Design of the (248,216) Reed-Solomon Decoder with Erasure Correction for Blu-ray Disc

Taegeun Park, *Senior Member*, IEEE

Abstract — *This paper presents an efficient VLSI architecture of the (248, 216) Reed-Solomon decoder with erasure correction capability for Blu-ray Disc (BD). The proposed architecture is designed with four-stage pipelines: the syndrome and erasure location polynomials calculation, the errata location polynomial calculation, the errata evaluation polynomial calculation, and the Chien search and errata value evaluation. Each stage is carefully balanced to maximize the throughput of the pipeline for BD applications. To solve the key equation, the Berlekamp-Massey algorithm is transformed into a symbol-serial structure and the fully utilized VLSI architecture, maintaining the maximum decoding performance, is proposed. Therefore, the proposed architecture maximizes the throughputs requiring less hardware resources. The gate counts for the proposed RS decoder is 74K using the Hynix 0.35um standard cell library and the maximum throughput is 700Mbps@100MHz, which is fast enough for 16x BD applications*[1].

Index Terms — Blu-ray disc, error correcting codes, Reed-Solomon code, VLSI architecture.

I. INTRODUCTION

As HDTV becomes more widespread, the consumer demand for recording HDTV programming will rise [1]. Blu-ray Disc (BD) is the next-generation optical disc format jointly developed by leading consumer electronics and pc companies. BD can record up to 27GB of video data which takes over 2 hours of digital high definition video and more than 13 hours of standard TV broadcasting. Blu-ray employs a 36Mbps data transfer rate, which is more than enough to record digital high definition broadcasts or high definition images from a digital video camera while maintaining the original picture quality. In addition, by fully utilizing an optical disc's random accessing functions, it is possible to easily edit video data captured on a video camera or play back a pre-recorded video on the disc while simultaneously recording images being broadcast on TV.

Among the various kinds of error correcting codes (ECC) in digital communication systems, the Reed-Solomon (RS) code is widely used block codes, especially in storage systems, due to its excellent error-correcting capabilities. Also, the need of erasure correction is increasing in order to reduce the burst error by utilizing the information of a known error location. It is well known that an (n, k) RS code has k message symbols and n coded symbols, where each symbol belongs to $GF(2^m)$. An (n, k) RS code can correct ν symbols and ρ erasures with $2\nu + \rho \le n - k$. From an RS code, an error can be corrected by finding out the error location and the error value, whereas an erasure is defined as an error with a known error location, and hence its correction is reduced to finding the error value.

Fig. 1 shows the block diagram of the error correcting code and modulation/demodulation blocks for BD. The user data is partitioned into blocks and each block contains 32 data frames of 2K bytes of user data. The partitioned data (clusters) of 64K bytes are protected by two RS blocks working together: LDC (Long Distance Code) and BIS (Burst Indicator Subcode). Since LDC deals with the user data, whereas BIS contains small amounts of address control data, the RS decoder for LDC requires much more performance than that for BIS in terms of bit rate. The LDC Endec (encoder/decoder) employs the interleaving/ deinterleaving to cope with random and burst errors. The (248, 216) RS decoder with erasure correcting capability is required to decode the 17PP-demodulated user data read from the storage. Therefore, high performance VLSI architecture with area-efficiency for the (248, 216) RS decoder is essential to meet the specifications of Blu-ray Disc.

Many researchers have proposed a number of VLSI architectures on the RS decoder for various applications, such as communication [5][6], consumer electronics [13], and storage [4], etc. They have focused on minimizing the complexity of decoders while maintaining the decoding performance. There have been two main approaches in solving the key equation which is the most difficult part to compute in RS decoding: the Berlekamp-Massey (BM) algorithm [4][7][9] and the extended Euclid's algorithm [5][6] [11][12][13]. The inverse-free BM algorithm and its VLSI architecture have been proposed for finding the error locator polynomial in the RS decoder [7], and it has been generalized to find both errors and erasures [8]. A low-complexity RS decoder chip for DVD applications has been proposed in [4]. They propose a decomposed, inverse-free BM algorithm to solve the key equation, which reduces the hardware complexity significantly. However, this architecture degrades the decoding performance in case that the RS decoder has a larger (n - k) value, such as (248, 216); the decoding with erasure correction makes it worse since the degrees of the errata location and evaluation polynomials become higher. The architecture in [4] adopts three-stage pipelining to

[1] This work was supported by grant No.R01-2005-000-11054-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

Taegeun Park is with the Information, Communication, and Electronics Engineering Department, The Catholic University of Korea, Bucheon 420-743, South Korea (e-mail: parktg@catholic.ac.kr).

