AREA EFFICIENT PARALLEL DECODER ARCHITECTURE FOR LONG BCH CODES

Yanni Chen and Keshab K. Parhi

Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455 USA

ABSTRACT

Long BCH codes achieve additional coding gain of around 0.6dB compared to Reed-Solomon codes with similar code rate used for long-haul optical communication systems. For our considered parallel decoder architecture, a novel group matching scheme is proposed to reduce the overall hardware complexity of both Chien search and syndrome generator units by 46% for BCH(2047, 1926, 23) code as opposed to only 22% if directly applying the iterative matching algorithm. The proposed scheme exploits the substructure sharing within a finite field multiplier (FFM) and among groups of FFMs.

1. INTRODUCTION

Forward-error correction codes used in long-haul optical communication systems should provide significant coding gains (error floor can only occur at much lower bit error rate (BER), such as 10^{-15}) with high code rate and moderate complexity. In International Telecommunication Union (ITU-T) G.975, the (255, 239) Reed-Solomon (RS) code has been standardized to resist burst errors for optical fiber submarine cable systems [1]. With only 7% overhead, this RS code can not only provide approximately 5.5dB coding gain at the BER of 10^{-12} for random errors correction, but also correct bursts of length up to 64 bit [2].

BCH and RS codes form the core of the most powerful known algebraic codes and are widely used [3]. From our simulation using hard decision errors-only decoding under AWGN channel, additional coding gain of approximately 0.6dB is observed for binary BCH codes compared to RS codes with similar code rate and codeword length. Hence, BCH code and its decoder architecture are of great interest.

To increase the decoding throughput, a parallel decoder is derived by developing parallel architectures for various building blocks. Among the three major building blocks in the syndrome-based BCH decoder, i.e., syndrome generator unit, key equation solver and the Chien search, the parallel Chien search block is the most area consuming unit according to [2]. It occupies more than 65% of logic core for both 10- and 40-Gb/s forward error correction devices. Therefore, how to develop an area efficient parallel Chien search circuit for high throughput BCH decoders is of great interest and is considered in this paper. Then the area efficient scheme is applied to syndrome generator unit as well.

This paper is organized as follows. In Section 2, the decoding performance of long BCH codes is presented and compared to RS codes. In Section 3, we briefly review the implementation of three major building blocks of the BCH decoder. Section 4 is devoted to the area efficient schemes to significantly reduce the complexity of parallel Chien search architecture as well as syndrome generator units. Section 5 provides the conclusions.

This work was supported by the Army Research Offi ce under grant number DA/DAAD19-01-1-0705.

2. HIGH RATE BCH CODES VERSUS REED-SOLOMON CODES

For the purpose of performance comparison, the BCH and RS codes with similar code rate as listed in Table 1 are considered. Here code parameters (n, k, d) represent codeword length, information length and minimum distance, respectively.

Table 1. Considered high-rate long BCH and RS codes

code parameters (n, k, d)	code rate	SNR (dB) at the BER of 10^{-5}
BCH(2047,1926,23)	0.941	6.4
BCH(8191,7684,79)	0.938	6.0
RS(255,239,17)	0.937	7.0
RS(1023,959,65)	0.937	6.6

Under AWGN channel using BPSK and hard decision errors-only decoding, the performance curves for the considered high rate codes are depicted in Fig. 1.

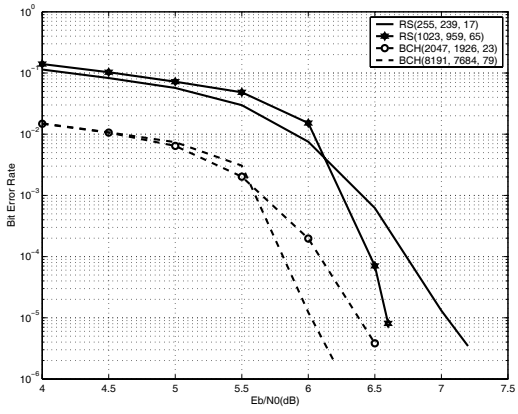

Fig. 1. Performance curves for high rate block codes

From Fig. 1, it is easily observed that BCH codes achieve slightly better performance compared to RS codes with similar code rate and codeword length. An additional coding gain of 0.6dB at the BER of 10^{-5} is achieved for BCH(2047, 1926, 23) compared to RS(255, 239, 17) code. Likewise, 0.6dB coding gain is also seen for BCH(8191, 7684, 79) compared to RS(1023, 959, 65). Due to its better decoding performance, only BCH decoder design will be considered in later sections.

