PAPER

High-Speed Two-Parallel Concatenated BCH-Based Super-FEC Architecture for Optical Communications*

Sangho YOON[†], ***Nonmember***, **Hanho LEE**[†a)], ***Member***, ***and*** **Kihoon LEE**[†], ***Nonmember***

SUMMARY This paper presents a high-speed Forward Error Correction (FEC) architecture based on concatenated Bose-Chaudhuri-Hocquenghem (BCH) for 100-Gb/s optical communication systems. The concatenated BCH code consists of BCH(3860, 3824) and BCH(2040, 1930), which provides 7.98 dB net coding gain at 10^{-12} corrected bit error rate. The proposed BCH decoder features a low-complexity key equation solver using an error-locator computation RiBM (ECRiBM) algorithm and its architecture. The proposed concatenated BCH-based Super-FEC architecture has been implemented in 90-nm CMOS standard cell technology with a supply voltage of 1.1 V. The implementation results show that the proposed architecture can operate at a clock frequency of 400 MHz and has a throughput of 102.4-Gb/s for 90-nm CMOS technology.

key words: *concatenated BCH code, FEC, architecture, optical*

1. Introduction

The Bose-Chaudhuri-Hocquenghem (BCH) codes are a class of powerful multiple error-correcting cyclic codes [1]. The BCH codes are kinds of cyclic codes, which are used in a broad class of error correcting codes such as optical fiber communication systems, second generation Digital Video Broadcasting (DVB-S2) standard and digital communication systems. The Reed-Solomon (RS) (255, 239) code is used and standardized in ITU-T G.975 and G.709 [2]. This code has 5.6 dB net coding gain at 10^{-12} decoder output bit error rate (BER) with 6.69% redundancy. For high-speed (40-Gb/s and beyond) optical fiber communication systems, more powerful forward error correction (FEC) codes have became necessary in order to achieve higher correction ability than RS(255, 239) code and compensate for serious transmission quality degradation. Thus, several Super-FEC schemes are considered and recommended in [2]. For example, Super-FEC that uses a combination of two FEC codes is well known, such as [RS code + RS code], [BCH code + BCH code], and [RS code + BCH code].

The concatenated BCH code described in G. 975.1 [2] uses the BCH(3860,3824) and BCH(2040,1930) codes for outer code and inner code, respectively. A BCH code has more powerful random error correction capability over the additive white Gaussian noise (AWGN) channel than a RS code with similar rate and codeword length. Furthermore, the concatenated BCH code applies three times iterative decoding which provides 7.98 dB net coding gain at 10^{-12} decoder output BER without additive redundancy compared to the RS(255,239) code. This technique can improve the error correction capability without increasing the coding rate. Figure 1(a) shows block diagrams of the concatenated BCH Super-FEC architecture which consists of BCH encoders, BCH decoders and interleaver/deinterleavers [2]. Each BCH(3860, 3824) and BCH(2040, 1930) code can correct up to 3 and 10 bit errors of one codeword, respectively. A BCH decoders can be implemented using the Berlekamp-Massey (BM) algorithm or modified Euclidean (ME) algorithm to solve the key equation. These algorithms are typically designed using theory for finite-field (also called Galois-field) arithmetic operations. For either the BM algorithm or ME algorithm can be used to solve a key equation for an error locator polynomial $\lambda(r)$ of BCH decoding procedure.

The very high-speed data transmission techniques that have been developed for the optical fiber communication systems have necessitated the implementation of high-speed FEC architectures to meet the continuing demands for ever higher data rates [3]–[5]. Specifically, standardization of a hard-decision FEC for 100 Gb/s optical transport network (OTN) is discussing at the ITU-T. As a result, the RS(255,239) code has become mandatory for short-reach systems. But, no specific FEC has been determined yet for metro and long-haul systems. Also, the concatenated BCH FEC decoder architecture proposed in [4] is difficult to achieve 100 Gb/s throughput in its present form.

In this paper, we present a high-speed two-parallel concatenated BCH Super-FEC architecture for 100 Gb/s optical communication systems. Also, a novel error-locator computation RiBM (ECRiBM) algorithm and its architecture are proposed with the aim of reducing the hardware complexity and improving the clock frequency for high-speed low-complexity BCH decoders. This design has a lower number of processing elements (PEs) compared to the conventional RiBM architecture. Therefore, it has a less number of Galois-Field (*GF*) multipliers and *GF* adders, which provide considerably reduced hardware complexity.

Section 2 shows the proposed concatenated BCH decoder with two-parallel processing. Section 3 shows the proposed ECRiBM algorithm and its architecture for the high-speed low-complexity BCH decoders. In Sect. 4, results and

Manuscript received August 6, 2009.
Manuscript revised December 1, 2009.

[†]The authors are School of Information and Communication Engineering, Inha University, Incheon, 402-751, Republic of Korea.

[*]This reseach was partly supported by the MKE, Korea, under the ITRC support program supervised by the (IITA-2009-C1090-0902-0019), and partly supported by the IT R&D program of the MKE/KEIT, [2009-F-010-01].

