Efficient Nested Key Equation Solver for Short Generalized Integrated Interleaved BCH Codes

Zhenshan Xie and Xinmiao Zhang
Dept. of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA
Email: {xie.855, zhang.8952}@osu.edu

***Abstract*—Generalized integrated interleaved (GII) codes can nest BCH sub-codewords to form stronger BCH codewords. They are among the best candidates for error correction in the new storage class memories (SCMs). However, SCMs require short codeword length and low redundancy. In this case, the nested key equation solver (KES), which is a key step in GII decoding, has a small number of iterations. The initialization and/or scalar pre-computation in previous nested KES designs have large area and may take even longer time than the iterations themselves. This paper proposes an efficient nested KES design for short GII-BCH codes. The polynomial updating is decomposed into two steps to reduce the critical path without requiring scalar pre-computation. Besides, the KES is reformulated to reduce the number of clock cycles without incurring any area overhead. For an example code over $GF(2^{10})$ that protects 2560 bits with 10% redundancy, the proposed design achieves at least 25% area reduction and 37% reduction on the area-time product averaged over the nested decoding rounds compared to prior efforts.**

I. Introduction

The new fast storage class memories (SCMs) may bring paradigm shifts to many systems, such as computer memory architecture, machine learning, and big data analytics. Error-correcting codes with hyper-speed decoding and excellent correction capability are essential to realizing the speed potential of SCMs. Generalized integrated interleaved (GII) codes [1], [2] that generate stronger BCH codewords by nesting short BCH sub-codewords are among the best candidates for SCMs.

GII decoding has two stages [2]. The first is the traditional BCH decoding on individual sub-words. A key equation solver (KES), such as the Berlekamp-Massey (BM) algorithm, takes the syndromes and computes the error locator polynomial. If some sub-words failed the decoding or were miscorrected, the multi-round second-stage nested decoding is activated to correct extra errors. In each round, higher-order sub-word syndromes are derived from those of the nested words. Then the nested KES is carried out to update the error locator polynomial and accordingly correct more errors.

Since the higher-order syndromes are not available in the beginning to initialize the discrepancy polynomial, the conventional reformulated inversionless (ri-)BM architecture [3] cannot be used to reduce the critical path of the nested KES. Re-initializing the discrepancy polynomial as in [4] before the nested KES requires many multiplier-adder trees. The critical path is reduced to two multipliers by incorporating higher-order syndromes iteratively into existing KES results in [5]–[7]. Although it can be further reduced by a half using the slow-down and re-timing techniques, the decoding of two sub-words needs to be interleaved. By pre-computing the combined polynomial scalars, the critical path is truly reduced to one multiplier in [8], [9] although 5 clock cycles are needed for the pre-computation.

Since SCMs require short codeword length and low redundancy, the error correction capabilities of the BCH codes involved in the GII codes and their differences are small. This means that the KES of each nested decoding round has a small number of iterations and the involved polynomials are short. In this case, the scalar pre-computations in [7], [9] targeting at longer codes require not only more clock cycles but also larger area compared to the nested KES iterations themselves.

This paper proposes an efficient nested KES design for short GII-BCH codes. The polynomial updating with long data path is decomposed into two steps so that the critical path is reduced to one multiplier without any scalar pre-computation. Shareable substructures are identified to reduce the area requirement. Additionally, by reformulating the nested KES algorithm, the order of the error locator and discrepancy polynomial updating is switched so that one more clock cycle is eliminated in the end. Efficient implementation architectures are also developed for the proposed nested KES algorithm. Although the proposed design needs two clock cycles for each nested KES iteration, it does not require expensive scalar pre-computation. For an example code over $GF(2^{10})$ that protects 2560 bits with 10% redundancy considered for SCMs, the proposed architecture achieves at least 25% area reduction and 37% improvement on the area-time product (ATP) averaged over the nested decoding rounds compared to prior work.

