t+42)/(72t+5)$ 8-b 2-to-1 multiplexers, respectively. For the high-code-rate BCH codes, the error-correcting capability t is small, implying that the proposed soft decoder can provide similar hardware complexity as hard decoders even though the complexity of hard and soft decoders is linear and quadratic to t, respectively.

Based on Table I, the effect of error-correcting capability t to the hardware complexity and latency can be illustrated. Our soft decoder can provide the competitive hardware complexity when t equals 2 or 3. For example, the normalized complexity of the soft BCH (255, 239; 2) decoder is around 144.5 8-b 2-to-1 multiplexers whereas that of the hard BCH (255, 239; 2) decoder with iBM/SiBM algorithms is 150/149 8-b 2-to-1 multiplexers, respectively. Moreover, the proposed soft decoder searches for error locations at error-locator evaluator procedure, leading to less than 62.6% latency compared with that from the hard BCH decoders when t is smaller than 4.

IV. IMPLEMENTATION RESULTS

In Table II, the BCH decoders with hard and soft-decision methods are implemented for BCH (255, 239; 2) and BCH (255, 231; 3) codes. The hard BCH decoders solve key equation with iBM and SiBM algorithms, respectively, as well as evaluate error locations with Chien search while the soft BCH decoder is designed with CEMS.

The implementation results reveal that the proposed soft BCH (255, 239; 2) and (255, 231; 3) decoders can reach 4.2 and 6.7 K gate count with 400 and 360 MHz operation frequency, respectively, in standard CMOS 90-nm technology, which are similar to that provided by the hard BCH decoders. Although the hard BCH decoders with SiBM can operate under 500-MHz frequency, our proposed soft decoders provide better throughput because of lower latency. Compared with the traditional hard BCH decoders, the proposed soft BCH decoders computing error locations without Chien search achieve 1.6–1.9 times throughput enhancement.

V. Conclusion

This brief provided the improved soft decoders with one extra error compensation. Compared with the conventional hard BCH decoder, our proposed soft BCH decoders not only achieved better error-correcting performance but also provided competitive hardware complexity. The decoders with soft information can reduce hardware complexity by focusing on the least reliable bits. Meanwhile, the error-correcting ability is improved with one extra error compensation. Experimental results show that the proposed soft BCH (255, 239; 2) and BCH (255, 231; 3) decoders can obtain 0.75- and 0.4-dB coding gain, respectively, over the corresponding hard BCH decoders at 10^{-5} BER. According to postlayout simulation in 90-nm CMOS technology, the proposed soft decoders can achieve up to 1.9 times throughput enhancement and 5% gate count reduction as compared with the traditional hard BCH decoders.

Acknowledgment

The authors would like to thank Prof. C.-C. Chung for layout assistance. The authors would also like to thank UMC and Chip Implementation Center for providing standard CMOS technology data and computer aided tools tools.

References

[1] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.

[2] G. D. Forney, "Generalized minimum distance decoding," *IEEE Trans. Inf. Theory*, vol. 12, no. 2, pp. 125–131, Apr. 1966.

[3] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inf. Theory*, vol. 18, no. 1, pp. 170–182, Jan. 1972.

[4] G. D. J. Forney and A. Vardy, "Generalized minimum-distance decoding of euclidean-space codes and lattices," *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1992–2026, Nov. 1996.

[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] C. Hartmann and L. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Inform. Theory*, vol. 22, no. 5, pp. 514–517, Sep. 1976.

[7] F. Therattil and A. Thangaraj, "A low-complexity soft-decision decoder for extended BCH and RS-like codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Sep. 2005, pp. 1320–1324.

[8] M. Baldi and F. Chiaraluce, "A simple scheme for belief propagation decoding of BCH and RS codes in multimedia transmissions," in *Int. J. Dig. Multimed. Broadcast. Conf.*, Apr. 2008, pp. 1–12.

[9] W. J. ReidIII, L. L. Joiner, and J. J. Komo, "Soft decision decoding of BCH codes using error magnitudes," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 1997, p. 303.

[10] Y.-M. Lin, C.-L. Chen, H.-C. Chang, and C.-Y. Lee, "A 26.9 K 314.5 Mb/s soft (32400, 32208) BCH decoder chip for DVB-S2 system," *IEEE J. Solid-State Circuits*, vol. 45, no. 11, pp. 2330–2340, Nov. 2010.

[11] Y.-M. Lin, H.-C. Chang, and C.-Y. Lee, "An improved soft BCH decoder with one extra error compensation," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 3941–3944.

[12] I. S. Reed, M. T. Shih, and T. K. Truong, "VLSI design of inverse-free Berlekamp-Massey algorithm," in *Proc. IEEE Inst. Elect. Eng*, Sep. 1991, pp. 295–298.

[13] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in *Proc. IEEE Workshop Signal Process. Syst.*, Oct. 2006, pp. 303–308.