a) E-mail: hhlee@inha.ac.kr

DOI: 10.1587/transfun.E93.A.769

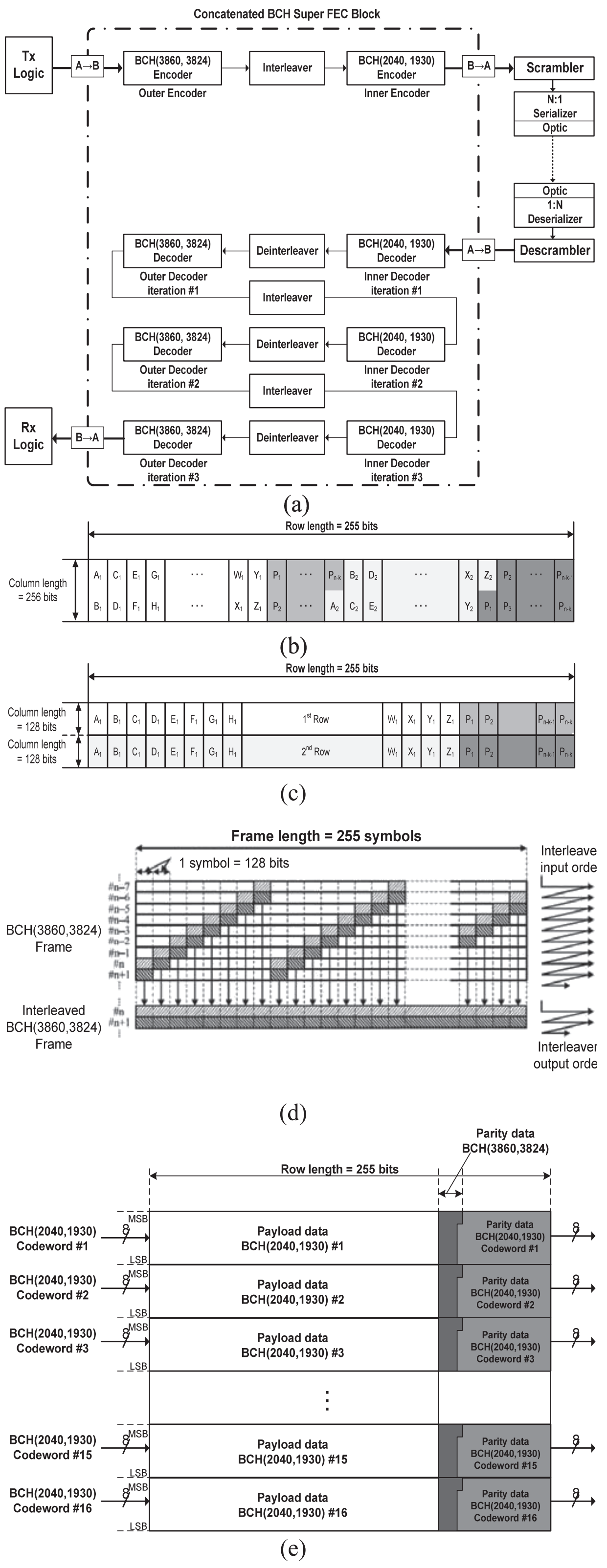

Concatenated BCH Super FEC Block
Tx Logic
A→B
BCH(3860, 3824) Encoder
Outer Encoder
Interleaver
BCH(2040, 1930) Encoder
Inner Encoder
B→A
Scrambler
N:1 Serializer
Optic
Optic
1:N Deserializer
Descrambler
BCH(3860, 3824) Decoder
Outer Decoder iteration #1
Deinterleaver
BCH(2040, 1930) Decoder
Inner Decoder iteration #1
Outer Decoder iteration #2
Inner Decoder iteration #2
Outer Decoder iteration #3
Inner Decoder iteration #3
Rx Logic
(a)
Row length = 255 bits
Column length = 256 bits
(b)
Column length = 128 bits
1st Row
2nd Row
(c)
Frame length = 255 symbols
1 symbol = 128 bits
Interleave input orde
BCH(3860,3824) Frame
Interleaved BCH(3860,3824) Frame
Interleaver output orde
(d)
Parity data BCH(3860,3824)
BCH(2040,1930) Codeword #1
Payload data BCH(2040,1930) #1
Parity data BCH(2040,1930) Codeword #1
MSB
LSB
BCH(2040,1930) Codeword #16
Payload data BCH(2040,1930) #16
Parity data BCH(2040,1930) Codeword #16
(e)

Fig. 1 (a) Two-parallel 32-ch. concatenated BCH Super-FEC architecture, (b) A-format, frame structure of OTN, (c) B-format, converted frame structure of OTN (d) BCH(3860,3824) interleaving scheme, and (e) interleaved BCH(2040,1930) frame.

performance comparison are presented. Finally, conclusions are provided in Sect. 5.

2. Concatenated BCH Decoder with Two-Parallel Processing

The concatenated BCH decoder with two-parallel processing consists of the BCH(3860, 3824) outer decoder, BCH(2040, 1930) inner decoder, and interleaver/deinterleaver. The original OTN frame structure, namely A-format, is organized as shown in Fig. 1(b). But it is not suitable for the proposed two-parallel structure because the data alignment inside OTN frame is not parallel structure. Thus we convert the frame structure of OTN frame into two-parallel frame structure, namely B-format, as shown in Fig. 1(c). In order to convert A-format into B-format in the OTN frame structure, the format converter is used at input and output ports of the concatenated BCH decoder as shown in Fig. 1(a). With the B-format OTN frame structure, encoding and decoding are executed with the interleaving scheme as shown in Figs. 1(d), (e). The brief interleaving scheme is as follows: 8 B-format OTN frames are block-interleaved as shown in Fig. 1(d) and each frame contains 8 BCH(3860,3824) codewords. Then the block-interleaved data in each frame is aligned into 16 BCH(2040, 1930) codewords as shown in Fig. 1(e). Detailed description of the interleaving scheme is discussed in [2].

Figure 2 shows the block diagrams of the proposed two-parallel BCH decoders. Figure 2(a) shows the block diagram of the outer decoder, which processes 16 interleaved BCH(3860, 3824) codewords simultaneously. Each upper and lower 128 bits processes 8 B-format OTN frame according to interleaving scheme proposed in [2]. The outer decoder has 16 syndrome computation (SC) blocks, 1 shared key-equation solver (KES) block based on ECRiBM algorithm, 16 chien search and error correction blocks. Each block processes 16 parallel bits and calculates on a $GF(2^{12})$ symbolic base. Figure 2(b) shows the block diagram of the inner decoder, which processe 32 interleaved BCH(2040, 1930) codewords simultaneously. The inner decoder has 32 syndrome computation blocks, 3 shared KES blocks based on ECRiBM algorithm, 32 chien search and error correction blocks. Each block processes 8 parallel bits and calculates on a $GF(2^{11})$ symbolic base. Both the inner and outer decoder has a throughput of 256 bits per one clock cycle.