3. BCH DECODER ARCHITECTURE

In this section, a parallel BCH decoder is presented.

The syndrome-based BCH decoding consists of three major steps [3], as depicted in Fig. 2, where R is the hard decision of received information from noisy channel and D is the decoded codeword. S and Λ represent syndromes of the received polynomial and error locator polynomial, respectively.

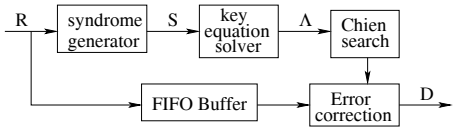

Fig. 2. Block diagram of BCH decoder

3.1. Syndrome Generator

For t-error-correcting BCH codes, $2t$ syndromes of the received polynomial could be evaluated as follows:

$$S_j = R(\alpha^j) = \sum_{i=0}^{n-1} R_i(\alpha^j)^i \tag{1}$$

for $1 \leq j \leq 2t$. If $2t$ conventional syndrome generator units shown in Fig. 3(a) are used at the same time independently, n clock cycles are necessary to complete computing all the $2t$ syndromes. However, if each syndrome generator unit in Fig. 3(a) is replaced by a parallel syndrome generator unit with parallel factor of p depicted in Fig. 3(b), which can process p bits per clock cycle, only $\lfloor n/p \rfloor$ clock cycles are sufficient.

It is worth noting that for binary BCH codes, even-indexed syndromes are the squares of earlier-indexed syndromes, i.e., $S_{2j} = S_j^2$. Based on this constraint, actually only t parallel syndrome generator units are required to compute the odd-indexed syndromes, followed by a much simpler field square circuit to generate those even-indexed syndromes.

3.2. Key Equation Solver

Either Peterson's or Berlekamp-Massey (BM) algorithm [3] could be employed to solve the key equations for $\Lambda(x)$. Inversion-free BM algorithm and its efficient implementations could be easily found in the literature [2] [4] and are not considered in this paper.

3.3. Chien Search

Once $\Lambda(x)$ is found, the decoder searches for error locations by checking whether $\Lambda(\alpha^i) = 0$ for $0 \leq i \leq (n-1)$, which is normally achieved by Chien search. A conventional serial Chien search architecture is shown in Fig. 4, and

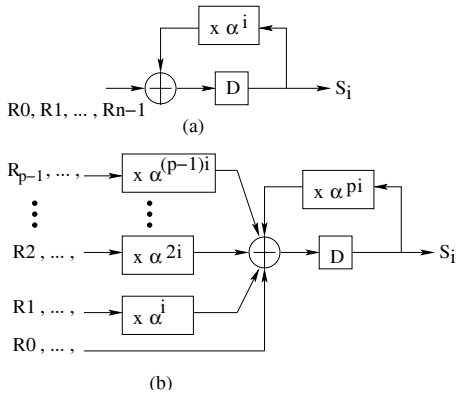

Fig. 3. Syndrome generator unit (a) Conventional architecture (b) Parallel architecture with parallel factor of p

$$\Lambda(\alpha^i) = \sum_{j=0}^{t} \Lambda_j \alpha^{ij} = \sum_{j=1}^{t} \Lambda_j \alpha^{ij} + 1 \tag{2}$$

where $0 \leq i \leq (n-1)$. All the multiplexers select $\Lambda(x)$ in the first clock cycle, then select the registered data afterwards.