Contributed Paper
Manuscript received May 24, 2005

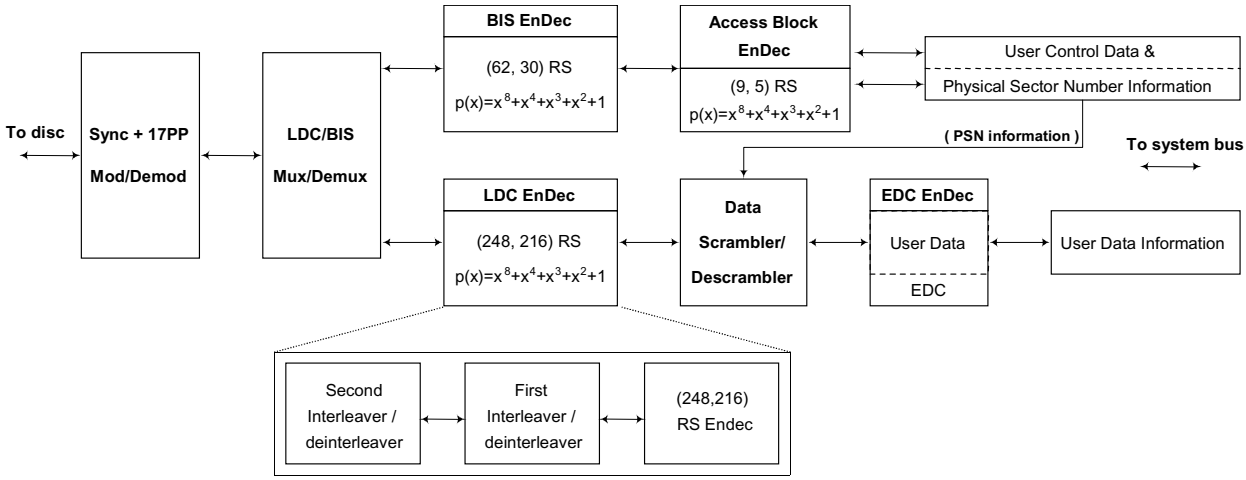


Fig. 1. ECC and Mod/Demod blocks in Blu-ray Disc.

produce the decoded outputs: syndrome calculation, key equation solver, and Chien search. A VLSI architecture that is scalable with respect to the throughput rate has been proposed [11], and the scalability can be achieved by applying a systematic time-sharing technique. Sarwate has proposed high-speed systolic architecture that eliminates the bottleneck of the BM algorithm by a series of algorithmic transformations [9]. A pipelined architecture that has the regular cell structure for the extended Euclidean algorithm has been proposed to reduce the critical path [12].

In this paper, we propose an area-efficient and high-performance VLSI architecture for the (248, 216) RS decoder with erasure correcting capability for Blu-ray Disc. The proposed RS architecture consists of carefully balanced 4-stage pipelines to maximize the data rate. The architecture does not require the polynomial expansion block to generate the erasure location or Forney syndrome polynomials. Instead, it utilizes the existing syndrome generation block without any performance degradation. Also, the inverse-free Berlekamp-Massey algorithm is designed in an efficient manner by adopting a pipeline-balanced, symbol-serial architecture. Consequently, the synthesis result shows that the proposed RS decoder requires small amounts of hardware resources and shows high enough performance to be used in BD applications. This paper is outlined as follows: Section 2 introduces Reed-Solomon codes. Section 3 describes the proposed RS decoder architecture. The performance analysis and design results for the architecture are also discussed in Section 3. Finally, concluding remarks are given in Section 4.

II. REED-SOLOMON CODES

An (n, k) RS code defined in the Galois field $GF(2^m)$ has code words of length $n = 2^m - 1$, where m is a positive integer and k is the number of information symbols in the codeword. The (n, k) RS code has a minimum distance of d, where $k = n - (d - 1)$ denotes the number of m-bit message symbols, and $d - 1$ denotes the number of parity symbols. Moreover, given ρ erased coordinates, the code words will have an effective minimum distance of ($d_{min} - \rho$) over the unerased coordinates [2]. It follows that we can correct

$$t_e = \frac{\lfloor d_{\min} - \rho - 1 \rfloor}{2}$$

errors in the unerased coordinates of the received word. In other words, we can correct ν errors and ρ erasures so long as $(2\nu + \rho) < d_{min}$.

Let $C(x)$ denote the transmitted codeword polynomial and let $R(x)$ denote the received word polynomial. The input to the decoder is $R(x)$, and it assumes that

$$R(x) = C(x) + E(x) + f(x), \tag{1}$$

where $E(x)$ denotes the error polynomial and $f(x)$ denotes the erasure polynomial.

