SCALED FAST NESTED KEY EQUATION SOLVER FOR GENERALIZED INTEGRATED INTERLEAVED BCH DECODERS

Zhenshan Xie and Xinmiao Zhang

The Ohio State University

ABSTRACT

The generalized integrated interleaved BCH (GII-BCH) codes are among the best error-correcting codes for next-generation terabit/s memories. The key equation solver (KES) in the nested decoding of GII codes limits the achievable clock frequency. Recently, by polynomial scalar pre-computation, the critical path of the nested KES for Reed-Solomon (RS)-based GII codes has been reduced to one multiplier. However, for GII-BCH codes, the nested KES has more complicated formulas in order to skip the odd iterations and hence prior techniques do not directly extend. This paper proposes novel reformulations of the nested BCH KES to enable scalar pre-computation. Additionally, polynomial scaling is incorporated to enable complexity reduction. As a result, the critical path of the nested BCH KES with odd iterations skipped is reduced to one multiplier. For an example GII-BCH code over $GF(2^{12})$, the proposed design reduces the average nested BCH KES latency to around a half with similar silicon area compared to the best prior design.

***Index Terms*—** BCH codes, Error-correcting codes, Generalized integrated interleaved codes, Key equation solver.

1. INTRODUCTION

Hyper-speed decoding and excellent random error-correction capability are essential to next-generation memories. These goals are reached by the generalized integrated interleaved (GII) codes [1, 2] that nest short BCH sub-codewords to generate codewords of stronger BCH codes. They achieve hundreds of megabit/s throughput with orders of magnitude better correction capability compared to traditional BCH codes [3].

The GII decoding has two stages. Conventional BCH decoding is first carried out on individual sub-codewords. The key equation solver (KES) step, such as the Berlekamp-Massey (BM) algorithm [4], takes syndromes to compute the error locator polynomial. It limits the achievable clock frequency. By using a discrepancy polynomial, the critical path of the KES is reduced to one multiplier in the reformulated inversionless (ri-)BM algorithm [5]. When there are extra errors, the second-stage nested decoding is activated and higher-order syndromes are acquired utilizing the nested codewords. The KES of the nested decoding can continue from the KES results of sub-codeword decoding [2]. However, the riBM algorithm cannot be applied to reduce the critical path, since the higher-order syndromes are not available in the first stage to initialize the discrepancy polynomial.

Re-initializing the discrepancy polynomial before the nested KES [6] requires many large multiplier-adder trees. Instead, the higher-order syndromes are incorporated into the nested KES iterations one by one in [7] and the area is reduced by scaling polynomials to enable product term sharing [3, 8]. However, the designs in [3, 7, 8] actually have two multipliers in the critical path. They rely on the slow-down and re-timing techniques [9] to reduce the critical path to one multiplier and waste half of the clock cycles when only one sub-codeword has extra errors, which happens with very high probability. By scalar pre-computation, the critical path is truly reduced to one multiplier in the fast nested KES algorithms for GII codes based on Reed-Solomon (RS) codes [10]. However, to skip the odd iterations, the nested KES for GII-BCH codes has more complicated formulas. As a result, it is much more difficult to pre-compute the scalars and the techniques in [10] for GII-RS codes do not directly apply.

This paper proposes a scaled fast nested KES algorithm for GII-BCH codes with the odd iterations skipped. Novel algorithmic reformulations are developed to pre-compute the polynomial scalars in parallel with the polynomial updating so that the critical path is truly reduced to one multiplier without applying the slow-down technique. To reduce the complexity, the polynomials are also scaled to enable product term sharing while keeping one multiplier in the critical path. As a result, the average nested BCH KES latency is reduced to almost a half in terms of the number of clock cycles. For an example GII-BCH code over $GF(2^{12})$, the proposed design has similar area and only one more gate in the critical path compared to the best prior work [3].