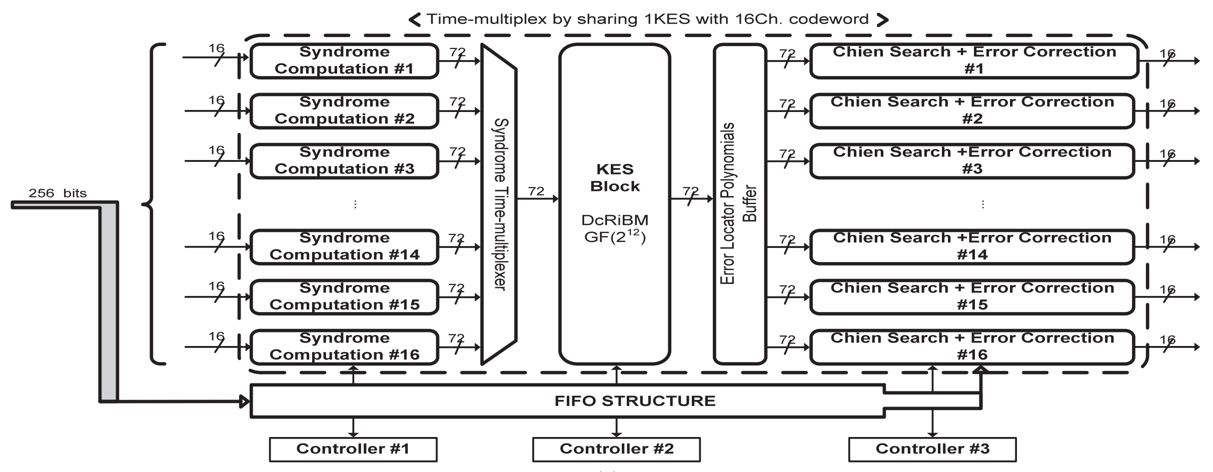
2.1 Syndrome Computation Block

The syndrome computation block calculates all the syndromes S_i $(0 \le i \le 2t-1)$ by putting the roots of generator polynomial $G(x)$ into the received codeword polynomial $R(x)$ as shown in Eqs. (1) and (2).

$$R(x) = r_{254}x^{254} + r_{253}x^{253} + r_{252}x^{252} + \cdots + r_1 x + r_0 \quad (1)$$

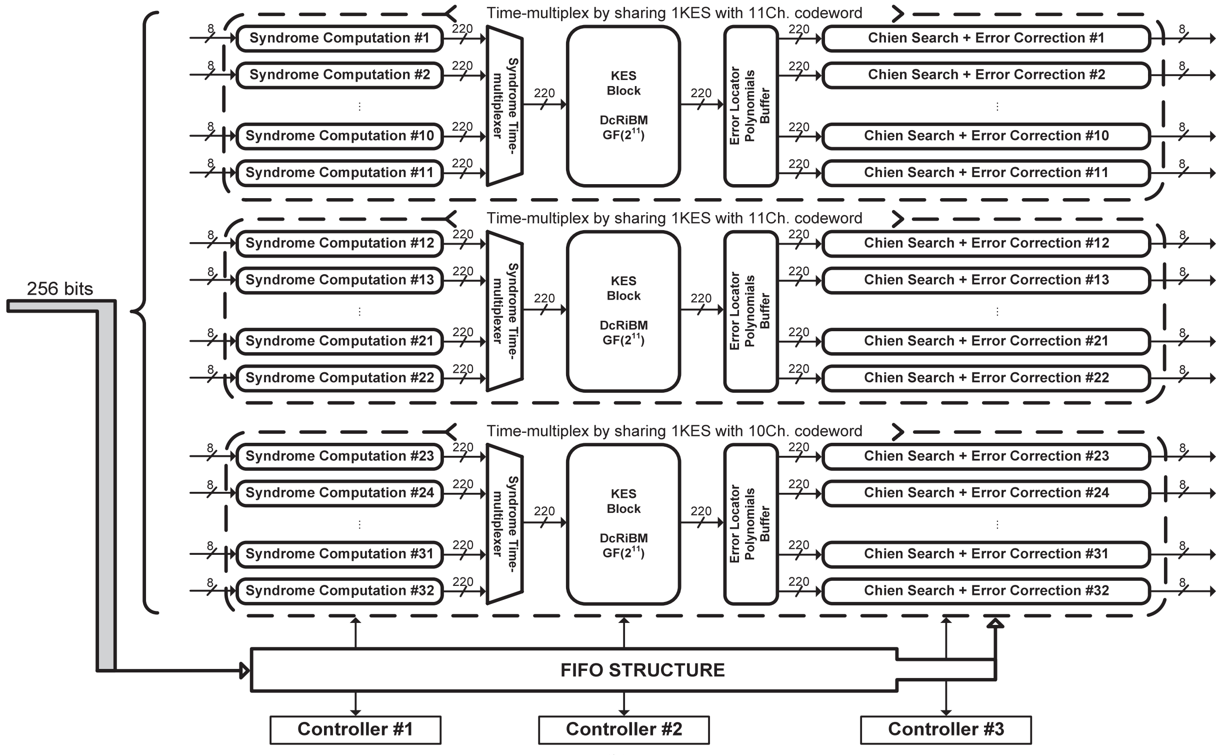
$$S i = r_{254}\alpha^{i*254} + r_{253}\alpha^{i*253} + r_{252}\alpha^{i*252} + \cdots + r_1\alpha^{i} + r_0 \quad (2)$$

Figure 3(a) shows SC block, in which all calculations



Time-multiplex by sharing 1KES with 16Ch. codeword
256 bits
Syndrome Computation #1
Syndrome Computation #2
Syndrome Computation #3
Syndrome Computation #14
Syndrome Computation #15
Syndrome Computation #16
Syndrome Time-multiplexer
KES Block
DcRiBM
GF(2^12)
Error Locator Polynomials Buffer
Chien Search + Error Correction #1
Chien Search + Error Correction #2
Chien Search +Error Correction #3
Chien Search +Error Correction #14
Chien Search + Error Correction #15
Chien Search + Error Correction #16
FIFO STRUCTURE
Controller #1
Controller #2
Controller #3
16
72

(a)



Time-multiplex by sharing 1KES with 11Ch. codeword
Time-multiplex by sharing 1KES with 11Ch. codeword
Time-multiplex by sharing 1KES with 10Ch. codeword
256 bits
Syndrome Computation #1
Syndrome Computation #2
Syndrome Computation #10
Syndrome Computation #11
Syndrome Computation #12
Syndrome Computation #13
Syndrome Computation #21
Syndrome Computation #22
Syndrome Computation #23
Syndrome Computation #24
Syndrome Computation #31
Syndrome Computation #32
Syndrome Time-multiplexer
KES Block
DcRiBM
GF(2^11)
Error Locator Polynomials Buffer
Chien Search + Error Correction #1
Chien Search + Error Correction #2
Chien Search + Error Correction #10
Chien Search + Error Correction #11
Chien Search + Error Correction #12
Chien Search + Error Correction #13
Chien Search + Error Correction #21
Chien Search + Error Correction #22
Chien Search + Error Correction #23
Chien Search + Error Correction #24
Chien Search + Error Correction #31
Chien Search + Error Correction #32
FIFO STRUCTURE
Controller #1
Controller #2
Controller #3
8
220

(b)

Fig. 2 Block diagrams of 32-ch. (a) BCH(3860, 3824) outer decoder, (b) BCH(2040, 1930) inner decoder.

are $GF(2^m)$ symbol-based where m is 11 and 12 bits for the inner and outer decoder, respectively. But SC block receives byte-based codeword, such as 16 and 8 bits in the outer and inner decoder respectively. So, we need to transform input bits into the symbol of $GF(2^m)$, which is named "Bit2Sym" for convenience sake.