An erasure location polynomial can be obtained using the erasure location indicators.

$$\Gamma(x) = \prod_{l=1}^{\rho}(1 - Y_l x) = \sum_{l=1}^{\rho}\Gamma_l x^l \ , \tag{2}$$

where $\Gamma_0 = 1$ and the Γ_j's are known functions of Y_1, Y_2, …, $Y\rho$ for $1 \le l \le \rho$ and $\Gamma(x)$ is the polynomial with zeros at the inverse erasure locations. Since the syndrome is only a function of the error/erasure polynomial, the syndrome can be defined in the following form.

$$S_l = r(\alpha^l) = \sum_{k=1}^{\nu} e_{i_k} X_k^l + \sum_{k=1}^{\rho} f_{i_k} Y_k^l \text{ , for } 1 \le k \le d-1 \tag{3}$$

where X_k is the kth error location, e_{i_k} is the i_k th error amplitude, Y_k is the kth erasure location, and f_{i_k} is the i_k th erasure amplitude. Define the syndrome polynomial as

$$S(x) = \sum_{l=1}^{2t} s_l x^l \tag{4}$$

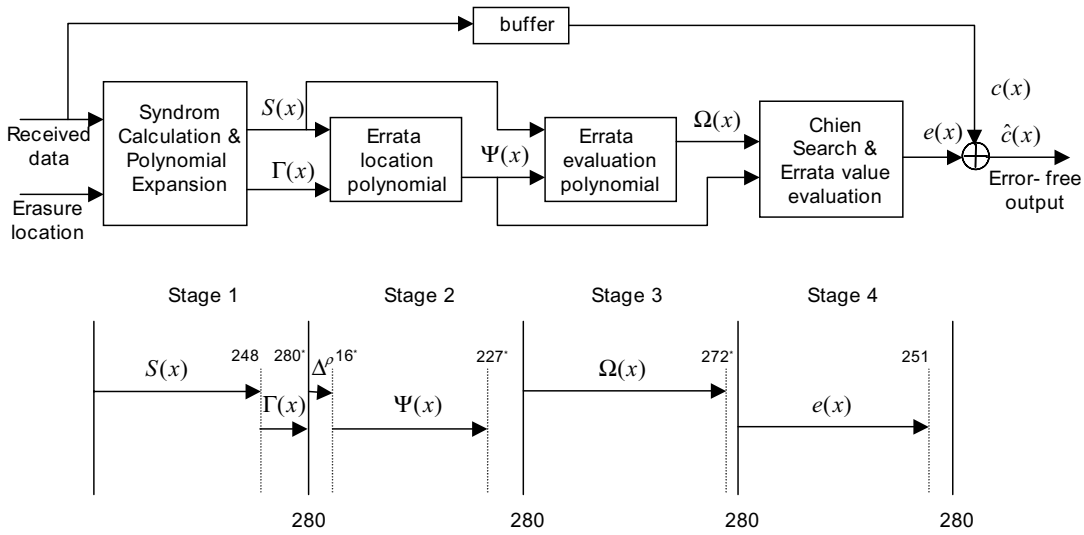


Fig. 2. The proposed four-stage pipeline architecture for (248, 216) the RS decoder and its timing.

with which the key equation for errors and erasure decoding is obtained.

$$\Lambda(x)\Gamma(x)[1+S(x)] \equiv \Omega(x) \bmod x^{2t+1} \tag{5}$$

Equation (5) can be simplified by combining all information into the Forney syndrome polynomial $\Xi(x)$

$$1+\Xi(x) \equiv \Gamma(x)[1+S(x)] \bmod x^{2t+1}, \tag{6}$$

where the Forney syndrome polynomial $\Xi(x) \equiv \Gamma(x)[1+S(x)]-1$.

The key equation now takes the form

$$\Lambda(x)[1+\Xi(x)] \equiv \Omega(x) \bmod x^{2t+1} \tag{7}$$

which can be solved by the BM algorithm or the extended Euclid's algorithm.

Once the error locator polynomial is known, combine it with the erasure locator polynomial to obtain a single error/erasure locator polynomial Ψ(x).

$$\Psi(x) = \Lambda(x)\Gamma(x) \tag{8}$$

A modified Forney algorithm can then be used to compute the error and erasure values.

$$e_{i_k} = \frac{-X_k \Omega(X_k^{-1})}{\Psi'(X_k^{-1})}, \quad \rho_{i_k} = \frac{-Y_k \Omega(Y_k^{-1})}{\Psi'(Y_k^{-1})} \tag{9}$$

where the error locators $X_1 = \alpha^{i_1}$, $X_2 = \alpha^{i_2}$, ⋯, $X_v = \alpha^{i_v}$ and the erasure locators $Y_1 = \beta^{j_1}$, $Y_2 = \beta^{j_2}$, ⋯, $Y_\rho = \beta^{j_\rho}$.