2. GII-BCH CODES AND NESTED KES

A GII codeword is divided into m sub-codewords $c_0, c_1, \cdots, c_{m-1} \in \mathcal{C}_0$ over $GF(2^q)$. Besides, the nesting of the sub-codewords as defined in (1) using primitive element $\alpha \in GF(2^q)$ generates $\tilde{c}_l \in \mathcal{C}_{v-l}$. $\mathcal{C}_v \subseteq \mathcal{C}_{v-1} \subseteq \cdots \subseteq \mathcal{C}_1 \subset \mathcal{C}_0$

This material is based upon work supported by Kioxia Corporation and the National Science Foundation under Award No. 2011785.

are RS or BCH codes over $GF(2^q)$. For GII-BCH codes, $\alpha^{il}=\alpha^{il}(x)$ as defined in [2]. Alternative nestings for improving the decoding locality are available in [11–13].

$$\mathcal{C}\triangleq\left\{c=[c_0,\cdots,c_{m-1}]: c_i\in C_0, \tilde{c}_l=\sum_{i=0}^{m-1}\alpha^{il}c_i\in\mathcal{C}_{v-l}, 0\le l<v\right\} \quad (1)$$

Denote the error-correction capability of $\mathcal{C}_i$ by t_i ($0\le i\le v$). In the sub-codeword decoding, $2t_0$ syndromes, $S_j=y(\alpha^{j+1})$ ($0\le j<2t_0$), are computed for each received sub-codeword $y(x)$. If any syndrome is nonzero, the KES is carried out. The r-th iteration of the BM algorithm computes a discrepancy coefficient $\delta^{(r)}=\sum_{i=0}^{L_\Lambda^{(r)}}\Lambda_i^{(r)}S_{r-i}$, where $L_\Lambda^{(r)}$ is the length of $\Lambda^{(r)}(x)$, and then uses $\delta^{(r)}$ to update the error locator polynomial $\Lambda^{(r)}(x)$. For BCH decoding, the odd iterations can be skipped since $S_{2i+1}=S_i^2$. In the end, root search is done for $\Lambda^{(2t_0)}(x)$ to find the error locations. Computing $\delta^{(r)}$ using a large multiplier-adder tree followed by $\Lambda^{(r)}(x)$ updating leads to long critical path. The critical path is reduced to one multiplier and one adder in the riBM algorithm [5] by using discrepancy polynomial $\hat{\Delta}^{(r)}(x)=\Lambda^{(r)}(x)S(x)/x^r$, whose $\hat{\Delta}_0^{(r)}=\delta^{(r)}$. It is initialized as $\sum_{j=0}^{2t_0-1}S_jx^j$ and updated simultaneously with $\Lambda^{(r)}(x)$.

In the second-stage nested decoding, higher-order syndromes are computed from (corrupted) nested codewords $\tilde{c}_l$. These syndromes are converted to higher-order syndromes of the sub-codewords by reversing the linear combination in (1). Then more errors in the sub-codewords can be corrected. The nested decoding is repeated for up to v rounds to correct the sub-codewords with up to $t_1, t_2, \cdots, t_v$ errors. More details about the decoding process are available in [2, 7].

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:
$\hat{\Delta}_{even}^{(u)}(x)=\hat{\Delta}_{even}^{(u)}(x)+S_u\Lambda_{even}^{(u)}(x)$
$\hat{\Theta}_{even}^{(u)}(x)=\hat{\Theta}_{even}^{(u)}(x)+S_uB_{even}^{(u)}(x)$
for $r=u,u+2,\cdots,w-2$
1) $\Lambda_{even}^{(r+2)}(x)=\gamma^{(r)}\Lambda_{even}^{(r)}(x)+\hat{\Delta}_0^{(r)}x^2B_{even}^{(r)}(x)$
2) $\Lambda_{odd}^{(r+2)}(x)=\gamma^{(r)}\Lambda_{odd}^{(r)}(x)+\hat{\Delta}_0^{(r)}x^2B_{odd}^{(r)}(x)$
3) $\hat{\Delta}_{even}^{(r+2)}(x)=\gamma^{(r)}(\hat{\Delta}_{even}^{(r)}(x)/x^2+S_{r+1}\Lambda_{odd}^{(r)}(x)/x+S_{r+2}\Lambda_{even}^{(r)}(x))$
$\quad+\hat{\Delta}_0^{(r)}(\hat{\Theta}_{even}^{(r)}(x)+S_{r+1}xB_{odd}^{(r)}(x)+S_{r+2}x^2B_{even}^{(r)}(x))$
if ($\hat{\Delta}_0^{(r)}\neq0$ 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) $B_{even}^{(r+2)}(x)=x^2B_{even}^{(r)}(x)$; $B_{odd}^{(r+2)}(x)=x^2B_{odd}^{(r)}(x)$
8) $\hat{\Theta}_{even}^{(r+2)}(x)=\hat{\Theta}_{even}^{(r)}(x)+S_{r+1}xB_{odd}^{(r)}(x)+S_{r+2}x^2B_{even}^{(r)}(x)$
9) $\gamma^{(r+2)}=\gamma^{(r)}$; $k^{(r+2)}=k^{(r)}+2$

