This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2581587, IEEE Transactions on Circuits and Systems II: Express Briefs

An Efficient Eligible Error Locator Polynomial Searching Algorithm and Hardware Architecture for One-Pass Chase BCH Codes Decoding

Nan Zheng, *Student Member, IEEE*, and Pinaki Mazumder, *Fellow, IEEE*

***Abstract*—In numerous memory and communication systems, Bose-Chaudhuri-Hocquenghem (BCH) codes are widely employed to enhance reliability. One-pass Chase soft-decision decoding algorithm for BCH code was previously proposed to achieve significant performance improvement over traditional hard-decision decoding while not increasing too much computational complexity. The bottleneck in a conventional one-pass Chase decoding is the procedure of judging whether an obtained error locator polynomial is valid. In this paper, a novel algorithm that can efficiently verify eligibility of each generated error locator polynomial is proposed. The problem is first reformulated as a polynomial modulo problem, where repeated squaring can be employed for further simplification. In order to decrease the critical path delay and hardware complexity, an efficient polynomial division algorithm based on polynomial inversion is also proposed. In addition, a VLSI architecture for the proposed algorithm is presented. The implemented results show that the proposed eligibility checking algorithm reduces the gate counts to only 12% of a conventional polynomial selection algorithm without introducing any speed penalty. The projected area reduction achieved in a complete one-pass Chase decoder is approximately 75%. In addition, post-layout simulation shows that the proposed algorithm is 20 times more power-efficient than the conventional method.**

***Index Terms*—BCH codes, error correction code, one-pass Chase, polynomial inversion, VLSI.**

I. Introduction

Bose-Chaudhuri-Hocquenghem (BCH) codes are one of the most popular error correction codes (ECC) employed in numerous memory and communication systems due to their relatively simple decoding complexity and outstanding error correction capability. Conventional BCH codes can correct up to $t = d_{min}/2$ errors, where d_{min} stands for the minimum code distance. Hard-decision decoding (HDD) procedure for BCH codes is well developed. The Berlekamp's algorithm is usually employed to find error locator polynomials, and the Chien search is then utilized to locate each error. To better utilize soft information or confidence of each bit before decoding, Chase proposed a soft-decision decoding (SDD) algorithm that first flips η least reliable bits before a conventional HDD [1]. By utilizing soft information, Chase II algorithm can correct up to $t + \eta$ errors.

The most straightforward and the widely adopted way to implement Chase decoding in hardware is to use existing BCH HDD circuit iteratively with a control circuitry that generates different testing patterns [2, 3]. This way, however, becomes prohibitively expensive when η is large since the computational time grows exponentially with η. To circumvent this, one-pass Chase decoding algorithm was proposed in [4]. The original work is aimed for Reed-Solomon (RS) codes, but it can be easily adapted for BCH codes. The Berlekamp's algorithm needs only to be applied once to obtain one starting error locator polynomial. Other error locator polynomials corresponding to different error patterns can then be derived easily from the starting polynomial by using polynomial update algorithm outlined in [4]. This is a significant result since the algorithm turns the exponentially scaled Chase decoding algorithm into a linearly scaled algorithm, making soft-decision Chase decoding more attractive.

To implement the one-pass Chase decoding algorithm, hardware architecture was discussed in [5]. Zeros in the error locator polynomial corresponding to the flipped bits were taken out to maintain the order of the polynomial, reducing computational efforts. In that work, it was stated that most area and power consumption were spent on the highly parallel Chien search that runs as fast as the polynomial update block. An interpolation based Chase decoding algorithm was developed in [6] to avoid the expensive parallel Chien search. A 2.3 times higher efficiency was reported in that work.