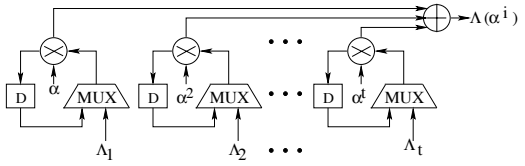

Fig. 4. Conventional Chien search circuit

Since all the n possible locations have to be evaluated for the $\Lambda(x)$, it takes n clock cycles to complete the Chien search process. To speed up this process, parallel Chien search architecture that evaluates several locations per clock cycle is essential. Two different possible architectures with parallel factor p are depicted in Fig. 5(a) [2] and Fig. 5(b) [4], where Fig. 5(a) actually is just a direct unfolded version of Fig. 4 with an unfolding factor of p.

As both designs in Fig. 5 can reduce the number of clock cycles searching for error locations from n down to $\lfloor n/p \rfloor$, they also share similar hardware complexity. Denoting the parallel factor as p, both designs have the exactly same $(p \times t)$ constant finite field multipliers (FFM), p t-input m-bit finite field adders (FFA), p m-bit registers and p m-bit multiplexers. However, the critical path of Fig. 5(a) is $(Tmux + p \times Tm + Ta)$ while it is only $(Tmux + Tm + Ta)$ for Fig. 5(b), where $Tmux$, Tm and Ta stand for the critical path of multiplexer, FFM and t-input m-bit FFA, respectively. Obviously, once the parallel factor p is greater than 1, much faster clock speed could be achieved for the design in Fig. 5(b) than that in Fig. 5(a). For example, assuming T_m is dominant, critical path of Fig. 5(b) is p times shorter.

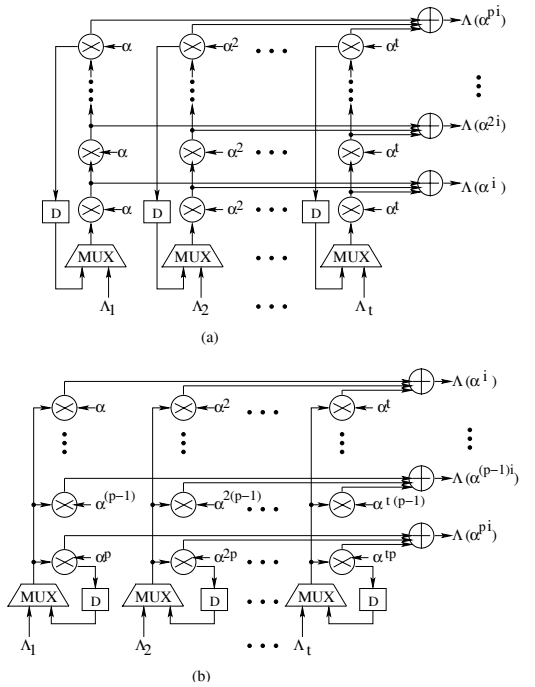

Fig. 5. Two different p-parallel Chien search architectures (a) direct unfolded version (b) equivalent architecture with shorter critical path

4. COMPLEXITY REDUCTION SCHEME

In this section, a complexity reduction scheme to eliminate the redundant computations of FFM is discussed in detail.

An optimization algorithm developed in [5] can reduce the number of XOR gates for constant FFM operations by up to 40% compared to straightforward implementation. In this paper, a different algorithm called *iterative matching algorithm* (IMA) is attempted to reduce the area. The main idea is to use iterative sub-structure sharing to eliminate the redundant computations.