The riBM algorithm cannot be used to continue the KES in the nested decoding since the higher-order syndromes are not available in the very beginning to initialize $\hat{\Delta}(x)$. To avoid using multiplier-adder trees to compute $\delta^{(r)}$, Algorithm 1 [3] has been developed. It can continue from the riBM results of the sub-codeword decoding or previous nested decoding to incorporate higher-order syndromes S_u through S_{w-1}. In this algorithm, $B(x)$ and $\hat{\Theta}(x)$ are auxiliary polynomials and odd iterations are skipped. As can be observed from Line 3, this algorithm has two multipliers in the critical path. To make the critical path one multiplier, the solution in [3] is to apply slow-down and re-timing [9], which require two sub-codewords to be interleaved in the nested decoding.

3. SCALED FAST NESTED BCH KES ALGORITHM AND ARCHITECTURE

With very high probability, only one sub-codeword has extra errors and needs nested decoding [10]. In this case, dummy zeros are inserted if the slow-down technique is applied as in [3] and half of the clock cycles are wasted. For GII-RS codes, by pre-computing the polynomial scalars, fast nested KES algorithms were developed [10] to truly reduce the critical path to one multiplier without applying slow-down. However, the nested BCH KES algorithm skipping every odd iteration (Algorithm 1) has much more complicated formulas than the nested RS KES algorithm [7]. Hence, the techniques in [10] do not directly extend to GII-BCH codes.

In this section, by reformulating Algorithm 1, a scaled fast nested BCH KES algorithm with one multiplier in the critical path is developed. The approach for critical path reduction in our design is to pre-compute the combined scalars needed for iteration r before iteration r starts with one multiplier in the data path, although it may take multiple clock cycles. Multiplying the scalars to the polynomials in iteration r also has one multiplier in the data path. Accordingly, the critical path is reduced to one multiplier. Besides, scaled versions of the polynomials in Algorithm 1 are used to enable product term sharing and hence reduce the number of multiplications.

First, let us scale the polynomials to reduce the total number of multiplications. In Algorithm 1, the sum of $\gamma^{(r)}S_{r+2}\Lambda_{even}^{(r)}(x)$ and $\hat{\Delta}_0^{(r)}S_{r+2}x^2B_{even}^{(r)}(x)$ from Line 3 equals $\Lambda_{even}'^{(r+2)}(x)=S_{r+2}\Lambda_{even}^{(r+2)}(x)$. If $\Lambda_{even}'^{(r+2)}(x)$ is used instead, no separate multiplication is needed and two multiplications are saved. Every polynomial should be scaled by the same factor in order to keep the decoding results the same. Different from the scaling scheme in [3], S_{r+2}^{-1} is multiplied back to $\Lambda_{even}'^{(r+2)}(x)$ in the next iteration to simplify the combined scalar and facilitate the scalar pre-computation. Although finite field inverters have longer data path, the computation of S_{r+2}^{-1} can start earlier and gets pipelined since the syndromes are available before the nested KES. Similarly, the first and second rows of Line 3 of Algorithm 1 are scaled versions of $\hat{\Theta}_{even}^{(r+2)}(x)$ in Lines 5 and 8, respectively. If $\hat{\Theta}_{even}'^{(r+2)}(x)=\gamma^{(r)}\hat{\Theta}_{even}^{(r+2)}(x)$ or $\hat{\Delta}_0^{(r)}\hat{\Theta}_{even}^{(r+2)}(x)$ is utilized,

additional multiplications are saved. However, unlike the scaling by syndromes, $\gamma^{(r)}$ or $\hat{\Delta}_0^{(r)}$ is not available in advance and their inversion increases the critical path. Instead, these additional scalars are multiplied to the other polynomials to make the overall scalar for every polynomial the same. Add '$'$' to the polynomials and coefficients to differentiate the scaled algorithm. Overall, the combined scalar for $\Lambda_{even}^{\prime(r)}(x)$ is $\gamma^{\prime(r-2)}\gamma^{\prime(r)}(S_{r+2}S_r^{-1})$ or $\hat{\Delta}_0^{\prime(r-2)}\gamma^{\prime(r)}(S_{r+2}S_r^{-1})$. Following this analysis and Algorithm 1, it is derived that the scalar of $\Lambda_{even}^{\prime(r)}(x)$ accumulated over the iterations is $\hat{\Delta}_0^{\prime(r-2)}\gamma^{\prime(r-2)}\sigma^{(r-4)}\cdots\sigma^{(u)}(S_{r+2}S_r^{-1})$, where $\sigma^{(i)}$=$\gamma^{\prime(i)}$ or $\hat{\Delta}_0^{\prime(i)}$ if Lines 4-6 or 7-9, respectively, of Algorithm 1 were executed in iteration i.