To better accommodate one-pass Chase decoding algorithm, an eligibility verification algorithm is proposed in this paper. The algorithm checks if the obtained error locator polynomial from the polynomial update algorithm or the Berlekamp's algorithm is able to generate a correct error pattern such that the corrected code is a valid code word. The main motivation of the proposed algorithm is that an invalid polynomial can be easily detected without actually finding out all roots of that polynomial. The problem of checking the eligibility is first converted into a problem of calculating polynomial modulus. Calculation of modulus can then be conveniently solved by repeated squaring. In addition, to further reduce the computation complexity as well as the critical path delay in a hardware implementation, a polynomial inversion algorithm is

Manuscript received. This work is supported by National Science Foundation under Grant CCF1421467

The authors are with University of Michigan, Ann Arbor, MI 48105 USA (e-mail:zhengn@umich.edu; mazum@umich.edu)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2581587, IEEE Transactions on Circuits and Systems II: Express Briefs

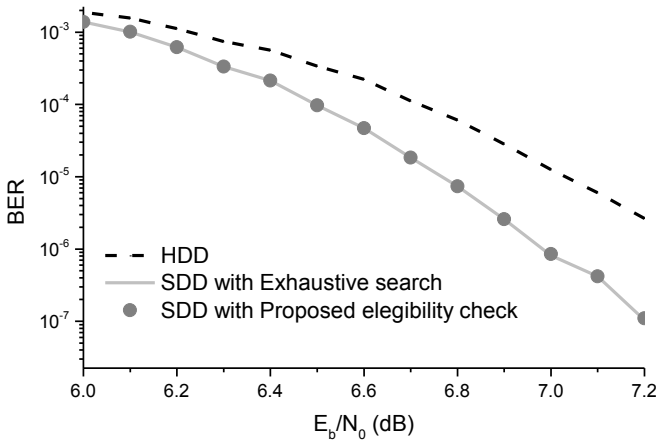

Fig. 1. Comparison of error correction performance of HDD, SDD with exhaustive polynomial search, and SDD with the proposed eligibility checking algorithm. The BCH code used here is a (4200, 4096) code over GF(2^{13}).

proposed. A Hardware architecture and complexity analysis for the proposed algorithm are provided in Section III. In Section IV, a design example is presented. Compared to the conventional exhaustive searching, the proposed algorithm achieves a reduction of 88% in gate counts.

II. Algorithm

It has been shown in [7] that sufficient and necessary conditions for an error locator polynomial to locate errors such that a legal (not necessarily correct) word can be recovered are as following:

1) The error locator polynomial $\Lambda(x)$ has exactly e distinct non-zero roots in GF(2^m).
2) $L_\Lambda = e$, where L_Λ is the length of a linear feedback shift register (LFSR) described by $\Lambda(x)$.

It is also known that $d_\Lambda \le L_\Lambda$ [8], where d_Λ is the degree of the polynomial $\Lambda(x)$. Consequently, the process of verifying an error locator polynomial can be divided into two cases:

1) If $d_\Lambda \neq L_\Lambda$, then the error locator polynomial is not a valid one.
2) If $d_\Lambda = L_\Lambda$, then check whether the number of distinct non-zero roots of $\Lambda(x)$ is equal to d_Λ. If they are not equal, then the error locator polynomial is not a valid one.

The condition $d_\Lambda = L_\Lambda$ can be easily checked by identifying the location of the first non-zero coefficient in $\Lambda(x)$. Furthermore, it is found in simulation that this condition is satisfied in most of the time even when the error locator polynomial is not a correct one. Therefore, counting the number of roots that $\Lambda(x)$ has is the key to determine whether the obtained error locator polynomial is valid.

Zero root in $\Lambda(x)$ can be identified by checking whether Λ_0 is zero. $\Lambda(x)$ with a zero root is discarded right away without further processing. Therefore, in the remainder of this paper, it is assumed that $\Lambda(x)$ does not have a root that is equal to zero.