II. GII-BCH Decoding and Nested KES Algorithms

Let $\mathcal{C}_v \subseteq \cdots \subseteq \mathcal{C}_1 \subset \mathcal{C}_0$ be $v+1$ BCH codes defined over $GF(2^q)$ with error-correction capabilities $t_v \geq \cdots \geq t_1 > t_0$. A GII-BCH $[m,v]$ code with m length-n sub-codewords can be constructed from these BCH codes as [2]:

$$\mathcal{C} \triangleq \Big\{ [c_0(x), c_1(x), \cdots, c_{m-1}(x)] : c_i(x) \in \mathcal{C}_0, \\ \tilde{c}_l(x) = \sum_{i=0}^{m-1} \alpha^{il}(x) c_i(x) \in \mathcal{C}_{v-l}, 0 \leq l < v \Big\}, \tag{1}$$

This work was supported in part by Kioxia Corporation and by the National Science Foundation under Award No. 2011785.

Algorithm 1: Nested GII-BCH KES Algorithm

Input: $\Lambda_{even}^{(u)}(x)$, $\Lambda_{odd}^{(u)}(x)$, $B_{even}^{(u)}(x)$, $B_{odd}^{(u)}(x)$, $\hat{\Delta}_{even}^{(u)}(x)$, $\hat{\Theta}_{even}^{(u)}(x)$, $\gamma^{(u)}$, $k^{(u)}$ from previous KES;
S_i ($u \le i < w$); $S_w = 0$

Initialization:
$\hat{\Delta}_{even}^{(u)}(x) = \hat{\Delta}_{even}^{(u)}(x) + S_u \Lambda_{even}^{(u)}(x)$
$\hat{\Theta}_{even}^{(u)}(x) = \hat{\Theta}_{even}^{(u)}(x) + S_u B_{even}^{(u)}(x)$

Iterations: for $r = u, u+2, \cdots, w-2$

1) $\Lambda_{even}^{(r+2)}(x) = \gamma^{(r)} \Lambda_{even}^{(r)}(x) + \hat{\Delta}_0^{(r)} x^2 B_{even}^{(r)}(x)$
2) $\Lambda_{odd}^{(r+2)}(x) = \gamma^{(r)} \Lambda_{odd}^{(r)}(x) + \hat{\Delta}_0^{(r)} x^2 B_{odd}^{(r)}(x)$
3) $\hat{\Delta}_{even}^{(r+2)}(x) = \gamma^{(r)} (\hat{\Delta}_{even}^{(r)}(x)/x^2 + S_{r+1} \Lambda_{odd}^{(r)}(x)/x + S_{r+2} \Lambda_{even}^{(r)}(x)) + \hat{\Delta}_0^{(r)} (\hat{\Theta}_{even}^{(r)}(x) + S_{r+1} x B_{odd}^{(r)}(x) + S_{r+2} x^2 B_{even}^{(r)}(x))$

if ($\hat{\Delta}_0^{(r)} \neq 0$ and $k^{(r)} \ge -1$)

4) $B_{even}^{(r+2)}(x) = \Lambda_{even}^{(r)}(x)$; $B_{odd}^{(r+2)}(x) = \Lambda_{odd}^{(r)}(x)$
5) $\hat{\Theta}_{even}^{(r+2)}(x) = \hat{\Delta}_{even}^{(r)}(x)/x^2 + S_{r+1} \Lambda_{odd}^{(r)}(x)/x + S_{r+2} \Lambda_{even}^{(r)}(x)$
6) $\gamma^{(r+2)} = \hat{\Delta}_0^{(r)}$; $k^{(r+2)} = -k^{(r)} - 2$

else

7) $B_{even}^{(r+2)}(x) = x^2 B_{even}^{(r)}(x)$; $B_{odd}^{(r+2)}(x) = x^2 B_{odd}^{(r)}(x)$
8) $\hat{\Theta}_{even}^{(r+2)}(x) = \hat{\Theta}_{even}^{(r)}(x) + S_{r+1} x B_{odd}^{(r)}(x) + S_{r+2} x^2 B_{even}^{(r)}(x)$
9) $\gamma^{(r+2)} = \gamma^{(r)}$; $k^{(r+2)} = k^{(r)} + 2$

where $c_i(x)$ is a sub-codeword, $\tilde{c}_l(x)$ is a nested codeword, α is a primitive element of $GF(2^q)$, and $\alpha^{il}(x)$ is the polynomial form of the standard basis representation of α^{il}.