Algorithm 2 Scaled Fast Nested GII-BCH KES Algorithm

input: $\Lambda_{even}^{\prime(u)}(x)$,$\Lambda_{odd}^{\prime(u)}(x)$,$B_{even}^{\prime(u)}(x)$,$B_{odd}^{\prime(u)}(x)$,$\hat{\Delta}_{even}^{\prime(u)}(x)$,$\hat{\Theta}_{even}^{\prime(u)}(x)$ $\gamma^{\prime(u)}$,$k^{(u)}$,$\rho^{(u)}$,$\xi^{(u)}$,$\lambda^{(u)}$,$\phi^{(u)}$,$\beta^{(u)}$,$\zeta^{(u)}$ from previous KES; S_i ($u\le i<w$); S_{w+j}=0 ($0\le j\le 8$); set S_{u-1}=1 if first round; set S_i'=S_i if $S_i\neq 0$, set S_i'=1 otherwise ($u-1\le i\le w+8$)

initialization:

$\hat{\Delta}_{even}^{\prime(u)}(x)$=$\hat{\Delta}_{even}^{\prime(u)}(x)$+$S_u\Lambda_{even}^{\prime(u)}(x)$

$\hat{\Theta}_{even}^{\prime(u)}(x)$=$\hat{\Theta}_{even}^{\prime(u)}(x)$+$S_uB_{even}^{\prime(u)}(x)$

$\Lambda_{even}^{\prime(u)}(x)$=$S_u'\Lambda_{even}^{\prime(u)}(x)$; $B_{even}^{\prime(u)}(x)$=$S_u'B_{even}^{\prime(u)}(x)$

execute Subroutine 1

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

1) if ($\hat{\Delta}_0^{\prime(r)}$=0) set $\rho^{(r)}$=1, $\rho_{S_i}^{(r)}$=$g_i^{(r)}$, $\xi^{(r)}$=$\xi_{S_i}^{(r)}$ = 0 (i=1,2)

2) $\Lambda_{even}^{\prime(r+2)}(x)$=$\rho_{S_2}^{(r)}\Lambda_{even}^{\prime(r)}(x)$+$\xi_{S_2}^{(r)}x^2B_{even}^{\prime(r)}(x)$

3) $\Lambda_{odd}^{\prime(r+2)}(x)$=$\rho_{S_1}^{(r)}\Lambda_{odd}^{\prime(r)}(x)$+$\xi_{S_1}^{(r)}x^2B_{odd}^{\prime(r)}(x)$

4) $\hat{\Delta}_{even}^{\prime(r+2)}(x)$=$\rho^{(r)}\hat{\Delta}_{even}^{\prime(r)}(x)/x^2$+$\rho_{S_1}^{(r)}\Lambda_{odd}^{\prime(r)}(x)/x$+$\rho_{S_2}^{(r)}\Lambda_{even}^{\prime(r)}(x)$ +$\xi^{(r)}\hat{\Theta}_{even}^{\prime(r)}(x)$+$\xi_{S_1}^{(r)}xB_{odd}^{\prime(r)}(x)$+$\xi_{S_2}^{(r)}x^2B_{even}^{\prime(r)}(x)$ (set $\rho_{S_i}^{(r)}$=$\xi_{S_i}^{(r)}$=0 if S_{r+i}=0 (i=1,2))

5) execute Subroutine 2

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

6) $B_{even}^{\prime(r+2)}(x)$=$\rho_{S_2}^{(r)}\Lambda_{even}^{\prime(r)}(x)$

7) $B_{odd}^{\prime(r+2)}(x)$=$\rho_{S_1}^{(r)}\Lambda_{odd}^{\prime(r)}(x)$

8) $\hat{\Theta}_{even}^{\prime(r+2)}(x)$=$\rho^{(r)}\hat{\Delta}_{even}^{\prime(r)}(x)/x^2$+$\rho_{S_1}^{(r)}\Lambda_{odd}^{\prime(r)}(x)/x$+$\rho_{S_2}^{(r)}\Lambda_{even}^{\prime(r)}(x)$ (set $\rho_{S_i}^{(r)}$=0 if S_{r+i}=0 (i=1,2))