An auxiliary polynomial $d(x)$ is defined according to (1), where the operator $gcd(a, b)$ stands for finding the greatest common divisor of a and b. Since $x^{2^m} - x$ has all the elements in GF(2^m) as its roots [9], it can be shown that the degree of $d(x)$ is equal to the number of roots that $\Lambda(x)$ has.

$$d(x) := gcd\left(x^{2^m} - x, \Lambda(x)\right) \tag{1}$$

Euclidean's algorithm can be employed here to obtain a further simplified expression, as shown in (2).

$$d(x) = gcd\left(\left(x^{2^m} - x\right) \bmod \Lambda(x), \Lambda(x)\right) \tag{2}$$

Following (2), it can be proven that the sufficient and necessary condition for $\Lambda(x)$ to have d_Λ distinct non-zero roots in GF(2^m) is

$$x^{2^m} \bmod \Lambda(x) = x \tag{3}$$

(3) is too expensive to be computed directly when m is large. Fortunately, squaring and multiply [10] can be utilized here to save a significant amount of computational labors. In addition, since 2^m is a power of 2, what we really need is just repeated squaring (that is, not even multiply). More specifically, we can calculate x^{2^m} iteratively as following. Starting from the trivial case

$$x^{2^{\lfloor \log_2(d_\Lambda - 1)\rfloor}} \bmod \Lambda(x) = x^{2^{\lfloor \log_2(d_\Lambda - 1)\rfloor}} \tag{4}$$

We then can compute $x^{2^{i+1}} \bmod \Lambda(x)$ from $x^{2^i} \bmod \Lambda(x)$ as

$$x^{2^{i+1}} \bmod \Lambda(x) = f_i(x) \bmod \Lambda(x) \tag{5}$$

where

$$f_i(x) = \left(x^{2^i} \bmod \Lambda(x)\right)^2 \tag{6}$$

By doing this, only $m - \lfloor \log_2(d_\Lambda - 1)\rfloor$ polynomial modulo operations are needed to compute $x^{2^m} \bmod \Lambda(x)$. Each polynomial modulo operation is at most of order t.

To carry out modulo operation, an old school long division can be employed. Each modulo operation takes about d_Λ^2 multiplications. One problem with this straightforward implementation is that the critical path is long, as will be shown in Section III.

Considering that divisors in all modulo operations are $\Lambda(x)$ (see (2) – (6)), it is worth spending some efforts on converting $\Lambda(x)$ into a form with which the division in the following stages can be performed more efficiently. Inspired by the algorithm in [11], we propose the following polynomial inversion algorithm to help improve the efficiency and critical path delay of the polynomial division.

Let $\Lambda^r(x)$ represent the polynomial with coefficients arranged in a reverse order of $\Lambda(x)$. That is, $\Lambda^r(x) = \Lambda_0 x^{d_\Lambda} + \Lambda_1 x^{d_\Lambda - 1} + \cdots + \Lambda_{d_\Lambda}$. Similar notations apply for other polynomials. Then it can be shown that the reverse quotient polynomial can be computed as

$$q_i^r(x) = \widehat{\Lambda}^r(x) f_i^r(x) \bmod x^{d_\Lambda - 1} \tag{7}$$

where $\widehat{\Lambda}^r(x)$ is defined as the inverse polynomial of $\Lambda^r(x)$ such that

$$\Lambda^r(x)\widehat{\Lambda}^r(x) = 1 \bmod x^{d_\Lambda - 1} \tag{8}$$

$\widehat{\Lambda}^r(x)$ can be conveniently computed through an iterative algorithm. Let $\widehat{\Lambda}_i^r(x)$ be the polynomial such that $\Lambda^r(x)\widehat{\Lambda}_i^r(x) = 1 \bmod x^i$, then it can be shown that by setting

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2581587, IEEE Transactions on Circuits and Systems II: Express Briefs