4.1. Iterative Matching Algorithm

Consider a constant multiplication in $GF(2^m)$ where P is the product of fixed operand α^i, where $1 \leq i \leq t$ for the design in Fig. 5(a) and $1 \leq i \leq (t \times p)$ for the design in Fig. 5(b), and variable field element B:

$$\begin{aligned} P &= \alpha^i B = \alpha^i (b_0 + b_1\alpha + \ldots + b_{m-1}\alpha^{m-1}) \qquad (3) \\ &= b_0\alpha^i + b_1\alpha^{i+1} + \ldots + b_{m-1}\alpha^{i+m-1} \\ &= \begin{pmatrix} \alpha_0^i & \alpha_0^{i+1} & \ldots & \alpha_0^{i+m-1} \\ \alpha_1^i & \alpha_1^{i+1} & \ldots & \alpha_1^{i+m-1} \\ \vdots & \vdots & \vdots & \\ \alpha_{m-1}^i & \alpha_{m-1}^{i+1} & \ldots & \alpha_{m-1}^{i+m-1} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \end{pmatrix} \\ &= [\alpha_{coeff}]_{m\times m} * [b]_{m\times 1} \end{aligned}$$

Since all the elements α_l^{i+j} in the matrix α_{coeff}, where $0 \leq l, j \leq (m-1)$, are simple binary elements and all the additions are modulo 2 operations, the computational complexity of FFM can also be defined by the number of XOR gates. Obviously, while computing all the m coefficients of the product $P, p_0, p_1, \ldots, p_{m-1}$, all of which are linear combinations of B coefficients, there are many redundant modulo 2 additions that allow a reduction of the number of operations.

Different from the algorithm in [5], our iterative matching algorithm based on [6] consists of following four basic steps.

1. Determine the number of bit-wise matches (nonzero bits) between all of the rows in the binary matrix α_{coeff};
2. Choose the best match;
3. Eliminate the redundancy from the best match; Return the remainders to the two rows that contribute the best match; Append an additional row at the bottom of the binary matrix to hold the redundancy;
4. Repeat steps 1-3 for all the rows in the binary matrix including the appended rows until no improvement is achieved, i.e., the best match is not greater than 1 bit.

4.2. Implementation Results and Group Matching

By applying the IMA to both Chien search architectures in Fig. 5, the implementation results in terms of number of XOR gates for BCH(2047, 1926, 23) code is listed in Table 2.

Table 2. Chien search complexity for BCH(2047, 1926, 23) with parallel factor of 32

implementation methods	design in Fig. 5(a)	design in Fig. 5(b)	area saving
straightforward implementation	6080	16624	0%
IMA within each FFM	5984	12257	26%
IMA among 4 FFMs	––	8468	49%
IMA among 8 FFMs	__	7598	54%
IMA among 16 FFMs	__	7058	58%
IMA among 32 FFMs	__	6653	60%

In Table 2 IMA is not only explored within individual FFMs, but also among g FFMs, where g is the group size and $1 < g \leq p$, in the same column (see Fig. 5). These FFMs share the same multiplicand Λ_i, where $1 \leq i \leq t$. This latter case is called *group matching* among *g* FFMs and g binary coefficient matrices α_{coeff} of g constant FFMs are combined together to search for the best match. In other words, the bit-wise matches are searched in a coefficient matrix with size of $gm \times m$ instead of $m \times m$. Note that group matching is solely possible for the design in Fig. 5(b), which provides another advantage in addition to the lower critical path compared to that in Fig. 5(a).

If implemented in a straightforward manner, the design in Fig. 5(a) has much smaller complexity than that in Fig. 5(b) simply because in the latter case the constant FFMs have more powers of α with higher Hamming weight as multiplicands. However, as the group matching factor g is increased, the number of

XOR gates is reduced significantly for the design in Fig. 5(b). When g is equal to the parallel factor 32, compared to the straightforward implementation, the area saving is 60% as opposed to merely 26% saving obtained if the iterative matching algorithm is applied to individual FFMs. Consequently, the complexity of Fig. 5(b) is very close to that of Fig. 5(a). Furthermore, the former design retains its shorter critical path advantage.