III. THE PROPOSED ARCHITECTURE

As shown in Fig. 2, we divide the decoding process into four steps: (1) calculating the syndrome and erasure location polynomials from the received codewords and erasure location indicators, (2) generating the errata location polynomial, (3) computing the errata evaluation polynomial, and (4) finally finding the errata locations by Chien search and evaluating the errata values by Forney's algorithm. In the figure, the number with an asterisk stands for the maximum number of clock cycles to complete the corresponding stage.

The proposed architecture consists of four balanced pipeline stages in terms of the number of clocks at each stage, thus the throughputs of the proposed RS decoder become maximized. In the first stage, the syndrome polynomial S(x) is computed with the received codewords. And then the computation for the erasure location polynomial Γ(x) starts right after the syndrome calculation has been completed. We schedule these in a series to share the hardware without any performance degradation because the hardware structures that calculate the syndromes and the erasure location polynomial look very similar. It takes at most 280 clock cycles to complete the first step. The second and third stages will solve the key equation to generate the errata location polynomial Ψ(x) and the errata evaluation polynomial Ω(x) based on the inverse-free Berlekamp-Massey algorithm [8]. We revise the BM algorithm to balance the pipelines by transforming the architecture into a two-symbol-serial structure, thus maximizing the throughputs. Ψ(x) can be computed from S(x) and Γ(x) generated in the previous stage by 227 clock cycles at worst. Also, it takes a maximum of 272 clock cycles to compute Ω(x) from S(x) and Ψ(x). Detailed discussions on architectures and performance can be found in the following subchapters. Finally, the errata locations and errata values can be found by the Chien search and Forney evaluation procedures in stage four.

A. Syndrome and errata location polynomials (S(x), Γ(x))

The structures for the syndrome calculation and the

polynomial expansion look very similar, thus we share the hardware to reduce the resources. Fig.3 shows the block diagram for generating the syndrome and erasure location polynomials. The syndrome polynomial is obtained from $S(x)=\sum_{l=1}^{2t} s_l x^l$, $S_l = R(\alpha^l)$, where α is primitive in $GF(256)$ and $R(x) = R_0 + R_1x + \cdots + R_{n-1}x^{n-1}$ is a received polynomial. If no error occurs, the syndromes are all zero. The coefficients of the syndromes can be efficiently computed by Horner's rule in a nested form $S_l = (\cdots(R_{n-1}\cdot\alpha^l + R_{n-2})\cdot\alpha^l + \cdots + R_1)\cdot\alpha^l + R_0$.

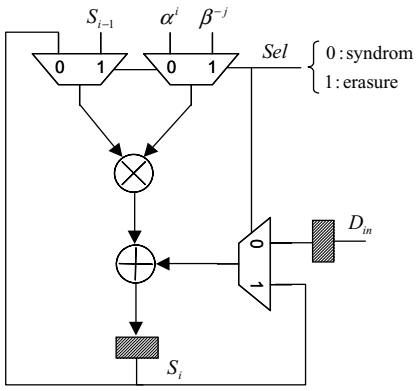


Fig. 3. Block for syndrome and erasure location polynomials.

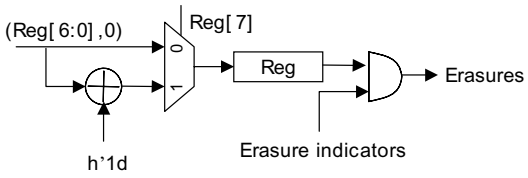


Fig. 4. Block for computing erasure location.

At first the syndrome is calculated from the received codewords and then the erasure polynomial can be computed based on the same hardware. Therefore, the bit-wide erasure location indicators are transformed to the roots for the erasure location polynomial $\Gamma(x)$ and buffered until the point at which they are used. Fig. 4 shows the block diagram for computing erasure location without using a finite-field multiplier (FFM). According to the most significant bit of Reg, the primitive element for the current erasure location will be either (Reg[6:0],0) which is a left-shifted value of Reg or (Reg[6:0],0) $\oplus$ h'1d, where h'1d denotes a primitive polynomial value in hexadecimal. The erasure location polynomial is computed from $\Gamma(x)=\prod_{\beta^{-j}\in\Gamma}(x-\beta^{-j})$, where Γ is the set of β^js which are the roots of the erasure location polynomial. $\Gamma(x)$ can also be expressed in a nested form and obtained by multiplying a linear shift of $\Gamma(x)$ with β^j and adding the result to itself recursively. In case that $v = 0$ and $\rho = 32$, this step takes 280 (248 + 32) clock cycles at worst.