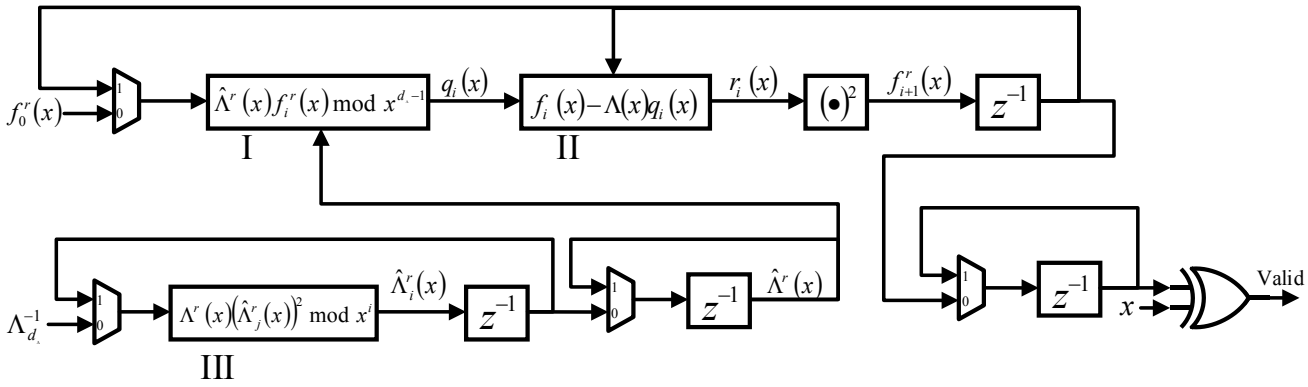

Fig. 2. A VLSI architecture of the proposed eligibility checking algorithm. Block I and II conduct multiply and modulo operations, and Block III conducts polynomial inversion operation.

TABLE I
SUMMARY OF HARDWARE COMPLEXITY OF THE PROPOSED ELIGIBILITY VERIFICATION CIRCUIT

	I	II	III
# of multiplier	$\frac{t(t+2)}{4}$	$\frac{(t-1)(t+2)}{2}$	$\frac{t}{2}$
# of adder	$\frac{t(t-2)}{4}$	$\frac{t(t-1)}{2}$	$\frac{t}{2}-1$
Critical path delay	$D_{mul}+D_{add}\times\log_2\left(\frac{t-1}{2}\right)$	$D_{mul}+D_{add}\times\log_2(t-1)$	$D_{mul}+D_{add}\times\log_2\left(\frac{t}{2}\right)$
# of cycles	$m-\lfloor\log_2(t-1)\rfloor$	$m-\lfloor\log_2(t-1)\rfloor$	t

the initial condition and iterative updating equation as (9) and (10), one can obtain $\widehat{\Lambda}^r(x)$.

$$\widehat{\Lambda}_1^r(x) = \Lambda_{d_\Lambda}^{-1} \tag{9}$$

$$\widehat{\Lambda}_i^r(x) = \Lambda^r(x)\left(\widehat{\Lambda}_j^r(x)\right)^2 \bmod x^i \tag{10}$$

where $j \geq \lceil i/2 \rceil$.

It is noted that in (10), only unknown coefficients need to be calculated. The process of computing $\widehat{\Lambda}^r(x)$ takes one inversion, $\frac{d_\Lambda-1}{2}$ squaring, $\frac{(d_\Lambda-1)^2}{4}+\frac{d_\Lambda-1}{2}$ multiplication and $\frac{(d_\Lambda-1)^2}{4}-\frac{d_\Lambda-1}{2}$ addition when $d_\Lambda-1$ is a power of two. As will be shown in Section III, computational complexity of deriving $\widehat{\Lambda}^r(x)$ is much less than the complexity of computing (5). Furthermore, pre-computing $\widehat{\Lambda}^r(x)$ can actually save efforts for computing (5).

One thing should be noted is that derivation of the proposed algorithm implicitly assumes that the BCH code is not shortened. Therefore, the proposed algorithm is not rigorously applicable to shortened BCH codes. It is theoretically possible for the error locator polynomial of a shortened BCH code to have the correct number of roots over $GF(2^m)$, yet one or more of these roots are not in the valid range. In this case, the proposed algorithm fails to detect the invalid error locator polynomial, whereas the exhaustive Chien search is still able to. Simulation, however, shows that probability of this malfunction is unnoticeably low. Therefore, we argue that the proposed algorithm can be applied to a shortened BCH code for practical purposes.

