Efficient Architectures for Generalized Integrated Interleaved Decoder

Zhenshan Xie and Xinmiao Zhang
The Ohio State University, Columbus, OH 43210, U.S.A.

***Abstract*—Generalized integrated interleaved (GII) codes nest short sub-codewords to generate parities shared by the sub-codewords. They allow hyper-speed decoding with excellent correction capability, and are essential to next-generation data storage. On the other hand, the hardware implementation of GII decoders faces many challenges, including low achievable clock frequency and large silicon area. This abstract presents novel algorithmic reformulations and architectural transformations to address each bottleneck. For an example GII code that has the same rate and length as eight un-nested (255, 223) Reed-Solomon (RS) codes, our GII decoder only has 30% area overhead compared to the RS decoder while achieving 7 orders of magnitude lower frame error rate. Its critical path only consists of 7 XOR gates, and can easily achieve more than 40GByte/s throughput.**

I. Introduction

Next-generation big data analytics, cloud storage, and high-performance computing require to access data from storage with hyper speed, such as more 40GByte/sec. Additionally, the continued technology scaling leads to deteriorating media. As a result, powerful yet hyper-speed error-correcting codes are required to ensure data reliability for next-generation storage. Such high speed is not achievable by low-density parity-check (LDPC) codes. On the other hand, Reed-Solomon (RS) or BCH codes can not meet the error-correcting performance requirement. Instead, the generalized integrated interleaved (GII) codes [1] are the best candidate.

GII codes nest a set of short RS/BCH codewords to generate codewords of stronger RS/BCH codes. Most of the time, the errors are corrected within the individual short sub-codewords. As a result, the decoder can achieve very high throughput. On the other hand, the stronger codewords generated from the nesting can be utilized to compute higher order syndromes and correct more errors when needed. Hence, much better error-correcting performance is achieved. Fig. 1 shows the frame error rates (FERs) of GII $[m, v]$ codes over $GF(2^8)$ with $m = 8$ sub-codewords and v levels of nesting compared to eight concatenated (255, 223) RS code without nesting. These codes have the same rate and length.

GII decoding has two stages. The first stage is essentially conventional RS/BCH decoding of individual sub-codewords. The second-stage nested decoding translates the higher-order syndromes of the more powerful nested codewords to additional syndromes for the sub-codewords to correct more errors. Although RS/BCH decoding has been well-studied, the nested decoding has many hardware implementation challenges. Even

This work is supported in part by Kioxia Corporation

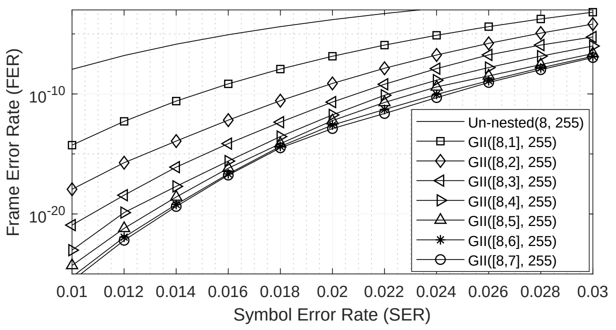

Fig. 1. FER comparisons of GII codes

though it can continue from intermediate results of the sub-codeword decoding [2], it has long critical path, which limits the achievable clock frequency, large silicon area, and low hardware utilization efficiency.

Through algorithmic reformulations and architectural modifications, each of the hardware implementation bottlenecks of GII decoders has been addressed in our recent work [3]. For an example GII [8,3] code with the same redundancy and codeword length as eight un-nested (255, 223) RS codes over $GF(2^8)$, our architectures reduce the critical path of the GII decoder from 18 to 7 gates. As a result, it can easily achieve more than 40GByte/s throughput on a Xilinx FPGA device. Our designs also reduce the area requirement by more than 50%. Compared to the RS decoder, the area overhead is less than 30% while the FER is 7 orders of magnitude lower.