B. Errata location and evaluation polynomials (Ψ(x),Ω(x))

There are two main methods used to solve the key equation: the Berlekamp-Massey (BM) algorithm and the Euclidean algorithm. The BM algorithm is generally known to require a smaller hardware complexity, whereas the Euclidean algorithm has efficiency in terms of regularity [4]. Many researchers have proposed the architectures for the BM algorithm [4][7][9][10]. Liu's architecture needs an FFI (Finite-Field Inverter) for a division operation, which requires larger area and processing time [10]. The inverse-free BM algorithms and architectures [7][8][9] have been proposed but their architectures run in parallel that requires 3v FFMs. If the erasure correction capability is added, the required resources will increase. Performance-wise, they solve the key equation too fast with too much hardware; nevertheless, it does not increase the decoding performance at all since the performance is limited by the other pipeline stages. The decomposed BM architecture [4] has been proposed to slow down the whole process in a sequential manner with the least number of FFMs. However the performance of this architecture becomes worse when the RS decoder has a larger (n - k) value, such as (248, 216), since the stage for computing the BM algorithm becomes a bottleneck for decoding. Furthermore, when the erasure correction capability is added, the throughput becomes even worse since the degree of the errata location and evaluation polynomials become higher. For example, when $v = 8$ and $\rho = 16$ in the (248, 216) RS decoder, the process for solving the key equation increases as many as 681 clock cycles, whereas the syndrome calculation still requires only 248 clock cycles. Thus, the overall performance is limited by the BM stage.

In this paper, we revise the inverse-free BM algorithm to balance the pipeline and maximize the decoding performance. The inverse-free BM algorithm is transformed into a two-symbol-serial structure as shown below. The initial discrepancy Δ^ρ needs to be computed from S(x) and $\Gamma(x)$ before solving the key equation. At each iteration in an inner loop, two coefficients Ψ^i_{2j} and Ψ^i_{2j+1} of $\Psi(x)$, and the intermediate discrepancy Δ^{i+1}_j are computed at the same time. From the iterative algorithm, it is clear that the degree of $\Psi^i(x)$ is increased at most by one during each iteration. Fig.5 shows the block diagram for calculating $\Psi(x)$. It requires 6 FFMs and some random logics. We use the FFM structure proposed in [13] which is modular and regular.

/* The algorithm for $\Psi(x)$ */
$\gamma^{\rho-1} = 0, \delta = 1,$
$\Psi^{\rho-1}(x) = \mathrm{T}^{\rho-1} = \Gamma(x),$
$\Delta^\rho = S_{\rho+1}\Gamma_0 + S_\rho\Gamma_1 + \cdots + S_1\Gamma_\rho$
for $i = \rho$ **to** n - k - 1
  $\Delta^{i+1}_{-1} = 0$
  for $j = 0$ **to** $\lfloor(v_i + \rho)/2\rfloor$
    if $j = 0$ **then** $\Psi^i_{2j} = \delta\cdot\Psi^{i-1}_{2j}$

else $\Psi^i_{2j} = \delta \cdot \Psi^{i-1}_{2j} + \Delta^i \cdot x\mathrm{T}^{i-1}_{2j-1}$

$\Psi^i_{2j+1} = \delta \cdot \Psi^{i-1}_{2j+1} + \Delta^i \cdot x\mathrm{T}^{i-1}_{2j}$

$\Delta^{i+1}_j = \Delta^{i+1}_{j-1} + S_{i-2j+2}\Psi^i_{2j} + S_{i-2j+1}\Psi^i_{2j+1}$

end loop

if $\Delta^i = 0$ **or** $2 \cdot \gamma^{i-1} > i - \rho$

then $\gamma^i = \gamma^{i-1}$, $\mathrm{T}^i(x) = x\mathrm{T}^{i-1}(x)$

else $\gamma^i = i - \rho + 1 - \gamma^{i-1}$, $\mathrm{T}^i(x) = \Psi^{i-1}(x)$, $\delta = \Delta^i$

end loop

$$\Psi(x) = \Psi_0 + \Psi_1 x + \cdots + \Psi_{\nu+\rho} x^{\nu+\rho}$$

where ρ is the number of erasures, Δ^i is the ith discrepancy, and δ is the previous nonzero discrepancy; $\mathrm{T}^i(x)$ is the correction polynomial, $\Psi^i(x)$ is the ith errata location polynomial with degree $\nu_i + \rho$, and $\Gamma(x)$ is the erasure locator polynomial; γ is the length of the shift register (degree variable). To compute $\Omega(x)$, we allocate the separate pipeline stage to balance the pipelining. The algorithm below is for computing $\Omega(x)$ from $\mathrm{S}(x)$ and $\Psi(x)$. Fig. 5 and Fig.6 show the block diagrams for computing $\Psi(x)$ and $\Omega(x)$. From the architecture point of view, the critical path delay for computing $\Psi(x)$ and $\Omega(x)$ is limited to the time of FFM + XOR + MUX. The result of logic synthesis shows that the FFM used in this design costs 380 gates and takes about 2.4ns using 0.35um technology.