GII-BCH decoding includes two stages. The first is the traditional BCH decoding that corrects $\le t_0$ errors in each received sub-word. $2t_0$ syndromes are computed from each received sub-word, $y(x)$, as $S_j = y(\alpha^{j+1})$ ($0 \le j < 2t_0$). Then a KES, such as the BM algorithm [10], uses the syndromes to iteratively compute the error locator polynomial $\Lambda(x)$. In iteration r, a discrepancy coefficient $\delta^{(r)} = \sum \Lambda_i^{(r)} S_{r-i}$ is calculated to update the polynomials. The riBM algorithm utilizes a discrepancy polynomial $\hat{\Delta}^{(r)}(x) = \Lambda(x) S(x)/x^r$, whose constant coefficient, $\hat{\Delta}_0^{(r)}$, equals $\delta^{(r)}$ [3]. $\hat{\Delta}(x)$ is initialized as $S(x)$ and updated in parallel with $\Lambda(x)$ so that the critical path is reduced to one multiplier and one adder.

The second-stage nested decoding can correct more errors. $2t$ higher-order syndromes are needed to correct t extra errors. Let the indices of the sub-words with extra errors be $i_0, \cdots, i_{b-1}$ ($b \le v$). Since the nested codewords are at least t_1-error-correcting, $2(t_1 - t_0)$ higher-order syndromes are computed as $\tilde{S}_j^{(l)} = \tilde{y}_l(\alpha^{j+1})$ ($0 \le l < b$, $2t_0 \le j < 2t_1$), where $\tilde{y}_l(x) = \sum_{i=0}^{m-1} \alpha^{il}(x) y_i(x)$. From (1), higher-order syndromes for the sub-words can be derived as

$$\left[S_j^{(i_0)}, S_j^{(i_1)}, \cdots, S_j^{(i_{b-1})}\right]^T = A^{-1} \left[\tilde{S}_j^{(0)}, \tilde{S}_j^{(1)}, \cdots, \tilde{S}_j^{(b-1)}\right]^T, \quad (2)$$

where $A_{u,w} = \alpha^{i_w u(j+1)}$ ($0 \le u, w < b$). Then the BM algorithm takes the $2(t_1 - t_0)$ higher-order syndromes to correct $\le t_1$ errors in each of the b sub-words. If there are $b' \le v-1$ sub-words with more errors, $2(t_2 - t_1)$ higher-order syndromes are derived in the next round and this process is repeated for up to v rounds.

The KES in the nested decoding cannot continue from the result of the sub-word decoding and use the riBM algorithm to shorten the critical path since the higher-order syndromes are not available in the beginning to initialize $\hat{\Delta}(x)$. In [4], $\hat{\Delta}(x)$ is re-initialized according to all the higher-order syndromes using expensive multiplier-adder trees. Algorithm 1 [7] incorporates two higher-order syndromes, S_{r+1} and S_{r+2}, to update $\hat{\Delta}(x)$ in iteration r so that the correct $\delta^{(r+2)}$ is always ready before iteration $r+2$ and the odd iterations are skipped for GII-BCH decoding. In this algorithm, 'even' and 'odd' denote the even and odd coefficients, respectively, of the polynomials. $B(x)$ and $\hat{\Theta}(x)$ are auxiliary polynomials assisting the updating of $\Lambda(x)$ and $\hat{\Delta}(x)$, respectively. A scaled nested KES (SNK) algorithm was also developed in [7] to reduce the area requirement. The critical paths of these algorithms have two multipliers due to the computations in Line 3. Although they are reduced to one multiplier by applying slow-down and re-timing [11], two sub-words are interleaved to increase the efficiency. The critical path is truly shortened to one multiplier in the scaled fast nested KES (SFNK) algorithm [9] by combining and pre-computing the scalars in Line 3, which introduce 5 clock cycles at initialization. Besides, both the SNK and SFNK designs require a number of multipliers for scalar pre-computation.