III. VLSI ARCHITECTURE

Even though the proposed algorithm can be used for a polynomial with an arbitrary degree, it is, in practice, enough to check polynomials with a degree of t. By doing this, the hardware complexity can be significantly reduced without noticeable performance degradation. Fig. 1 compares the performance achieved by a SDD employing the proposed eligibility verification algorithm with the performance of a SDD using direct polynomial search and the performance of a HDD. In the proposed eligibility checking algorithm, d_Λ is set to t. That is, the verification process only runs on polynomials with a degree of t. For polynomials with degrees less than t, they are adopted as valid solutions. It is shown in the figure that, the proposed eligibility verification algorithm can effectively identify valid error locator polynomials. Performance of the proposed algorithm is degraded neither by applying the algorithm to a shortened BCH code nor by only running verification on polynomials with degree of t.

A VLSI architecture of the proposed eligibility checking algorithm is shown in Fig. 2. There are three main blocks: block I and II are for polynomial multiplication and block III is used for polynomial inversion. Proper pipelining can be applied to locations marked with arrows in order to improve throughput of the system.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2581587, IEEE Transactions on Circuits and Systems II: Express Briefs

TABLE II
COMPARISON OF HARDWARE COMPLEXITY AND CRITICAL PATH DELAY

Algorithm	Sub-block	NAND2	Flip-Flops	Critical path delay
Proposed Algorithm w/ polynomial inversion	Polynomial Inversion	634	94	$D_{AND} + 6D_{XOR}$
	Polynomial Multiplication	39K	104	$2D_{AND} + 13D_{XOR}$
	Total	46K	453	$2D_{AND} + 13D_{XOR}$
Exhaustive polynomial search		416K	485	$6D_{AND} + 6D_{XOR}$

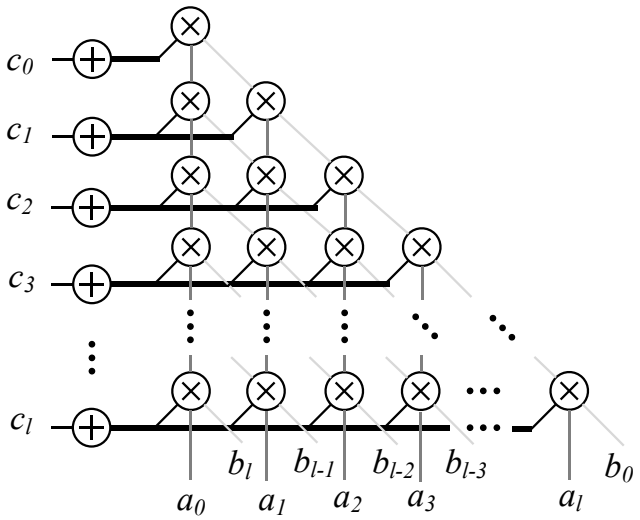

Fig. 3. Diagram of the proposed polynomial multiplication array. Same multiplicands b_i are shared along diagonals, and multipliers a_i are shared along columns. Multiplication results c_i are obtained by adding each row with XOR trees.

There are mainly three finite filed operations shown in Fig. 2 besides the standard multiplex, delay and compare operations. The first operation is subtracting polynomials in a finite field. This operation can be done by simply doing bitwise XOR operations on each coefficient of the two polynomials. The second operation is squaring a polynomial in a finite field, which can be achieved by squaring each coefficient of the polynomial. The third operation is to multiply and modulo polynomials. This is the main operation in the proposed algorithm. It can be formulated as a matrix-vector multiplication where the matrix is a Toeplitz matrix. Thanks to a unique property of Toeplitz matrices, the multiplication-and-modulo operation can be efficiently conducted by employing circuit shown in Fig. 3. In Fig. 3, diagonal multipliers share the same multiplicands b_i. Multipliers at the same columns share the same multipliers a_i. Products at the same row are then added by a XOR tree to get the final results c_i. Since a finite field multiplication can be expressed as a matrix-vector multiplication, matrices associated with the shared multiplicands only need to be calculated once and distributed along the diagonals, reducing gate counts and the critical path delay.