9) execute Subroutine 3

10) $k^{(r+2)}$=$-k^{(r)}-2$

else

11) if ($\hat{\Delta}_0^{\prime(r)}$=0) set $\xi^{(r)}$=1, $\xi_{S_i}^{(r)}$=$g_i^{(r)}$ (i=1,2)

12) $B_{even}^{\prime(r+2)}(x)$=$\xi_{S_2}^{(r)}x^2B_{even}^{\prime(r)}(x)$

13) $B_{odd}^{\prime(r+2)}(x)$=$\xi_{S_1}^{(r)}x^2B_{odd}^{\prime(r)}(x)$

14) $\hat{\Theta}_{even}^{\prime(r+2)}(x)$=$\xi^{(r)}\hat{\Theta}_{even}^{\prime(r)}(x)$+$\xi_{S_1}^{(r)}xB_{odd}^{\prime(r)}(x)$+$\xi_{S_2}^{(r)}x^2B_{even}^{\prime(r)}(x)$ (set $\xi_{S_i}^{(r)}$=0 if S_{r+i}=0 (i=1,2))

15) execute Subroutine 4

16) $k^{(r+2)}$=$k^{(r)}$+2

The accumulated scalar has $(r-u)/2+2$ terms to multiply even if $(S_{r+2}S_r^{-1})$ is pre-computed. To keep one multiplier in the data path, at most $i+1$ terms can be iteratively multiplied up in i iterations. Hence, the combined scalar cannot be computed by one multiplier before iteration r starts. To address this issue, $c_u^{(r-4)}$=$\sigma^{(r-4)}\cdots\sigma^{(u)}$ is dropped from all scalars. The remaining part of the $\Lambda_{even}^{\prime(r)}(x)$ scalar is

$$\rho_{S_2}^{(r)} = \hat{\Delta}_0^{\prime(r-2)}\gamma^{\prime(r-2)}S_{r+2}S_r^{-1}. \tag{2}$$

All computations on the syndromes can be finished in advance, and $\gamma^{\prime(r-2)}$ equals either $\gamma^{\prime(r-4)}$ or $\hat{\Delta}_0^{\prime(r-4)}$. Hence, $\gamma^{\prime(r-2)}S_{r+2}S_r^{-1}$ can be pre-computed in iteration $r-4$ as either $\gamma^{\prime(r-4)}(S_{r+2}S_r^{-1})$ or $\hat{\Delta}_0^{\prime(r-4)}(S_{r+2}S_r^{-1})$. Then, it is multiplied with $\hat{\Delta}_0^{\prime(r-2)}$ in iteration $r-2$ to derive $\rho_{S_2}^{(r)}$. The combined scalars for $\hat{\Delta}_{even}^{\prime(r)}(x)$ and $\Lambda_{odd}^{\prime(r)}(x)$ are $\rho^{(r)}$=$\hat{\Delta}_0^{\prime(r-2)}\gamma^{\prime(r-2)}$ and $\rho_{S_1}^{(r)}$=$\hat{\Delta}_0^{\prime(r-2)}\gamma^{\prime(r-2)}S_{r+1}S_{r-1}^{-1}$, respectively. Similarly, they are pre-computed with one multiplier in the data path.

The scalar of $\hat{\Theta}_{even}^{\prime(r)}(x)$ is still $\hat{\Delta}_0^{\prime(r)}$ after the additional scaling for product term sharing. It can be derived that

$$\begin{aligned}\xi^{(r)} = \hat{\Delta}_0^{\prime(r)}/c_u^{(r-4)} &= \gamma^{\prime(r-2)}\hat{\Delta}_2^{\prime(r-2)} + \hat{\Delta}_0^{\prime(r-2)}\phi^{(r-2)} \\ &+ \gamma^{\prime(r-2)}S_{r-1}S_{r-3}^{-1}\Lambda_1^{\prime(r-2)} + \gamma^{\prime(r-2)}S_rS_{r-2}^{-1}\Lambda_0^{\prime(r-2)},\end{aligned} \tag{3}$$