A major application of GII codes is SCMs and they require short codes with low redundancy, such as 2560 data bits protected by 10% redundancy. An example GII-BCH code for these code parameters is a [4,3] code over $GF(2^{10})$ with sub-codeword length $n = 704$ and $[t_0, t_1, t_2, t_3]$=[3,5,6,11]. For such short codes, the 5 clock cycles for scalar pre-computation [9] are more than the number of clock cycles for the nested KES iterations themselves, which is $t_i - t_{i-1}$ for nested decoding round i. Also when t_v is smaller, the scalar pre-computation in [7] and [9] may dominate the overall nested KES area.

III. Efficient KES for GII-BCH Nested Decoding

An alternative approach is proposed in this section to reduce the critical path of the nested KES to one multiplier. Instead of combining and pre-computing the scalars, the computations in Line 3 of Algorithm 1 are broken down and implemented in two clock cycles. Besides, reformulations are carried out to eliminate one clock cycle in the end. As a result, the proposed nested KES for decoding iteration i requires $2(t_i - t_{i-1})$ clock cycles with no extra latency for scalar pre-computation or initialization. For small $t_i - t_{i-1}$ as in short GII codes, the proposed design achieves significant latency reduction compared to previous approaches. Additionally, multipliers are shared among the computations and no complicated scalar pre-calculation is needed. Hence, the proposed design also requires much smaller area than prior architectures.

Each term in the parentheses in Line 3 of Algorithm 1 has up to one scalar, and the second scalar is outside of the parentheses. Denote the sums in the first and second parentheses by $\hat{\Delta}_{even}'^{(r)}(x)/x^2$ and $\hat{\Theta}_{even}'^{(r)}(x)$, respectively. To reduce the critical path to one multiplier, they are computed in the first clock cycle and then multiplied with $\gamma^{(r)}$ and $\hat{\Delta}_0^{(r)}$ in the second clock cycle. They are also used to update $\hat{\Theta}_{even}^{(r+2)}(x)$ as in Line 5 and 8 of Algorithm 1. The coefficients of the same degree in these polynomials can be calculated using four multipliers. In the second clock cycle, each coefficient

Algorithm 2: Nested GII-BCH KES w. Switched Updating

Input: $\Lambda_{even}^{(u)}(x)$, $\Lambda_{odd}^{(u)}(x)$, $B_{even}^{(u)}(x)$, $B_{odd}^{(u)}(x)$, $\bar{\Delta}_{even}^{(u)}(x)$, $\bar{\Theta}_{even}^{(u)}(x)$, $\gamma^{(u)}$, $k^{(u)}$ from previous KES;
syndromes S_i $(u{-}1{\le}i{\le}w{-}2)$
set $S_{u-1}{=}0$ for the first nested decoding round

Iterations: for $r{=}u, u{+}2, \cdots, w{-}2$

1) $\hat{\Delta}_{even}^{(r)}(x){=}\bar{\Delta}_{even}^{(r)}(x){+}S_{r-1}\Lambda_{odd}^{(r)}(x)/x{+}S_r\Lambda_{even}^{(r)}(x)$
2) $\hat{\Theta}_{even}^{(r)}(x){=}\bar{\Theta}_{even}^{(r)}(x){+}S_{r-1}B_{odd}^{(r)}(x)/x{+}S_rB_{even}^{(r)}(x)$