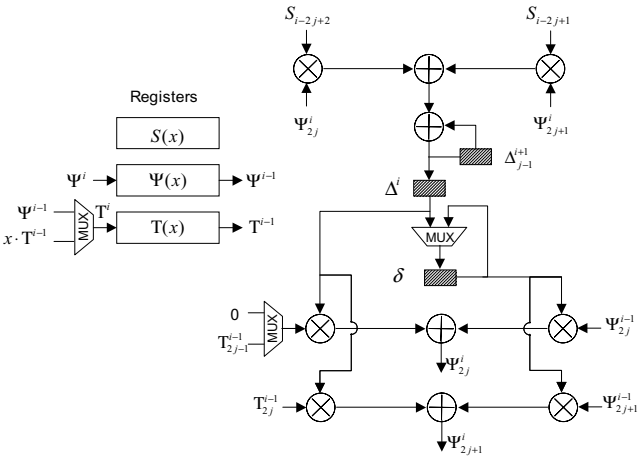


Fig. 5. Block for the errata location polynomial Ψ(x).

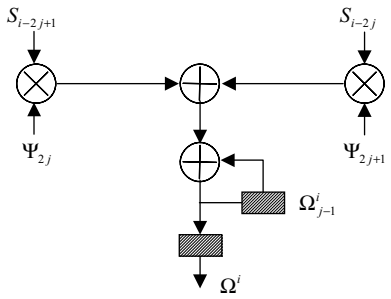


Fig. 6. Block for the errata evaluation polynomial Ω(x).

/* The algorithm for $\Omega(x)$ */

for $i = 0$ **to** $\nu + \rho - 1$

$\Omega^i_{-1} = 0$

for $j = 0$ **to** $\lfloor i/2 \rfloor$

$\Omega^i_j = \Omega^i_{j-1} + S_{i-2j+1}\Psi_{2j} + S_{i-2j}\Psi_{2j+1}$

end loop

end loop

$$\Omega(x) = \Omega_0 + \Omega_1 x + \cdots + \Omega_{\nu+\rho-1} x^{\nu+\rho-1}$$

where $\Omega^i(x)$ is the ith errata evaluation polynomial with degree $\nu_i + \rho$ - 1. In case that the indexes for the coefficients, S and Ψ, fall outside the range for different i and j, the values for the corresponding coefficients are set to zero.

Now we analyze the total number of clock cycles required to compute $\Psi(x)$ and $\Omega(x)$ on the proposed architecture. In case that both errors and erasures are corrected, $2\nu + \rho \le n - k$ and $\lceil (\rho + 1)/2 \rceil$ cycles are needed to compute the initial discrepancy value Δ^ρ. The number of cycles to compute $\Psi(x)$ is:

$$\lceil (\rho+1)/2 \rceil + \sum_{i=\rho}^{\rho+\nu-1} (\lfloor i/2 \rfloor + 1) + \sum_{i=\rho+\nu}^{n-k-1} (\lfloor (\nu+\rho)/2 \rfloor + 1) \qquad (10)$$

The degree of the ith error location polynomial ν_i is $\nu_i \le \rho + i$, for $0 \le i < \nu$ and $\nu_i \le \rho + \nu$, for $\nu \le i < 2\nu$, where ν denotes the number of correctable errors. The number of cycles required to compute $\Omega(x)$ is:

$$\sum_{i=0}^{\nu+\rho-1} (\lfloor i/2 \rfloor + 1) \qquad (11)$$

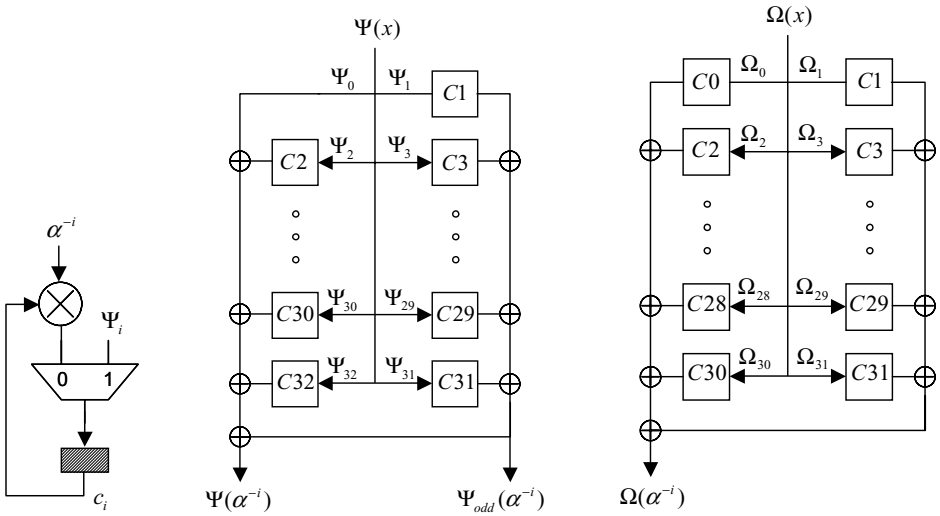
Table 1 shows the number of cycles required to compute $\Psi(x)$ and $\Omega(x)$ for different ν and ρ in the (248, 216) RS decoder. The maximum number of cycles for $\Psi(x)$ and $\Omega(x)$ are 227 and 272, respectively.