2.2 Key Equation Solver Block

The key equation solver block is used to obtain the error locator polynomial $\sigma(x)$ by solving the key equation $S(x)\sigma(x) = \sigma_{odd}(x)\ mod\ G(x)$, where $\sigma_{odd}(x)$ is the odd terms of $\sigma(x)$. A well known reformulation of inversionless Berlekamp-Massey (RiBM) algorithm [6] calculates the error locator polynomials and the error evaluator polynomial through the iterative procedure for solving the key equation. This algorithm is used to compute the current polynomial update. The architecture based on RiBM algorithm has extremely regular systolic structure with the critical path only $(T_{mult} + T_{add})$ as compared to the critical path of previous iBM algorithm $(> 2 \cdot (T_{mult} + T_{add}))$. Moreover, the RiBM architecture also supports lower gate complexity and simpler control structure as compared to architectures based on the ME algorithm. But this architecture uses a lot of GF multipliers

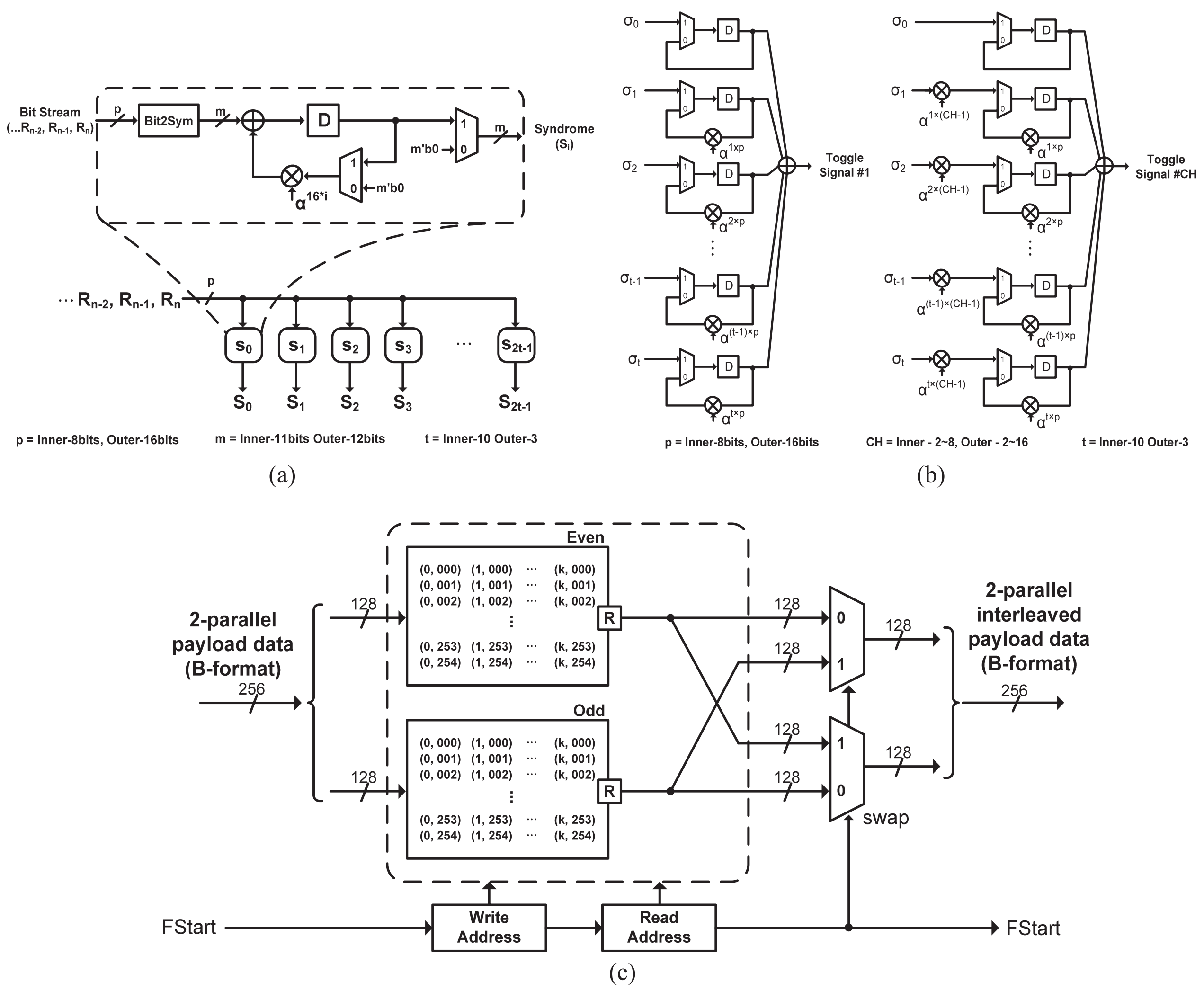

Bit Stream (...Rn-2, Rn-1, Rn)
p
Bit2Sym
m
D
m'b0
Syndrome (Si)
α16*i
... Rn-2, Rn-1, Rn
S0 S1 S2 S3 ... S2t-1
p = Inner-8bits, Outer-16bits
m = Inner-11bits Outer-12bits
t = Inner-10 Outer-3
(a)
σ0 σ1 σ2 σt-1 σt
Toggle Signal #1
Toggle Signal #CH
α1×p α2×p α(t-1)×p αt×p
α1×(CH-1) α2×(CH-1) α(t-1)×(CH-1) αt×(CH-1)
p = Inner-8bits, Outer-16bits
CH = Inner - 2~8, Outer - 2~16
t = Inner-10 Outer-3
(b)
2-parallel payload data (B-format)
256
128
Even
Odd
(0, 000) (1, 000) ··· (k, 000)
(0, 001) (1, 001) ··· (k, 001)
(0, 002) (1, 002) ··· (k, 002)
(0, 253) (1, 253) ··· (k, 253)
(0, 254) (1, 254) ··· (k, 254)
R
swap
2-parallel interleaved payload data (B-format)
FStart
Write Address
Read Address
FStart
(c)

Fig. 3 Block diagrams of (a) Syndrome computation block, (b) Chien search block, and (c) two-parallel 32-ch. interleaver/deinterleaver.

and *GF* adders, which occupy extensive silicon area and results in high power consumption. Thus, we propose a novel ECRiBM algorithm and its architecture with the aim of reducing the hardware complexity for the BCH decoders [7]. This design does not require the error-value computation unit to compute the error evaluator polynomial and has a lower number of processing element (PE) processors compared to the conventional RiBM architecture. Therefore, it has fewer *GF* multipliers and *GF* adders, which provide considerably reduced hardware complexity. Also for t $\leq$ 3, Peterson-Gorenstein-Zierler (PGZ) architecture is preferred due to its low hardware complexity [8]. However, for t=3, our design uses lower number of *GF* multipliers and *GF* adders compared with the PGZ architecture in [8]. The detailed explanation for the proposed ECRiBM algorithm and architecture is described in Sect. 3.