---

3) $\Lambda_{even}^{(r+2)}(x){=}\gamma^{(r)}\Lambda_{even}^{(r)}(x){+}\hat{\Delta}_0^{(r)}x^2B_{even}^{(r)}(x)$
4) $\Lambda_{odd}^{(r+2)}(x){=}\gamma^{(r)}\Lambda_{odd}^{(r)}(x){+}\hat{\Delta}_0^{(r)}x^2B_{odd}^{(r)}(x)$
5) $\bar{\Delta}_{even}^{(r+2)}(x){=}\gamma^{(r)}\hat{\Delta}_{even}^{(r)}(x)/x^2{+}\hat{\Delta}_0^{(r)}\hat{\Theta}_{even}^{(r)}(x)$
6) if ($\hat{\Delta}_0^{(r)}{\neq}0$ and $k^{(r)}{\geq}{-}1$)
7) $\quad B_{even}^{(r+2)}(x){=}\Lambda_{even}^{(r)}(x)$; $B_{odd}^{(r+2)}(x){=}\Lambda_{odd}^{(r)}(x)$
8) $\quad \bar{\Theta}_{even}^{(r+2)}(x){=}\hat{\Delta}_{even}^{(r)}(x)/x^2$
9) $\quad \gamma^{(r+2)}{=}\delta^{(r)}$; $k^{(r+2)}{=}{-}k^{(r)}{-}2$
10) else
11) $\quad B_{even}^{(r+2)}(x){=}x^2B_{even}^{(r)}(x)$; $B_{odd}^{(r+2)}(x){=}x^2B_{odd}^{(r)}(x)$
12) $\quad \bar{\Theta}_{even}^{(r+2)}(x){=}\hat{\Theta}_{even}^{(r)}(x)$
13) $\quad \gamma^{(r+2)}{=}\gamma^{(r)}$; $k^{(r+2)}{=}k^{(r)}{+}2$

in $\hat{\Delta}_{even}^{(r+2)}(x){=}\gamma^{(r)}\hat{\Delta}_{even}^{\prime(r)}(x)/x^2{+}\hat{\Delta}_0^{(r)}\hat{\Theta}_{even}^{\prime(r)}(x)$ is computed by using another two multipliers. Besides, the four multipliers for $\hat{\Delta}_{even}^{\prime(r)}(x)/x^2$ and $\hat{\Theta}_{even}^{\prime(r)}(x)$ calculation are shared to update $\Lambda^{(r+2)}(x)$ as in Line 1 and 2 and also reused for the $\hat{\Delta}_{even}^{(u)}(x)$ and $\hat{\Theta}_{even}^{(u)}(x)$ initialization. Each iteration has two clock cycles and the initialization needs one. Hence, $1{+}2(t_i{-}t_{i-1})$ clock cycles are required to finish the nested KES over one sub-word in decoding round i using this split polynomial updating. This latency is much shorter than the $5{+}(t_i{-}t_{i-1})$ clock cycles for the SFNK design when $(t_i{-}t_{i-1})$ is small, such as in the case of the short code with $[t_0, t_1, t_2, t_3]{=}[3, 5, 6, 11]$.

In the above split polynomial updating, $\hat{\Delta}_{even}^{\prime(r)}(x)/x^2$ and $\hat{\Theta}_{even}^{\prime(r)}(x)$ are first computed and then $\Lambda^{(r+2)}(x)$ is updated. However, only $\Lambda(x)$ is needed as the KES output. If the order of polynomial updating can be switched, then the $\hat{\Delta}_{even}^{(r)}(x)/x^2$ and $\hat{\Theta}_{even}^{(r)}(x)$ computation for the last iteration can be skipped. Accordingly, the nested KES latency can be reduced by one clock cycle, which is a significant reduction when $(t_i{-}t_{i-1})$ is small. It can be observed from Algorithm 1 that the $\Lambda_{even}^{(r+2)}(x)$ in Line 1 multiplied with S_{r+2} happens to be the sum of the third items in the two parentheses of Line 3. Besides, the $\Lambda_{odd}^{(r+2)}(x)$ in Line 2 multiplied by S_{r+1}/x is the sum of the second items in the parentheses of Line 3. Hence, $\Lambda_{even}^{(r+2)}(x)$ and $\Lambda_{odd}^{(r+2)}(x)$ can be reused to calculate $\hat{\Delta}_{even}^{(r+2)}(x)$ in the next clock cycle. Besides, the critical path can be kept as one multiplier since the rest of the two terms in the $\hat{\Delta}_{even}^{(r+2)}(x)$ formula also have one single scalar each.