II. Generalized integrated interleaved codes

Let $\mathcal{C}_v(n, k_v), \mathcal{C}_{v-1}(n, k_{v-1}) \cdots, \mathcal{C}_0(n, k_0)$ be $v+1$ RS/BCH codes over $GF(2^q)$ with $k_v \leq k_{v-1} \leq \cdots < k_0$, and their error-correcting capabilities are $t_v, t_{v-1}, \cdots, t_0$, respectively. Denote a set of m codewords of $\mathcal{C}_0$ by $c_0, c_1, \cdots, c_{m-1}$. Assume that α is a primitive element of $GF(2^q)$. The GII code is defined as [1], [2]

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

Let $y_i(x) = c_i(x) + e_i(x)$ be the i-th received sub-codeword, and $e_i(x)$ is the error vector. In the sub-codeword decoding, syndromes $S_j^{(i)} = y_i(\alpha^{j+1}) = e_i(\alpha^{j+1})$ $(0 \leq j < 2t_0)$ are first computed. From $S(x) = \sum_{j=0}^{2t_0-1} S_j x^j$, the KES, such as the Berlekamp-Massey algorithm (BMA), calculates an error locator polynomial $\Lambda(x)$ and an error evaluator polynomial $\Omega(x)$ satisfying $\Omega(x) \equiv \Lambda(x)S(x) \bmod x^{2t_0}$ using an iterative process. The inverse roots of $\Lambda(x)$ are the error locations and the error magnitudes can be computed from $\Lambda(x)$ and $\Omega(x)$.

$2t$ syndromes are needed to correct t errors. When $t > t_0$, higher-order syndromes can not be computed from y_i. Instead, since the nested codewords $\tilde{c}_l$ have higher correction capability, more syndromes can be calculated for them. Let $\tilde{y}_l(x) = \sum_{i=0}^{m-1} \alpha^{il} y_i(x)$. Then $\tilde{S}_j^{(l)} \triangleq \tilde{y}_l(\alpha^{j+1}) = \tilde{e}_l(\alpha^{j+1})$ $(2t_0 \le j < 2t_{v-l}, 0 \le l < v)$ are valid syndromes for the nested codewords. They can be converted to higher-order sub-codeword syndromes according to the nesting in (1) to correct more errors. In the KES, more iterations can be carried out to incorporate the higher-order syndromes when they become available. As a result, the KES for such nested decoding utilizing higher-order syndromes do not have to restart [2].

III. High-speed and low-complexity GII decoder architectures

The clock frequency bottleneck of the GII decoder lies in the KES and nested KES due to the iterative computations. In the r^{th} iteration of the inversionless BM algorithm [4], a discrepancy coefficient is first computed as $\delta^{(r)} = \sum \Lambda_i^{(r)} S_{r-i}$. Then $\delta^{(r)}$ is used to update $\Lambda^{(r)}(x)$ so that $\sum \Lambda_i^{(r+1)} S_{r-i}$ becomes zero. The $\delta^{(r)}$ computation needs a large multiplier-adder tree. Such a tree and the data dependency lead to long critical path. Alternatively, $\delta^{(r)}$ can be interpreted as the r^{th} coefficient of $\Lambda^{(r)}(x)S(x)$. The reformulated inversionless BM (riBM) algorithm [5] initializes a discrepancy polynomial as $\hat{\Delta}^{(0)}(x) = S(x)$. It is updated with $\Lambda(x)$ as

$$\begin{aligned} \Lambda^{(r+1)}(x) &= \gamma^{(r)}\Lambda^{(r)}(x) + \hat{\Delta}_0^{(r)} x B^{(r)}(x) \\ \hat{\Delta}^{(r+1)}(x) &= \gamma^{(r)}\hat{\Delta}^{(r)}(x)/x + \hat{\Delta}_0^{(r)}\hat{\Theta}^{(r)}(x) \end{aligned} \tag{2}$$

where $\hat{\Theta}(x)$ and $B(x)$ are auxiliary polynomials and $\gamma^{(r+1)}$ is a scalar updated in each iteration. The constant coefficient of $\hat{\Delta}^{(r)}(x)$, denoted by $\hat{\Delta}_0^{(r)}$, happens to be the discrepancy $\delta^{(r)}$ needed for iteration r. Using (2), the critical path of the riBM architecture is reduced to just one multiplier and one adder. However, the higher-order syndromes derived from the nested codewords are not available in the very beginning of the KES to initialize $\hat{\Delta}^{(0)}(x)$. Therefore, the riBM algorithm can not be adopted in the KES iterations for those higher-order syndromes in the nested decoding to reduce the critical path.