2.3 Chien Search Block

The error locator polynomial $\sigma(x)$ is obtained by the KES block. The chien search block searches for the error locations by finding roots of $\sigma(x)$. The inversed power of roots indicates the error location in the codeword. The chien search block can be parallelized using an equivalent circuit as shown in Fig. 3(b).

2.4 Interleaver/Deinterleaver

The two-parallel 32-channel interleaver/deinterleaver consist of two block memories and read/write address control block, which are used to swap two-parallel interleaved payload data with memory output data from even and odd block memories, as shown in Fig. 3(c). This two-parallel architecture needs the converter in order to parallelize two serial frames at input and output port of the B-format OTN frame structure. The required memory size for each interleaver/deinterleaver and frame converter is 32,640 bytes and 8,160 bytes, respectively.

3. Proposed Error-Locator Computation RIBM Algorithm and Architecture

3.1 Proposed Error-Locator Computation RiBM Algorithm

The odd-numbered syndromes can be computed by evaluating the received polynomials at the necessary odd-powered roots of the generator polynomial. The necessary roots of the generator polynomial comprise at least $2t$ consecutive symbol values $\alpha^{L+1}, \alpha^{L+2}, \cdots, \alpha^{L+2t}$, where α represents the primitive element and L represents an appropriate power of roots.

The Syndromes S_i for BCH codes possess a property as described by the following Eq. (3).

$$S_{2j} = (S_j)^2, \quad (j = 1, 2, 3, \cdots, t) \tag{3}$$

Equation (3) is used to generate a regular repeating control signal. The discrepancy $\tilde{\delta}(r)$ is given as the multiple of appropriate symbol value, so the odd-numbered iterative operation regularly generates value of discrepancy $\tilde{\delta}(r)$ '0' as proved in the following Eq. (4). Using this property, the look-ahead transformation can be applied and the number of iterative operations in a BCH decoding process can be half-reduced.

$$\begin{aligned} \delta(r+1)|_{r=even\,number} &= \gamma(r) \cdot \tilde{\delta}(r) - \tilde{\delta}(r) \cdot \tilde{\theta}_0(r) \\ &= \alpha^L (S_{2j} - S_j \cdot S_j) = 0 \end{aligned} \tag{4}$$

The proposed ECRiBM algorithm is described in the following pseudo-code.

```
The ECRiBM Algorithm:
Initialization:
  $\tilde{\delta}_{2t-1}(0) = \tilde{\theta}_{2t}(0) = \gamma(0) = 1$
  $\tilde{\delta}_{2t}(0) = \tilde{\delta}_{i-1}(0) = \tilde{\theta}_i(0) = 0, (i = 2t-2, 2t-1)$
Input: $S_i, (i = 0, 1, 2, \cdots, 2t-1)$
  $\tilde{\delta}_i(0) = S_{i+3}, \quad (i = 1, 2, \cdots, 2t-4)$
  $\tilde{\theta}_i(0) = S_{i+2}, \quad (i = 1, 2, \cdots, 2t-3)$
  $\tilde{\delta}_i(0) = S_2, \tilde{\theta}_i(0) = S_1, \quad (i = 0)$
  $\tilde{\delta}(0) = S_0,$
  For r = 0 step 1 until 2t−1 do
  Begin
    if (r≠0 and r≠2 and r≠odd) $\tilde{\theta}_{2t-2-r} = 0$
    if (r≠0 and r≠odd)    $\tilde{\theta}_{2t-3-r} = 0$
    if (r mod 2)
    Begin
      if $(\tilde{\delta}(r))\tilde{\delta}_i(r+1) = \tilde{\delta}_i(r)\tilde{\delta}(r) + \tilde{\theta}_i(r)\tilde{\delta}(r)$
      else $\tilde{\delta}_i(r+1) = \tilde{\delta}_i(r)\gamma(r) + \tilde{\theta}_i(r)\tilde{\delta}(r)$
      $\tilde{\theta}_i(r+1) = \vartheta_i(r)$
      $\vartheta_i(r+1) = \tilde{\delta}_i(r)$
      $\tilde{\delta}(r+1) = \tilde{\delta}_0(r+1)$
    End
    else
    Begin
      if (r == 0)
      Begin
        $\tilde{\delta}_i(r+1) = \tilde{\delta}_i(r)\gamma(r)$
        $\tilde{\vartheta}_i(r+1) = \tilde{\delta}_i(r)$
      End
      else
      Begin
        $\tilde{\delta}_i(r+1) = \tilde{\delta}_{i+1}(r)\gamma(r)$
        $\tilde{\vartheta}_i(r+1) = \tilde{\delta}_{i+1}(r)$
      End
      $\tilde{\theta}_i(r+1) = \tilde{\theta}_i(r)\tilde{\delta}(r)$
      $\tilde{\delta}(r+1) = \tilde{\delta}(r)$
    End
    if $(\tilde{\delta}(r))\gamma(r+1) = \delta(r)$
    else    $\gamma(r+1) = \gamma(r)$
  End
Output: $\lambda_i = \delta_i(2t), i = (1, 2, \cdots, t+1)$
```

3.2 Proposed Error-Locator Computation RiBM Architecture

The disadvantage of the conventional RiBM architecture described in [6] is that it uses a lot of GF multipliers and GF adders, which occupy extensive silicon area and results in high power consumption. Figure 4 shows the novel ECRiBM architecture, in which there is no error-value computation unit and it has a lower number of PE processors. By using 1-bit counter, not only PE block but also $\delta(r)$ blocks are controlled consecutively. Also the value '0' is processes instead of $\tilde{\theta}_{2t-i-r}(r)$, $(i = 2, 3)$ latch by $\lceil \log_2 t \rceil$ -bit counter to eliminate the error value computation circuit from the conventional RiBM architecture.