Hardware complexity of the main blocks in the proposed eligibility checking circuit is summarized in Table I. The proposed circuit has an area-latency product on the order of $\left(m - \log_2(t)\right)t^2$. This is much less than nt, the area-latency product of a conventional exhaustive search. Area-latency product is defined as the product of the number of multipliers and the number of clock cycles needed to complete the task. It serves as a quick estimation of how the complexity of the circuit grows with the size of the problem. The reason that only finite field multipliers are counted is that they dominate the area of the circuit. To give an immediate comparison, gate count ratio between the proposed algorithm and the conventional method is less than $(n-k)/n$, which is the redundancy ratio of an error correction code. The redundancy ratio in most memory system is much less than one. Even though the employed non-constant multiplier takes larger area than the constant multiplier employed in the Chien search, the saving on area-latency product is still significant, as will be shown in Section IV. In addition, the diagonal-sharing technique mentioned above also helps reduce the gate counts effectively.

In addition, compared to the straightforward implementation with long division that requires $\left(m - \log_2(t)\right)t^2$ multipliers, the proposed algorithm only needs approximately $\frac{3}{4}\left(m - \log_2(t)\right)t^2$ multipliers, reducing gate counts by roughly 25%. This saving is achieved by pre-computing the inverse polynomial $\widehat{\Lambda}^r(x)$.

IV. DESIGN EXAMPLE

In this section, the proposed eligibility checking circuit is implemented for a (4200, 4096) code over GF(2^{13}). Inversion of $\Lambda_{d_\Lambda}^{-1}$ and block III are pipelined with block II and III. The process of eligibility verification takes 11 clock cycles.

Comparisons of gate counts and critical path delay are summarized in Table II. In this design example, the finite field multiplier and squaring circuit proposed in [12] are used. The multiplier in [12] is not the optimal choice in terms of gate counts. It is adopted in our design example because of its simplicity. More sophisticated multipliers such as those in [13] can be employed to further reduce the gate counts. In the table, numbers of flip-flops are directly read out from the synthesized netlists, and combinational gate counts reported by the employed synthesis tool are converted to equivalent NAND2 gate counts for comparison.

As shown in the table, gate counts of the proposed eligibility checking circuit are only around 12% of the exhaustive Chien

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2016.2581587, IEEE Transactions on Circuits and Systems II: Express Briefs

search. This number can be further projected to estimate the overall saving on area of the decoder. In [5], it is shown that the polynomial searching block occupies an area that is 85% of the total area of the decoder. Therefore, it is estimated that the proposed eligibility verification circuit can reduce the area of the one-pass Chase decoder by 75% while having a similar performance. Furthermore, thanks to the polynomial inversion step, the critical path delay of the proposed circuit is reduced to a value similar to the one of the Chien search.

The proposed design and the conventional Chien search block are synthesized in 65 nm technology using Synopsys Design Compiler. The synthesized designs are then automatically placed and routed with Cadence Encounter. Areas of the obtained layouts are reported in Table III. Netlists obtained after place and route are simulated using Synopsys Finesim with extracted interconnect parasitics. Power consumption and critical path delay are simulated. The maximum clock frequency reported in Table III is calculated according to the simulated critical path delay with a 10% margin. As noted from Table III. The proposed eligibility checking circuit is 20 times more power-efficient than the conventional exhaustive Chien search. This number is larger than the area saving ratio. This is mainly because conventional Chien search has a larger activity factor.