For longer BCH(8191, 7684, 79) code (see Table 3), the complexity for the design in Fig. 5(a) grows very quickly as its error correcting capability t is increased to 39. However, complexity increases only slightly for the design in Fig. 5(b). Moreover, the area saving for the design in Fig. 5(b) after employing group matching is more significant compared to BCH(2047, 1926, 23) code. In fact its number of XOR gates is already smaller than that of the design in Fig. 5(a) even when the group matching is carried out among 4 FFMs. While the group matching factor g is increased to 32, the complexity of design in Fig. 5(b) is approximately 30% less than that of the design in Fig. 5(a). This implies that for longer BCH codes, lower complexity and faster design could be achieved by applying the proposed group matching scheme.

Table 3. Chien search complexity for BCH(8191, 7684, 79) with parallel factor of 32

implementation methods	design in Fig. 5(a)	design in Fig. 5(b)	area saving
straightforward implementation	89664	102277	0%
IMA within each FFM	57504	63997	37%
IMA among 4 FFMs	––	51029	50%
IMA among 8 FFMs	__	46329	55%
IMA among 16 FFMs	__	42534	58%
IMA among 32 FFMs	__	39365	62%

4.3. Apply Group Matching to Both Chien Search and Syndrome Generator Units

In a similar manner, the group matching scheme described above can also be applied for the constant FFMs in the feedback loop of syndrome generator units in Fig. 3(b) to reduce the number of XOR gates. Since one of the multiplicands in the feed-forward FFMs is simple a binary number, no multiplication is performed and hence no complexity reduction is needed for those FFMs. The combined complexity of Chien search and syndrome generator units is listed in Table 4.

In Table 4 the results for group matching are obtained by applying the IMA among 32 FFMs for Chien search and individual matching for parallel syndrome generator units. From Table 4 we can observe that the complexity of parallel syndrome generator units is dominated by the p-input m-bit FFAs instead of FFM in the feedback loop, which explains why the area saving is small for syndrome part. However, for BCH(2047, 1926, 23) code, 46% area is saved for the combined complexity. This is a significant improvement compared to the case of directly applying IMA, which only saves 22% XOR gates. Similar results are observed for BCH(8191, 7684, 79) code.

Table 4. Combined complexity for both Chien search and syndrome units with parallel factor of 32

code parameters	implementation methods	Chien Search	Syndrome	Combined
BCH (2047, 1926, 23)	straightforward implementation	16624	4372	20996
	after individual matching	12257	4181	16438
	area saving	26%	4%	22%
	after group matching	6653	4181	10834
	area saving	60%	4%	46%
BCH (8191, 7684, 79)	straightforward implementation	102277	18933	121210
	after individual matching	63997	17733	81730
	area saving	37%	6%	33%
	after group matching	39365	17733	57098
	area saving	62%	6%	53%

5. CONCLUSIONS

In this paper, to reduce the area consumption of binary high throughput BCH decoder, a novel complexity reduction scheme is proposed. As a result, the parallel decoder architecture can reduce the number of XOR gates by roughly 50% compared to the original design. It also shows a significant improvement to the previous results where the iterative matching algorithm was applied within individual FFMs. Consequently, an area-efficient design with very short critical path is obtained. All the techniques presented in this paper can be easily extended to RS codes.

6. REFERENCES

[1] "Forward error correction for submarine systems," Telecommunication standardization section, International Telecommunication Union, G.975, 1996.

[2] L. Song, M.-L. Yu, and M. S. Shaffer, "10- and 40-Gb/s forward error correction devices for optical communications," *IEEE J. Solid-State Circuits*, vol. 37, pp. 1565-1573, Nov. 2002.

[3] S. B. Wicker, *Error control systems for digital communication and storage*, Upper saddle river: NJ: Prentice-Hall, 1995.

[4] H. C. Chang, C. B. Shung, and C. Y. Lee, "A Reed-Solomon Product-Code (RS-PC) decoder chip for DVD applications," *IEEE J. Solid State Circuits*, vol. 36, pp. 229-238, Feb. 2001.

[5] C. Paar, "Optimized arithmetic for Reed-Solomon encoders," *IEEE Proc. of ISIT'97*, pp. 250, 1997.

[6] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 2, pp. 151-165, Feb. 1996.