Note that the discrepancy $\tilde{\delta}(r)$ generated fixed PE (zero-th) position is always '0' in odd number iteration. Using this property, regular repeating control signal can reduces the control block complexity described in RiBM architecture.

$$\begin{aligned} \tilde{\delta}_i(r+2) &= \tilde{\delta}(r) \cdot \{\gamma(r) \cdot \tilde{\delta}_{i+1}(r) - \tilde{\delta}(r) \cdot \tilde{\theta}_i(r)\} \\ &= \tilde{\delta}(r) \cdot \tilde{\delta}_i(r+1) \end{aligned} \tag{5}$$

Based on the new initial conditions compared with the existing RiBM algorithm, the novel ECRiBM algorithm performs the computation of next coefficients of the reformulated Δ polynomial $\delta_i(r+2)$ as Eq. (5), where $\tilde{\delta}(r)$ is the discrepancy, which feed into every PE processor at the same time. The upper equation in Eq. (5) has critical path $2{\cdot}T_{mult} + T_{add}$, so we reformulated the upper equation in Eq. (5) to guarantee the reduction of the critical path delay $T_{mult} + T_{add} + 2 \cdot T_{mux}$.

The proposed ECRiBM architecture has almost similar critical path delay with the conventional RiBM architecture by reformulating Eq. (5). Furthermore, the proposed ECRiBM architecture has systolic array structure, in which PE processor consists of $2GF$ multipliers, $1GF$ adder, 2

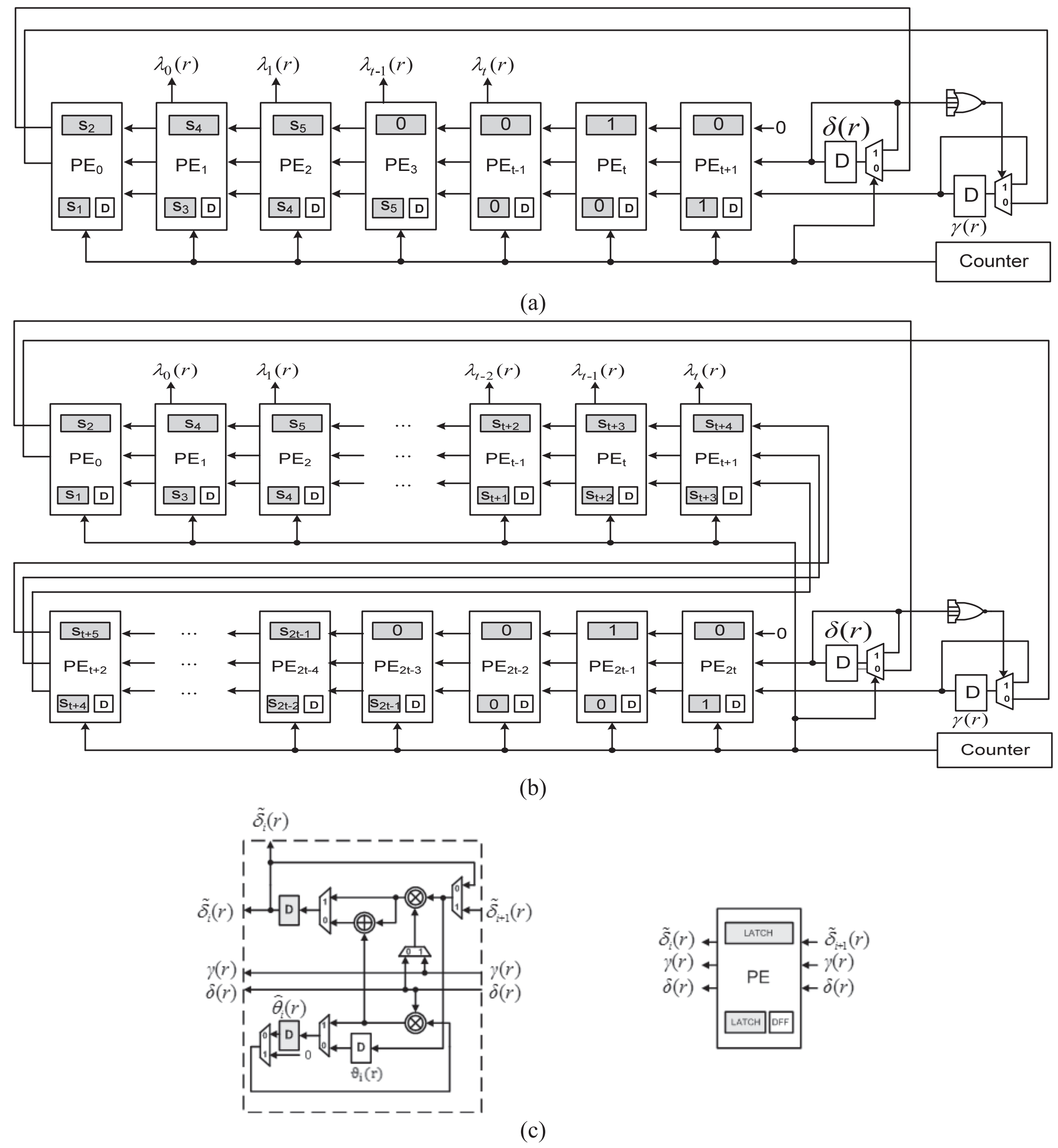


Fig. 4 Block diagrams of error-locator computation RiBM (ECRiBM) architecture for (a) BCH(3860, 3824) decoder, (b) BCH(2040, 1930) decoder, and (c) block diagram of PE processor.

latches, 1 D flip-flop and 5 muxes, as shown in Fig. 4(c). The conventional RiBM architecture for RS decoders needs to evaluate both the error evaluator polynomial and error locator polynomial, but in the BCH decoding procedure, errors can be detected and corrected using only error locator polynomial. Thus, the proposed ECRiBM architecture computes only error locator polynomial. The conventional RiBM architecture with $3t+1$ PE processors use the former t PE processors to obtain the error evaluator polynomial and the latter PE processors from PE_t to PE_{2t} to obtain the error locator polynomial. Therefore the former t PE processors were eliminated and replaced by PE_t to PE_{2t} processors, because BCH decoder does not need the error evaluator polynomial. After $2t$ clock cycle, the ECRiBM architecture can generate the coefficients of the error locator polynomial simultaneously from PE_1 to PE_{t+1} processor in the BCH decoding procedure.