TABLE III
SUMMARY OF THE PROPOSED DESIGN

	This work	**Conventional exhaustive Chien search**
Equivalent NAND2 gate count	49K	418K
Area after place and route	67,600 μm^2	640,000 μm^2
Power consumption @ 1.2 V & 400MHz	21.9 mW	408 mW
Maximum clock frequency	568 MHz	455 MHz

V. CONCLUSIONS

In this paper, we present a novel eligibility verification algorithm aiming to avoid the area and power consumption penalty incurred by the parallel Chien search in a conventional one-pass Chase soft-decision BCH decoder. The proposed algorithm can effectively check the correctness of a derived error locator polynomial by counting the number of roots it has. The root-counting problem is transformed into a polynomial modulo problem, which can be efficiently solved by repeated squaring. In addition, an iterative polynomial inversion algorithm is presented to reduce the area and the critical path delay. A hardware architecture for the proposed algorithms is also presented in this paper. Hardware complexity is carefully examined. A design example is implemented for a (4200, 4096) code over $GF(2^{13})$. The obtained gate counts and critical path delay are compared with a conventional design. Our newly proposed design achieves more than 88% area reduction while having a similar critical path delay. This translates into a 75% reduction in the overall decoder area. The proposed design is also placed and routed. The transistor-level simulation shows that, with a similar critical path delay compared to the conventional method, a 95% power saving is achieved.

ACKNOWLEDGEMENT

We would like to thank Dr. Jyrki Lahtonen for inspiring discussion on the eligibility verification algorithm. We are also grateful to Dr. Jiangli Zhu for discussion on the conventional implementation of a one-pass Chase soft-decision BCH decoder in [5]. We would also like to thank anonymous reviewers for their valuable comments.

REFERENCES

[1] D. Chase, "Class of algorithms for decoding block codes with channel measurement information," *Information Theory, IEEE Transactions on,* vol. 18, pp. 170-182, 1972.

[2] Y. Chia-Hsiang, H. Ting-Ying, L. Mao-Ruei, and U. Yeong-Luh, "A 5.4 μW Soft-Decision BCH Decoder for Wireless Body Area Networks," *Circuits and Systems I: Regular Papers, IEEE Transactions on,* vol. 61, pp. 2721-2729, 2014.

[3] L. Yi-Min, C. Chih-Lung, H.-C. Chang, and L. Chen-Yi, "A 26.9 K 314.5 Mb/s Soft (32400,32208) BCH Decoder Chip for DVB-S2 System," *Solid-State Circuits, IEEE Journal of,* vol. 45, pp. 2330-2340, 2010.

[4] W. Yingquan, "Fast Chase Decoding Algorithms and Architectures for Reed-Solomon Codes," *Information Theory, IEEE Transactions on,* vol. 58, pp. 109-129, 2012.

[5] Z. Xinmiao, Z. Jiangli, and W. Yingquan, "Efficient one-pass chase soft-decision BCH decoder for multi-level cell NAND flash memory," in *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, 2011, pp. 1-4.

[6] Z. Xinmiao, "An Efficient Interpolation-Based Chase BCH Decoder," *Circuits and Systems II: Express Briefs, IEEE Transactions on,* vol. 60, pp. 212-216, 2013.

[7] I. Giacomelli, "Improved Decoding Algorithms for Reed-Solomon Codes," *arXiv preprint arXiv:1310.2473,* 2013.

[8] W. Yingquan, "New List Decoding Algorithms for Reed-Solomon and BCH Codes," *Information Theory, IEEE Transactions on,* vol. 54, pp. 3611-3630, 2008.

[9] D. Costello and S. Lin, "Error control coding," *New Jersey,* 2004.

[10] J.-P. Deschamps, *Hardware implementation of finite-field arithmetic*: McGraw-Hill, Inc., 2009.

[11] M. Sudan, "Algebra and Computation," Cambidge, MA,2012. Available: http://people.csail.mit.edu/madhu/ST12/scribe/lect06.pdf

[12] Z. Xinmiao and K. K. Parhi, "Fast factorization architecture in soft-decision Reed-Solomon decoding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 13, pp. 413-426, 2005.

[13] A. Halbutogullari and C. K. Koc, "Mastrovito multiplier for general irreducible polynomials," *IEEE Transactions on Computers,* vol. 49, pp. 503-518, 2000.