Our proposed reformulated nested GII-BCH KES algorithm is listed in Algorithm 2. This algorithm also takes the output of the KES results from the sub-word decoding or the previous nested decoding round as the input. In the first nested decoding round, S_{u-1} is set to zero. Hence, the computations in Line 1 and 2 of Algorithm 2 are essentially the same as the polynomial initialization in Algorithm 1. Accordingly, the

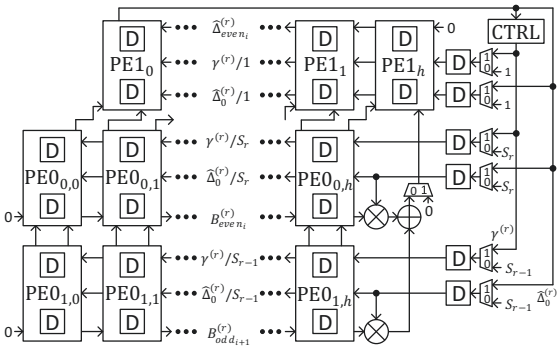

Fig. 1. Overall architecture for proposed nested GII-BCH KES.

constant coefficient of $\hat{\Delta}_{even}^{(r)}(x)$ is the discrepancy coefficient. In the next clock cycle, all the computations in Line 3 through 13 are carried out. The dashed line in Algorithm 2 is used to separate the computations executed in the even and odd clock cycles. The updating of $\Lambda_{even}^{(r+2)}(x)$ and $\Lambda_{odd}^{(r+2)}(x)$ are identical to those in Algorithm 1. At the same time, the first items in the two parentheses in Line 3 of Algorithm 1 are multiplied with the corresponding scalars as in Line 5 of Algorithm 2. The result is denoted by $\bar{\Delta}_{even}^{(r+2)}(x)$. Then in the first clock cycle of the next iteration, $\bar{\Delta}_{even}^{(r+2)}(x)$ is further added with another two items as in Line 1. As explained previously, the sum of these two terms equals the sum of the other four terms of the $\hat{\Delta}_{even}^{(r+2)}(x)$ formula in Algorithm 1. Therefore, the $\hat{\Delta}_{even}^{(r)}(x)$ computed in Line 1 of Algorithm 2 is the same as the $\hat{\Delta}_{even}^{(r)}(x)$ in Algorithm 1. Similar computations are carried out to derive $\hat{\Theta}_{even}^{(r)}(x)$. At the end of the second clock cycle of the last iteration, $\Lambda^{(w)}(x)$ is computed. Hence this process takes $2(t_i - t_{i-1})$ clock cycles for nested decoding round i. The $\bar{\Delta}_{even}^{(w)}(x)$ and $\bar{\Theta}_{even}^{(w)}(x)$ derived in the end are updated to $\hat{\Delta}_{even}^{(w)}(x)$ and $\hat{\Theta}_{even}^{(w)}(x)$, respectively, at the beginning of the KES of the next nested decoding round according to Line 1 and 2 of Algorithm 2.

IV. Hardware Architectures and Comparisons

An efficient architecture is developed in this section to implement the proposed low-latency nested KES algorithm for the decoding of short GII-BCH codes. Then the hardware complexity is analyzed and compared with prior designs using the example code over $GF(2^{10})$ with $[t_0, t_1, t_2, t_3]{=}[3, 5, 6, 11]$.

Fig. 1 shows the overall architecture of the nested KES for GII-BCH decoding. It consists of $\lceil(t_v{+}1)/2\rceil$ groups of processing elements (PEs) and a control unit. The PEs carry out the polynomial updating in Algorithm 2 in parallel. In the first clock cycle of each iteration, they implement Line 1 and 2 of Algorithm 2. The discrepancy coefficient, $\hat{\Delta}_0^{(r)}$, is calculated in the leftmost PE1 at the end of this clock cycle. Then in the second clock cycle of each iteration, $\hat{\Delta}_0^{(r)}$ is sent to each PE through the control logic to update the rest polynomials. Only the highest coefficient of $\hat{\Theta}_{even}^{(r)}(x)$ needs to be calculated by the two highest PE0s. Hence, they are simplified to two multipliers and one adder as in Fig. 1.