3.3 Performance Comparison

Table 1 summarizes the hardware complexity and path delay of the various KES architectures, and compares them with the proposed ECRiBM architecture. The ECRiBM architecture reduces the hardware complexity over the previous RiBM architecture. The RiBM architecture requires $3t+1$ *GF* adders, $6t+2$ *GF* multipliers, $6t+2$ latches and

Table 1 Comparison of hardware complexity and path delay for KES architectures.

Architecture	Adders	Multipliers	Latches	D-FFs	Muxes	Clock Cycles	Critical Path Delay
iBM(Berlekamp) [1]	$3t$+1	$5t$+3	-	$6t$+2	$2t$+1	$2t$	$>2\cdot(T_{\mathrm{mul}}+T_{\mathrm{add}})$
RiBM [6]	$3t$+1	$6t$+2	$6t$+2	-	$3t$+1	$2t$	$T_{\mathrm{mul}}+T_{\mathrm{add}}$
PGZ, t=3 [8]	11	18	17	-	-	3	$T_{\mathrm{mul}}+T_{\mathrm{add}}$
ME [9]	$8t$	$8t$	-	$78t$+4	$40t$+2	$10t$	$3\cdot T_{\mathrm{or2}}+T_{\mathrm{xor2}}+T_{\mathrm{mux2}}$
pDcME [10]	$4t$	$8t$	-	$54t$	$20t$	$10t$	$T_{\mathrm{inv}}+T_{\mathrm{and}}+3\cdot T_{\mathrm{mux}}$
Proposed ECRiBM	$2t$+1	$4t$+2	$4t$+2	$2t$+1	$10t$+5	$2t$	$T_{\mathrm{mul}}+T_{\mathrm{add}}+2\cdot T_{\mathrm{mux}}$

Table 2 Implementation results of the KES blocks for BCH(3860, 3824) decoder and BCH(2040, 1930) decoder.

Architecture	Proposed ECRiBM for BCH(3860, 3824)	RiBM for BCH(3860, 3824)	PGZ in [8] for BCH(3860, 3824)	Proposed ECRiBM for BCH(2040, 1930)	RiBM for BCH(2040, 1930)
Technology	90nm CMOS 1.1V	90nm CMOS 1.1V	90nm CMOS 1.1V	90nm CMOS 1.1V	90nm CMOS 1.1V
Total # of Gates	10,800	16,700	12,000	24,700	36,400
Clock Rate (MHz)	420	430	420	400	410
Latency (clocks)	6	6	3	20	20

$3t$+1 muxes. In contrast, the proposed ECRiBM architecture consisting of $2t$+1 PE processors requires $2t$+1 *GF* adders, $4t$+2 *GF* multipliers, $4t$+2 latches, $2t$+1 D flip-flops and $10t$+5 muxes. Due to requiring only $2t$+1 PEs, the proposed ECRiBM architecture requires a lower number of *GF* multiplier and *GF* adders, and thus it has smaller area compared to the RiBM architecture. The proposed architecture has a critical path $T_{mul} + T_{add} + 2 \cdot T_{mux}$, which is comparable to the critical path in the RiBM architecture.

4. Result and Comparison

The proposed ECRiBM KES architecture was designed in Verilog HDL and simulated to verify its functionality. The proposed ECRiBM KES architecturehas been synthesized using appropriate time and area constraints. Both simulation and synthesis steps were carried out using SYNOPSYS design tools and 90-nm CMOS technology optimized for a 1.1 V supply voltage. Table 2 shows the implementation results of the proposed ECRiBM KES architecture, conventional RiBM KES architecture, and PGZ architecture for BCH(3860, 3824) and BCH(2040, 1930) decoders. The ECRiBM KES architecture for BCH(2040,1930) decoder operates approximately at a clock frequency of 400 MHz and requires approximately 32% fewer gate counts and extremely simpler control logic than the conventional RiBM KES architecture. Also for BCH(3860,3824) decoder, implementation result shows that proposed design has an advantage in hardware complexity over both the RiBM and PGZ KES architectures.

Figure 5 shows a timing chart for two-parallel 32-channel concatenated BCH(2040, 1930) decoder. The sharing of the KES block requires time multiplexing between each single channel. The syndrome computation block provides $2t$ syndromes after the processing delay of 255 clock cycles required for computing the syndrome polynomial. Since multiple syndrome computation blocks are connected by one KES block, each syndrome values of a time-multiplexed channel are entered into the KES block sequentially. The ECRiBM algorithm takes only $2t$ clock cycle in solving the key equation. After 255 clock cycles, which is codeword latency, KES block outputs the error locator polynomials $\sigma(x)$ in parallel feeding to the chien search block and the error correction block can correct up multiple errors by toggling error location bit from the chien search block.

The proposed two-parallel concatenated BCH Super-FEC architecture was modeled in Verilog HDL and simulated to verify their functionality using a test pattern generated from C simulator. After complete verification of the design functionality it was then synthesized using appropriate time and area constraints. Both simulation and synthesis steps were carried out using SYNOPSYS design tool and 90-nm CMOS technology optimized for a 1.1 V supply voltage.

Table 3 shows the implementation results of the concatenated BCH decoder architectures. The total number of gates is 1,400,000 from the synthesized results excluding the RAM used in the interleaver/deinterleavers, frame converters and FIFOs. Required memory size for the concatenate decoder is approximately 270 kbytes including all FIFOs, 2 frame converters, 2 interleavers and 3 deinterleavers. The proposed architecture has a critical path $T_{mult} + T_{add} + 2 \cdot T_{mux}$ in KES block. From pre-layout simulation, the proposed concatenated BCH Super-FEC architecture can operate at a clock frequency of 400 MHz and has a data processing rate of 102.4-Gb/s in 90-nm CMOS technology. Although the proposed architecture has 5 times lager hardware complexity compared to the RS(255,239)-based FEC architecture, it has additional 2.4 dB coding gain without additive redundancy compared with the RS(255,239)-based FEC architecture. Also, the proposed architecture has very high data processing rate and can be used for next generation 100 Gb/s optical communications.