where $\phi^{(r-2)}$=$\hat{\Theta}_0^{\prime(r-2)}/c_u^{(r-4)}$. Similarly, each term in (3) can be also computed with one multiplier in data path before iteration r. The combined scalars for $B_{odd}^{\prime(r)}(x)$ and $B_{even}^{\prime(r)}(x)$ are $\xi_{S_1}^{(r)}$=$\xi^{(r)}S_{r+1}S_{r-1}^{-1}$ and $\xi_{S_2}^{(r)}$=$\xi^{(r)}S_{r+2}S_r^{-1}$, respectively. They are also pre-computed with one multiplier in data path. Overall, the proposed scaled fast nested BCH KES algorithm is summarized in Algorithm 2 and the four subroutines.

Subroutine 1 Scalar Initialization

1) $g_i^{(u)}$=$S_{u+i}'S_{u+i-2}^{\prime-1}$ ($1\le i\le 8$); $g_{j,k}^{(u)}$=$g_j^{(u)}g_k^{(u)}$ (j=5,6; k=1,2,3,4)

2) $h_i^{(u)}$=$\gamma^{\prime(u)}g_i^{(u)}$ ($1\le i\le 4$); $h_{j,k}^{(u)}$=$\gamma^{\prime(u)}g_{j,k}^{(u)}$ (j=3,4; k=1,2)

3) $\rho_{S_1}^{(u)}$=$g_1^{(u)}\rho^{(u)}$; $\rho_{S_2}^{(u)}$=$g_2^{(u)}\rho^{(u)}$

4) $\xi_{S_1}^{(u)}$=$g_1^{(u)}\xi^{(u)}$+$g_1^{(u)}S_u\lambda^{(u)}$; $\xi_{S_2}^{(u)}$=$g_2^{(u)}\xi^{(u)}$+$g_2^{(u)}S_u\lambda^{(u)}$

5) $\xi^{(u)}$=$\xi^{(u)}$+$S_u\lambda^{(u)}$

6) $\phi_{S_1}^{(u)}$=$g_3^{(u)}\phi^{(u)}$+$g_3^{(u)}S_u\beta^{(u)}$; $\phi_{S_2}^{(u)}$=$g_4^{(u)}\phi^{(u)}$+$g_4^{(u)}S_u\beta^{(u)}$

7) $\phi^{(u)}$=$\phi^{(u)}$+$S_u\beta^{(u)}$

Subroutine 2 Scalar Updating

1) if ($\hat{\Delta}_0^{\prime(r)}\neq 0$) set $\tau^{(r)}$=$\hat{\Delta}_0^{\prime(r)}$, otherwise set $\tau^{(r)}$=$\zeta^{(r)}$

2) if ($\hat{\Delta}_0^{\prime(r)}$=0) set $\rho^{(r)}$=1

3) $\rho^{(r+2)}$=$\gamma^{\prime(r)}\tau^{(r)}$; $\rho_{S_1}^{(r+2)}$=$h_3^{(r)}\tau^{(r)}$; $\rho_{S_2}^{(r+2)}$=$h_4^{(r)}\tau^{(r)}$

4) $\xi^{(r+2)}$=$\gamma^{\prime(r)}\hat{\Delta}_2^{\prime(r)}$+$h_1^{(r)}\Lambda_1^{\prime(r)}$+$h_2^{(r)}\Lambda_0^{\prime(r)}$+$\hat{\Delta}_0^{\prime(r)}\phi^{(r)}$

$\xi_{S_1}^{(r+2)}$=$h_3^{(r)}\hat{\Delta}_2^{\prime(r)}$+$h_{3,1}^{(r)}\Lambda_1^{\prime(r)}$+$h_{3,2}^{(r)}\Lambda_0^{\prime(r)}$+$\hat{\Delta}_0^{\prime(r)}\phi_{S_1}^{(r)}$

$\xi_{S_2}^{(r+2)}$=$h_4^{(r)}\hat{\Delta}_2^{\prime(r)}$+$h_{4,1}^{(r)}\Lambda_1^{\prime(r)}$+$h_{4,2}^{(r)}\Lambda_0^{\prime(r)}$+$\hat{\Delta}_0^{\prime(r)}\phi_{S_2}^{(r)}$

(set $h_i^{(r)}$=$h_{3,i}^{(r)}$=$h_{4,i}^{(r)}$=0 if S_{r+i}=0 (i=1,2))