Given the $\hat{\Delta}^{(2t_0)}(x)$ from the end of the riBM algorithm for sub-codeword KES, the discrepancy coefficients for later iterations can be rewritten as [3]

$$\begin{aligned} \Delta'^{(2t_0)}_{2t_0} &= (\Delta^{(2t_0)}_{2t_0} + \Lambda^{(2t_0)}_0 S_{2t_0}) \\ \Delta'^{(2t_0)}_{2t_0+1} &= (\Delta^{(2t_0)}_{2t_0+1} + \Lambda^{(2t_0)}_1 S_{2t_0}) + \Lambda^{(2t_0)}_0 S_{2t_0+1} \\ \Delta'^{(2t_0)}_{2t_0+2} &= (\Delta^{(2t_0)}_{2t_0+2} + \Lambda^{(2t_0)}_2 S_{2t_0}) + \Lambda^{(2t_0)}_1 S_{2t_0+1} + \Lambda^{(2t_0)}_0 S_{2t_0+2} \\ &\vdots \end{aligned}$$

Only one discrepancy coefficient is needed in each iteration for the nested KES starting from $\delta^{(2t_0)} = \Delta'^{(2t_0)}_{2t_0}$. Hence, the higher-order syndromes $S_{2t_0}, S_{2t_0+1}, \cdots$ can be incorporated one by one. Considering that $\Lambda(x)$ also updates with the iterations, the nested KES can be reformulated as [3]

$$\begin{aligned} \Lambda^{(r+1)}(x) &= \gamma^{(r)}\Lambda^{(r)}(x) + \hat{\Delta}_0^{(r)} x B^{(r)}(x) \\ \hat{\Delta}^{(r+1)}(x) &= \gamma^{(r)}(\hat{\Delta}^{(r)}(x)/x + S_{r+1}\Lambda^{(r)}(x)) \\ &\quad + \hat{\Delta}_0^{(r)}(\hat{\Theta}^{(r)}(x) + S_{r+1} x B^{(r)}(x)). \end{aligned} \tag{3}$$

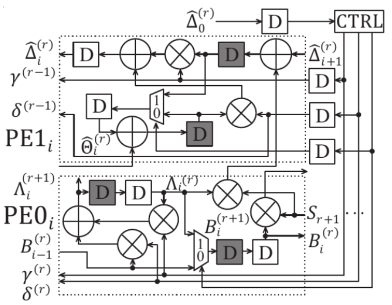

Fig. 2. Processing element of nested KES

TABLE I
Decoder complexity and performance comparisons of the example GII [8,3] code and eight un-nested (255,223) RS codes

	Proposed GII decoder	un-nested RS decoder
buffer (bits)	83200 (1.69)	49152 (1)
logic gate (# of XORs)	326912 (1.17)	278304 (1)
critical path (# of XORs)	7	7
# of clks/codeword	27	32
FER @p_s=0.02	2.09e-11	1.55e-4

The critical path of such computations consists of two multipliers and two adders, and is further reduced to one multiplier and one adder by applying the 'slow-down' and 're-timing' techniques and interleaving the decoding of two sub-codewords. Fig. 2 shows the processing element (PE) architecture of our nested KES.

Besides eliminating the clock frequency bottleneck, another three major contributions were made in our design: i) The higher-order syndrome computation is reformulated so that a unified architecture is used to implement both the nested KES and higher-order syndrome computations; ii) A modified nesting scheme utilizing subfield elements was proposed to reduce the complexities of nested syndrome computation and conversion by a significant factor; iii) Low-overhead hardware units scalable with the number of actual errors have been developed to substantially improve the hardware utilization efficiency.

Using our designs, the GII decoder silicon area has been reduced by more than 50%. Complexity comparisons with un-nested RS decoders are provided in Table I. Overall, our GII decoder has only less than 30% overhead compared to the RS decoder, while achieving 7 orders of magnitude lower FER. Even on an Xilinx UltraSCALE+ device, our design can achieve more than 550Mz clock frequency, which translates to a throughput of >40GByte/s.

References

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

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

[3] X. Zhang and Z. Xie, "Efficient architectures for generalized integrated interleaved decoder," *IEEE Trans. on Circuits and Syst.-I*, 2019.

[4] I. S. Reed, M. T. Shih, and T. K. Truong, "VLSI design of inverse-free Berlekamp-Massey algorithm," *IEE Proc. E-Computers and Digital Techniques*, vol. 138, no. 5, pp. 295-298, Sep. 1991.

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