TABLE I
THE NUMBER OF CYCLES TO COMPUTE Ψ(X) AND Ω(X) WITH DIFFERENT (ν, ρ)'S IN THE (248, 216) RS DECODER

ν	ρ	Cycles ($\Psi(x)$)	Cycles ($\Omega(x)$)
0	32	0	272
4	24	127	210
8	16	197	156
12	8	227	110
16	0	217	72

C. Chien search and errata value evaluation

Once the key equation is solved, the roots of $\Psi(x)$ can be located through the well-known Chien search, which evaluates $\Psi(x)$ at all field elements in $GF(256)$ and determines whether the current position has an error or not. In other words, if $\Psi(\bar{\alpha}^i) = 0$, there is an error at the ith symbol R_i in the received polynomial $R(x)$. Fig.7 shows the Chien search cell and Chien search block. Each cell in Fig.7(b) computes the partial result for the corresponding field element.



(a) Chien search cell **(b) Chien search block** **(c) Errata value evaluation block**

Fig. 7. Chien search and errata value evaluation.

To evaluate the errata values, the Forney algorithm is preferred due to its lower circuit complexity. Using the Forney algorithm, the errata values can be computed as stated in equation (9). The first derivative of $\Psi(x)$ can be represented as:

$$\Psi'(x) = (\sum_i \Psi_i x^i)' = \frac{1}{x}\Psi_{odd}(x) \tag{12}$$

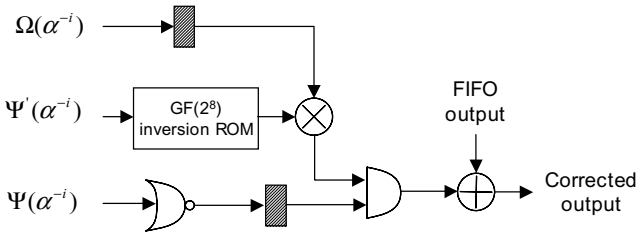


Fig. 8. Forney's algorithm and errata correction block.

Fig.8 shows the errata correction block using the Forney algorithm. The evaluation of $\Omega(\alpha^i)$ is computed in parallel with the Chien search, and therefore an additional few cycles are needed to produce the correct term e_i. The architecture to compute the Chien search and errata value evaluation seems quite straightforward and regular, but requires considerable gate counts mostly caused by FFMs. As shown in Fig. 7(b) and (c), the number of FFMs is 64. We can optimize the gate counts for each FFM using the fact that one of the inputs of FFM is a constant. The syndrome calculation block explained earlier has a similar structure, however it is difficult to optimize further because the FFMs in this block are shared with the polynomial expansion block for $\Gamma(x)$ that needs both inputs of FFM as variables. According to the Forney algorithm, a finite-field inversion is required to compute the errata values. We can implement this with either a LUT (Look-Up Table) ROM or an algorithm-based FFI circuit. When we apply the Fermat's algorithm for inversion, the inversion logic with one FFM takes m-1 cycles in $GF(2^m)$. This is hardly acceptable due to the latency constraint. Therefore, choosing the ROM table seems to be more practical unless the access time of the ROM limits the decoding performance. Finally, the correcting term e_i is added to the delayed input data to produce the error-free codewords.

D. Design and implementations

The proposed (248, 216) four-stage pipelined RS decoder is designed with VerilogHDL in a ModelSim environment and synthesized by Synopsis using a Hynix 0.35um standard cell library. Table 2 shows the synthesized gate counts for the functional blocks in the proposed (248, 216) RS decoder. If we analyze the architecture of the four stages in the proposed RS decoder, the critical path delay is limited to the time of one FFM + α. However, the static timing analysis tells us that the random logic (including one FFM) for controlling the BM algorithm causes the critical path to limit the decoding performance.

The maximum throughput of the proposed decoder is about 700Mbps@100MHz, which is fast enough for 16x BD applications. The maximum latency of the proposed architecture is 280 x 4 = 1120 clock cycles. In table 2, we note that the gate counts for stage 4, which requires 64 FFMs, are much lower than the gate counts for stage 1, which contains only 33 FFMs. This is because the multiplicand of the FFM in stage 4 is a constant, so we can further optimize the logics using Boolean algebra as explained in subchapter 3.3. If we slow the operating clock frequency down to 50MHz (still fast enough for 8x BD applications) to allow for further optimization in terms of the logic gates, the synthesized gate counts can be reduced to 65K. Since the state-of-art VLSI technology is 0.13um or less these days, we believe that this architecture has more room to have better performance.