The details of one group of PEs are illustrated in Fig. 2(a). The critical path is denoted by the ticker wires. Applying re-

TABLE I
COMPLEXITIES OF NESTED GII-BCH KES ARCHITECTURES FOR EXAMPLE CODE OVER $GF(2^{10})$ WITH $[t_0, t_1, t_2, t_3] = [3, 5, 6, 11]$

	Mult.	Add	Reg.	Mux	Mux with const. input	Inv.	total # XORs	crit. path # gates	# clks/sub-word of KES in nested round 1, 2, 3	normalized ATP nested round 1, 2, 3
re-init. [4]	86	58	36	28	0	0	16904	10	3, 2, 6 (1+$(t_i - t_{i-1})$)	1.60, 2.13, 1.28
SNK [7]	46	42	98	18	39	0	11739	9	5, 3, 11 (1+2$(t_i - t_{i-1})$)*	1.67, 2.00, 1.47
SFNK [9]	78	57	85	37	40	2	18042	10	7, 6, 10 (5+$(t_i - t_{i-1})$)	3.98, 6.83, 2.28
proposed	38	43	48	28	9	0	8807	9	4, 2, 10 (2$(t_i - t_{i-1})$)	1.00, 1.00, 1.00

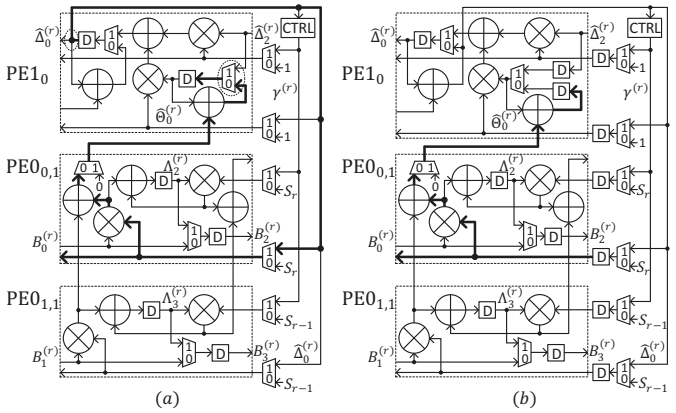

Fig. 2. PE architectures for proposed nested GII-BCH KES: (a) before re-timing; (b) after re-timing.

timing along the cutsets denoted by the dashed lines, the critical path is reduced to 1 multiplier and 3 adders/multiplexers as shown in the architecture of Fig. 2(b).

The complexity of the proposed design is listed in Table I. The control unit is simple and its area is negligible. Since t_v=11, h+1=$\lceil(11+1)/2\rceil$=6 groups of PEs are needed. Over $GF(2^{10})$, each multiplier using normal basis representation can be implemented by the area of around 174 XOR gates with 6 gates in the critical path. The areas of each adder, register, and multiplexer are about those of 10, 30, and 10 XOR gates, respectively. A multiplexer with a constant input has a half of the complexity of a general multiplexer. From these assumptions, the overall complexity of the proposed design for the example code can be estimated as listed in Table I.

For comparisons, the complexities of the nested KES architecture using the re-initialization scheme [4], SNK architecture [7], and SFNK architecture [9] are also included in Table I. In the design of [4] for the example code, 50 multipliers and 40 adders organized in tree structures are required to re-initialize $\hat{\Delta}_{even}(x)$ and $\hat{\Theta}_{even}(x)$, before a simplified version of the riBM architecture for BCH codes is used to carry out the KES process. As a result, this design requires around twice the area with longer critical path compared to the proposed design.

Even though the SNK and SFNK designs have the same number of PE groups as the proposed architecture and each group also has 6 multipliers, they require many more multipliers for scalar pre-computation. Besides, the application of the slow-down technique in the SNK architecture doubles the number of registers. Also an inverter over $GF(2^{10})$ takes the area of around 390 XOR gates to implement. Accordingly, it can be estimated that the proposed design requires 25% and 51% smaller area than the SNK and SFNK architectures, respectively. Additionally, the SFNK design has one more gate in the critical path.