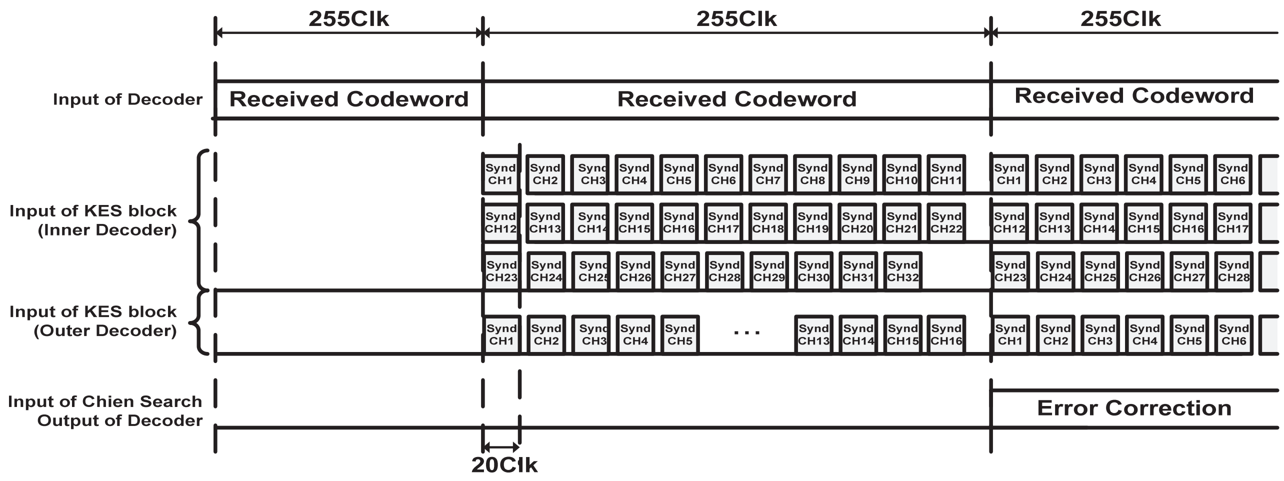

255Clk
255Clk
255Clk
Input of Decoder
Received Codeword
Received Codeword
Received Codeword
Input of KES block (Inner Decoder)
Input of KES block (Outer Decoder)
Input of Chien Search Output of Decoder
Error Correction
20Clk

Fig. 5 Timing chart of 32-ch. two-parallel concatenated BCH decoder.

Table 3 Implementation result of the 32-ch. concatenated BCH decoder architectures.

Design	BCH (3860, 3824)	BCH (2040, 1930)	3-iteration Concatenated BCH(3860, 3824) +BCH(2040, 1930)
Technology	90nm CMOS 1.1V	90nm CMOS 1.1V	90nm CMOS 1.1V
Total No. of Gates	122,000	345,000	1,400,000
Memory Size (bytes)	15,488	16,320	266,784
Clock Rate (MHz)	420	400	400
Latency (clocks)	484	510	5,852 (14.6μs)
Throughput (Gb/s)	107.5	102.4	102.4

5. Conclusion

This paper presents the design and implementation of the high-speed two-parallel concatenated BCH Super-FEC architecture for optical communication systems. The decoding process is used to achieve 100-Gb/s throughput in 90-nm CMOS technology. A low-complexity ECRiBM algorithm block is applied to the concatenated BCH decoder. Two-parallel processing by converting frame format and block interleaving methods allow the decoder to achieve higher correction ability and compensate for serious transmission quality degradation. As a result, two-parallel concatenated BCH Super-FEC architecture has very high data processing rate and additional 2.4 dB coding gain without additive redundancy compared with RS(255,239)-based FEC architecture. Thus, it has potential applications in the next generation FEC schemes for 100 Gb/s optical communications.

References

[1] H.O. Burton, "Inversionless decoding of binary BCH codes," IEEE Trans. Inf. Theory, vol.IT-17, no.4, pp.464–466, July 1971.
[2] "Forward error correction for high bit-rate DWDM submarine system," Telecommunication Standardization Section, International Telecom Union, ITU-T G. 975.1, Feb. 2004.
[3] 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, no.11, pp.1565–1573, Nov. 2002.
[4] K. Seki, K. Mikami, A. Katayama, S. Suzuki, N. Shinohara, and M. Nakabayashi, "Single-chip FEC codec using a concatenated BCH code for 10 Gb/s long-haul optical transmission systems," Proc. 2003 IEEE Custom Integrated Circuits Conference, pp.279–282, Sept. 2003.
[5] S. Lee, C-S. Choi, and H. Lee, "Two-parallel Reed-Solomon based FEC architecture for optical communications," IEICE Electronics Express, vol.5, no.10, pp.374–380, May 2008.
[6] 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.
[7] S. Yoon and H. Lee, "A discrepancy-computationless RiBM algorithm and its architecture for BCH decoders," 21st Annual IEEE International SoC Conference, pp.379–382, Sept. 2008.
[8] H.Y. Hsu, S.F. Wang, and A.Y. Wu, "A novel low-cost multi-mode reed solomon decoder design based on Peterson-Gorenstein-Zierler algorithm," The Journal of VLSI Signal Proc., pp.251–259, Nov. 2004.
[9] H. Lee, "High-speed VLSI architecture for parallel Reed-Solomon decoder for optical communications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.11, no.2, pp.288–294, April 2003.
[10] S. Lee and H. Lee, "A high-speed pipelined degree-computationless modified euclidean algorithm architecture for Reed-Solomon decoders," IEICE Trans. Fundamentals, vol.E91-A, no.3, pp.830–835, March 2008.

Sangho Yoon received M.S. and B.S. degrees, both in Information & Communication engineering from Inha University, Incheon, Korea, in 2007 and 2009 respectively. His research interests are digital VLSI circuits and systems design for communication including design and implementation of decoding algorithm for RS/BCH forward error correction codes.

Hanho Lee received the Ph.D. and M.S. degrees, both in Electrical & Computer Engineering, from the University of Minnesota, Minneapolis, in 2000 and 1996 respectively, and the B.S. degree in Electronics Engineering from Chungbuk National University, S. Korea, in 1993. In 1999, he was a Member of Technical-Staff-1 at Lucent Technologies, Bell Labs, Holmdel, NJ. From April 2000 to August 2002, he was a Member of Technical Staff at the Lucent Technologies (Bell Labs Innovations), Allentown, where he was responsible for the development of VLSI architectures and implementation of high-performance DSP multiprocessor SoC for wireless infrastructure systems. From August 2002 to August 2004, he was an assistant professor at the Department of Electrical & Computer Engineering, University of Connecticut. Since August 2004, he has been with the School of Information and Communication Engineering, Inha University, where he is presently an Associate Professor. His research interests include design of VLSI circuits and systems for communications, System-on-a-Chip (SoC) design, reconfigurable architecture, and forward error correction coding.

Kihoon Lee received the B.S. degree in Information & Communication engineering in 2009, from Inha University, Incheon, Korea, where he is currently working toward the M.S degree. His research interests are digital VLSI circuits and systems design for communications, with emphasis on designing encoding and decoding algorithms for forward error correction codes and their efficient hardware implementation.