TABLE II
GATE COUNTS OF THE FUNCTIONAL BLOCKS IN THE PROPOSED (248, 216) RS DECODER

Pipeline stage	Functional blocks	Gate counts (@100MHz)	Gate counts (@50MHz)
1	Syndrome calculation & polynomial expansion	22K	22K
2	Errata location polynomial	26K	19K
3	Errata evaluation polynomial	13K	11K
4	Chien search & errata value evaluation	13K	13K
	Total	74K	65K

IV. CONCLUSIONS

An efficient (248, 216) Reed-Solomon decoder, based on the Berlekamp-Massey (BM) algorithm for Blu-ray Disc, is proposed. The proposed RS decoder has a four-stage pipeline, which is carefully balanced to maximize the decoding throughput. The BM algorithm used to solve the key equation is transformed into a symbol-serial structure, which processes two symbols each clock cycle, and its architecture requires less hardware while maintaining the decoding performance. The architecture operates the RS decoding, limited to the time of only one FFM + α. After synthesis, using the Hynix 0.35um standard cell library, the number of gate counts is 74K and the maximum throughput is 700Mbps@100MHz, which is fast enough for 16x BD applications.

ACKNOWLEDGMENT

Authors are grateful to IC Design Education Center for providing us with a design environment.

REFERENCES

[1] Blu-ray Disc official homepage, *http://www.blu-ray.com* .
[2] S. Wicker, *Error control systems for digital communication and storages*, Prentice-Hall, 1995.
[3] E. Berlekamp, *Algebraic coding theory*, McGraw-Hill, 1968.
[4] H. Chang, C. Shung, and C. Lee, "A Reed-Solomon Product-Code (RS-PC) decoder chip for DVD applications," *IEEE J. of Solid-State Circuits*, vol.36, no.2, pp.229-238, 2001.
[5] H. Hsu and A. Wu, "VLSI design of a reconfigurable multi-mode Reed-Solomon codec for high-speed communication systems," *Proc. of ASIA-Pacific Conf. on ASIC*, pp.359-362, 2002.
[6] J. Huang, A. Wu, M. Shieh, and C. Wu, "An area-efficient versatile Reed-Solomon decoder for ADSL," *Proc. of IEEE Int'l Sym. on Circuits and Systems*, vol.1, pp.517-520, 1999.
[7] I. Reed, M. Shieh, and T. Truong, "VLSI design of inverse-free Berlekamp-Massey algorithm," *IEE Proc.-E*, vol.138, no.5, pp.295-298, 1991.
[8] T. Truong, J. Jeng, and K. Hung, "Inversionless decoding of both errors and erasures of Rees-Solomon code," *IEEE Trans. on Communications*, vol.46, no.8, pp.973-976, 1998.
[9] D. Sarwate and N. Shanbhag, "High-speed architectures for Reed-Solomon decoders," *IEEE Trans. on VLSI*, vol.9, no.5, pp.641-655, 2001.
[10] K. Liu, "Architecture for VLSI design of Reed-Solomon decoders," *IEEE Trans. Computers*, vol.C-33, pp.178-189, 1984.
[11] W. Wolhelm, "A new scalable VLSI architecture for Reed-Solomon decoders," *IEEE J. Solid-State Circuits*, vol.34, no.3, pp.388-396, 1999.
[12] H. Lee, "Modified Euclidean algorithm block for high speed Reed-Solomon decoder," *IEE Electronics Letters*, vol.37, no.14, pp.903-904, 2001.
[13] S. Kwon and H. Shin, "An area-efficient VLSI architecture of a Reed-Solomon decoder/encoder for digital VCRs," *IEEE Trans. on Consumer Electronics*, vol.43, no.4, pp.1019-1027, 1997.
[14] E. Mastrovito, "VLSI design for multiplication over finite fields $GF(2^m)$," *Proc. 6th Int'l Conf. Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes (AAECC-6)*, pp.297-309, 1988.

Taegeun Park (M'96-SM'03) received the B.S. degree in electronic engineering from Yonsei University, Seoul, Korea in 1985, and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, in 1988 and 1993 respectively.

He has worked as a VLSI design engineer for Coherent Research Inc., Syracuse NY, where he participated in the design and development of the general-purpose associative architecture, focusing on circuit and physical design. He has been a staff design engineer at System IC R&D, Hyundai Electronics Industries Co., Ltd., working on micro controller design and application development. He is currently an associate professor of information, communication, and electronic engineering at The Catholic University of Korea, Bucheon, Korea. His research interests include VLSI design, CAD, DSP, and computer architecture.