The numbers of clock cycles needed for the KES over one sub-word in the i-th nested decoding round of different designs for the example GII-BCH code are listed in Table I. Note that the SNK design interleaves the decoding of two sub-words and hence its latency for two sub-words is the same as that for one sub-word. Despite that the proposed design requires two clock cycles for each iteration, it does not have the complicated scalar pre-computation as in the SFNK design, which takes 5 clock cycles regardless of $t_i - t_{i-1}$. As a result, the proposed design achieves lower latency for the short example code in the first two nested decoding rounds. Earlier nested decoding rounds are carried out with much higher probabilities than later rounds in GII decoding. Also, the proposed reformulation saves one clock cycle compared to the SNK architecture. Although the design from [4] has even lower latency in terms of the clock cycle number, its area requirement is almost twice. ATP is usually used to compare designs with different area and latency. It can be calculated that the proposed design achieves at least 1-(1/1.60+1/2.13+1/1.28)/3=37% lower ATP averaged over the three nested decoding rounds for the example code compared to prior efforts.

For GII-BCH codes with smaller t_v, the area saving of the proposed design would be even larger, because the scalar pre-computation takes a significant portion of the overall area in the SNK and SFNK architectures and its complexity does not change with t_v. For codes with larger v, which have better error-correcting performance under the same redundancy, $t_i - t_{i-1}$ are even smaller. In this case, the proposed design can achieve even more significant latency reduction.

V. CONCLUSIONS

This paper proposes an efficient design to reduce the area complexity and decoding latency of the nested KES for short GII-BCH codes. The proposed split polynomial updating reduces the critical path to one multiplier without pre-computing combined scalars. Besides, substantial area saving is achieved by sharing hardware units for polynomial updating. Additionally, one clock cycle is eliminated from the nested KES of each decoding round by reformulating the nested KES algorithm. Efficient hardware architectures are also developed and the critical path is further reduced by re-timing. Overall, the proposed nested KES design can achieve significant reductions in the area and latency for short GII-BCH codes compared to previous designs. Future work will study other decoder components for short GII-BCH codes.

REFERENCES

[1] X. Tang and R. Koetter, "A novel method for combining algebraic decoding and iterative processing," in *Proc. IEEE Int. Symp. Inf. Theory*, Seattle, WA, USA, Jul. 2006, pp. 474-478.

[2] Y. Wu, "Generalized integrated interleaved codes," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 1102-1119, Feb. 2017.

[3] D. V. Sarwate and N. R. Shanbhag, "High-speed architecture for Reed-Solomon decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 5, pp. 641-655, Oct. 2001.

[4] W. Li, J. Lin, and Z. Wang, "A 124-Gb/s decoder for generalized integrated interleaved codes," *IEEE Trans. Circuits and Syst. I: Reg. Papers*, vol. 66, no. 8, pp. 3174-3187, Aug. 2019.

[5] X. Zhang and Z. Xie, "Efficient architectures for generalized integrated interleaved decoder," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 66, no. 10, pp. 4018-4031, Oct. 2019.

[6] Z. Xie and X. Zhang, "Scaled nested key equation solver for generalized integrated interleaved decoder," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 67, no. 11, pp. 2457-2461, Nov. 2020.

[7] Z. Xie and X. Zhang, "Reduced-complexity key equation solvers for generalized integrated interleaved BCH decoders," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 67, no. 12, pp. 5520-5529, Dec. 2020.

[8] Z. Xie and X. Zhang, "Fast nested key equation solvers for generalized integrated interleaved decoder," *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 68, no. 1, pp. 483-495, Jan. 2021.

[9] Z. Xie and X. Zhang, "Scaled fast nested key equation solver for generalized integrated interleaved BCH decoders," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Toronto, Canada, Jun. 2021, pp. 7883-7887.

[10] E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, 1968.

[11] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, John Wiley & Sons, 1999.