5) $\zeta^{(r+2)}$=$\rho^{(r)}\tau^{(r)}$; $\lambda^{(r+2)}$=$h_2^{(r)}\Lambda_0^{\prime(r)}$

6) $g_i^{(r+2)}$=$g_{i+2}^{(r)}$ ($1\le i\le 6$); $g_j^{(r+2)}$=$S_{r+j+2}'S_{r+j}^{\prime-1}$ (j=7,8)

$g_{k,l}^{(r+2)}$=$g_{k+2}^{(r)}g_{l+2}^{(r)}$ (k=5,6; l=1,2,3,4)

Algorithm 2 can start from the KES results from subcodeword decoding or previous nested decoding round. In

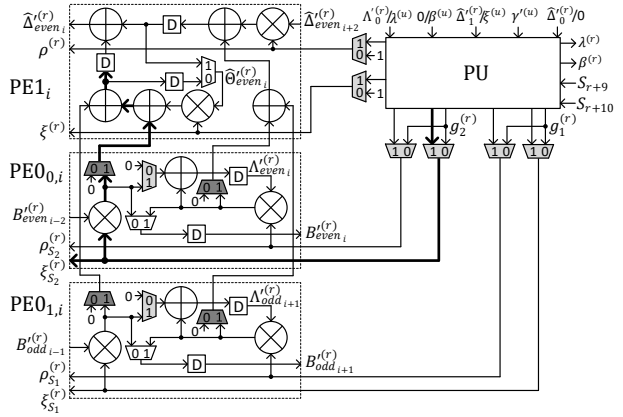

Fig. 1. PE architectures of scaled nested GII-BCH KES

the former case, $\gamma'^{(u)}$, $\rho^{(u)}$, $\xi^{(u)}$, $\lambda^{(u)}$, $\phi^{(u)}$, $\beta^{(u)}$, and $\zeta^{(u)}$ inputs are set to $\gamma^{(u)}$, $\gamma^{(u)}$, $\hat{\Delta}_0^{(u)}$, $\Lambda_0^{(u)}$, $\hat{\Theta}_0^{(u)}$, $B_0^{(u)}$, and $\gamma^{(u)}$, respectively. When an additional scalar introduced for product sharing is zero, the combined scalars are adjusted to eliminate its effect, such as in Lines 4, 8, and 14 of Algorithm 2. As it can be observed from Algorithm 2 and the four subroutines, none of the computations requires more than one multiplier in the data path. As a result, the overall critical path only consists of one multiplier.

Subroutine 3 Scalar Updating in 'if'

1) $\gamma'^{(r+2)}=\hat{\Delta}_0'^{(r)}$
2) $h_i^{(r+2)}=\hat{\Delta}_0'^{(r)} g_{i+2}^{(r)}$ ($1\le i\le 4$); $h_{j,k}^{(r+2)}=\hat{\Delta}_0'^{(r)} g_{j+2,k+2}^{(r)}$ (j=3,4;k=1,2)
3) $\phi^{(r+2)}=\hat{\Delta}_2'^{(r)}+g_1^{(r)}\Lambda_1'^{(r)}+g_2^{(r)}\Lambda_0'^{(r)}$
$\phi_{S_1}^{(r+2)}=g_5^{(r)}\hat{\Delta}_2'^{(r)}+g_{5,1}^{(r)}\Lambda_1'^{(r)}+g_{5,2}^{(r)}\Lambda_0'^{(r)}$
$\phi_{S_2}^{(r+2)}=g_6^{(r)}\hat{\Delta}_2'^{(r)}+g_{6,1}^{(r)}\Lambda_1'^{(r)}+g_{6,2}^{(r)}\Lambda_0'^{(r)}$
(set $g_i^{(r)}=g_{5,i}^{(r)}=g_{6,i}^{(r)}=0$ if $S_{r+i}=0$ (i=1,2))
4) $\beta^{(r+2)}=g_2^{(r)}\Lambda_0'^{(r)}$

Subroutine 4 Scalar Updating in 'else'

1) $\gamma'^{(r+2)}=\gamma'^{(r)}$
2) $h_i^{(r+2)}=\gamma'^{(r)} g_{i+2}^{(r)}$($1\le i\le 4$); $h_{j,k}^{(r+2)}=\gamma'^{(r)} g_{j+2,k+2}^{(r)}$($j$=3,4;$k$=1,2)
3) $\phi^{(r+2)}=\phi^{(r)}$; $\phi_{S_1}^{(r+2)}=g_5^{(r)}\phi^{(r)}$; $\phi_{S_2}^{(r+2)}=g_6^{(r)}\phi^{(r)}$
4) $\beta^{(r+2)}=0$

Algorithm 2 can be implemented by the architecture in Fig. 1. The processing elements (PEs) implement the polynomial updating and the four subroutines are implemented in the pre-processing unit (PU). There are $\lceil (t_v+1)/2 \rceil$ groups of PEs but only one single PU. Fig. 1 only shows the group of PEs processing the i-th coefficients of the polynomials. The PU architecture can be easily derived from the four subroutines. For conciseness, its details are not shown. The critical path is highlighted by the thicker wires in Fig. 1. It only consists of one multiplier and four adders/multiplexers. Adopting the scaling for product term sharing, each PE group only requires 6 instead of 14 multipliers.

Table 1. Complexities of nested GII-BCH KES architectures

	Add.	Mult.	Reg.	Mux.	Total # XORs	Crit. path # gates	Average # clks
Nested KES (Alg.1) [3]	210	300	367	90	77112	11	10.06
Scaled nested KES [3]	210	190	410	273	57648	12	10.06
proposed	225	222	253	293	59308	13	5.03

4. COMPLEXITY ANALYSES AND COMPARISONS

This section analyzes the complexity of the proposed scaled fast nested BCH KES architecture using a 4kB GII-BCH code over $GF(2^{12})$ with v=3 and error-correction capability [t_0, t_1, t_2, t_3]=[28, 32, 39, 58] as an example. This code is a good candidate for Flash memory applications.

The nested KES architecture has $\lceil (t_v+1)/2 \rceil$=30 groups of PEs. The complexity of the PEs can be counted from Fig. 1 and the PU is implemented by 15 adders, 42 multipliers, 41 registers, 23 multiplexers, and 2 inverters. Each adder, multiplier, register, multiplexer, and inverter over $GF(2^{12})$ can be implemented by the area of 12, 201, 36, 12, and 233 XORs, respectively [3]. Overall, the complexity is estimated as in Table 1.

As shown in Table 1, the proposed design has similar area as the scaled nested KES design in [3]. The area overhead would be even smaller for codes with larger t_v. In the case that the i-th nested decoding round is carried out for only one sub-codeword, the proposed design needs $t_i-t_{i-1}+1$ instead of $2(t_i-t_{i-1}+1)$ clock cycles, since it does not interleave two sub-codewords as a result of applying the slow-down technique and hence insert dummy zeros. For practical settings, with more than 99% probability, there is only one sub-codeword with extra errors and the number of errors does not exceed t_1. As a result, with only one more gate in the critical path, the proposed design reduces the average number of clock cycles for the nested KES to almost a half. In terms of throughput over area ratio, the proposed design achieves $(2\times 12/13)/(59308/57648)$=180% higher efficiency. The achievable latency reduction remains about the same for different GII-BCH codes. The designs in [7, 8] are not compared because they are for the nested KES of GII-RS codes and require twice the iterations.

5. CONCLUSION

This paper proposes a scaled fast nested KES algorithm and architecture for GII-BCH codes. Novel algorithmic reformulations are developed to enable the pre-computation of combined scalars and polynomial updating in parallel with only one multiplier in the critical path without applying the slow-down technique. Besides, polynomial scaling is incorporated to substantially reduce the area. As a result, the average latency of the nested KES for GII-BCH codes is reduced to about a half with similar area requirement compared to the best prior design. Future work will simplify the other components of GII-BCH decoders.

6. 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] 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.

[4] E. R. Berlekamp, *Algebraic Coding Theory*, rev. ed., Laguna Hills, CA: Aegean Park Press, 1984.

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

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

[7] 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.

[8] 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.

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

[10] 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.

[11] X. Zhang, "Modified generalized integrated interleaved codes for local erasure recovery," *IEEE Commun. Lett.*, vol. 21, no. 6, pp. 1241-1244, Jun. 2017.

[12] X. Zhang, "Generalized three-layer integrated interleaved codes," *IEEE Commun. Lett.*, vol. 22, no. 3, pp. 442-445, Mar. 2018.

[13] X. Zhang and Z. Xie, "Relaxing the constraints on locally recoverable erasure codes by finite field element variation," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1680-1683, Oct. 2019.