Design of Reed-Solomon Decoder for High Speed Data Networks

Young-Shig Choi, Heyk-Hwan Choi

Div. of Electronic, Computer and Telecommunication Eng.

요 약

본 논문에서는 Modified Euclid 알고리즘을 이용하여 고속의 Reed-Solomon 복호기을 설계하였다. Reed-Solomon 부호의 복호 알고리즘은 오증을 계산하고, 에러 위치 다항식을 구한 후, 에러를 판단하여, 에러 크기 값을 구하는 4단계로 이루어지는데, 본 논문에서는 복호기의 속도를 증가시키고 Latency를 줄이기 위하여 병렬구조의 신드롬 생성기와 빠른 클록 속도의 Modified Euclid 알고리즘 블록을 사용하였으며, Chien Search 블록에서는 에러 위치 다항식을 짝수항과 홀수항으로 나누어 설계하였다. 먼저, 알고리즘과 회로의 동작을 확인하기 위해 C++로 프로그램을 작성하여 검증을 한 후, 이를 바탕으로 Verilog로 하드웨어를 기술하였다. 또한, 각 블록에 대한 로직 시뮬레이션을 거친 후, 0.25μm CMOS 라이브러리를 이용하여 Synopsys사의 합성 툴로 합성을 하고, 최종적으로 후반부 설계인 레이아웃을 시행하였다. 본 논문의 칩은 최대 동작 주파수가 250MHz로서 최대 데이터 전송률은 1Gbps이다.

ABSTRACT

In this work, a high speed 8-error correcting Reed-Solomon decoder is designed using the modified Euclid algorithm. Decoding algorithm of Reed-Solomon codes consists of four steps, those are, compute syndromes, find error-location polynomials, decide error-locations, and determine error values. The decoding speed is increased and the latency is reduced by using the parallel architecture in the syndrome generator and a faster clock speed in the modified Euclid algorithm block. In addition, the error locator polynomial in Chien search block is separated into even and odd terms to increase the overall speed of the decoder. All the functionalities of the decoder are verified first through C++ programs. Verilog is used for hardware description, and then the decoder is synthesized with a 0.25μm CMOS TML library. The functionalities of the chip is also verified through test vectors. The clock speed of the chip is 250MHz, and the maximum data rate is 1Gbps.

Key words

Reed-Solomon Decoder, Modified Euclid Algorithm, Syndrome, Chien Search, Forney Algorithm

Ⅰ. Introduction

As the volume of information data and the need for real-time transmission increase very rapidly, the importance of error correction at high speed increases as well. Reed-Solomon codes are known very powerful in correcting random and concatenated errors, and have been widely used in data storage systems and in data transmission systems such as hard disk and wireless communications. In a digital data transmission system, errors occur mainly due to channel impairments. However, the impairments can be alleviated with a combination of error encoder and error decoder as shown in Figure 1.

접수일자 : 2003. 3. 12

In this work, we design an 8-error correction (204,188) Reed-Solomon decoder which is compliant to European satellite broadcasting standards. In implementing the decoder in VLSI, emphasis is given to its operation speed rather than its chip size.

In designing the decoder, we use the Modified Euclid Algorithm(MEA) which becomes popular nowadays in designing a Reed-Solomon decoder[1]~[7]. We can implement the algorithm relatively easily in VLSI, and can avoid the inverse element calculation required in the Euclid algorithm[8]. In [4], 600 Mbps data throughput has been achieved at 75MHz clock speed. In [8], a maximum 87MHz clock speed has been obtained with a 0,35μm process. The designed Reed-Solomon decoder in this paper has 250MHz maximum clock speed, 1Gbps maximum data throughput, and 342 clocks latency.

This paper is organized as follows. In Section II, we describe the detail procedure of implementing (204,188) Reed-Solomon decoder in VLSI followed by the decoder's structure. In section III, the results of simulation, synthesis, and layout are discussed, and finally conclusions follow in section IV.

II. Algorithm and architecture of Reed-Solomon Decoder

A digital data transmission system with error correction function is shown in Fig. 1. Reed-Solomon codes are known one of the most powerful block codes to correct errors, and have been used in various applications. In this paper, we design a (204,188) Reed-Solomon decoder which has a capability of correcting errors up to 8 bytes, and we design it for high speed data transmission applications. The decoder has generation and primitive polynomial expressed in Eq. (1) and (2), respectively.

Generation Polynomial

$$: g(x) = x^{15} + x^{14} + 1 \quad (1)$$

Primitive Polynomial

$$: p(x) = x^{8} + x^{4} + x^{3} + x^{2} + 1 \quad (2)$$

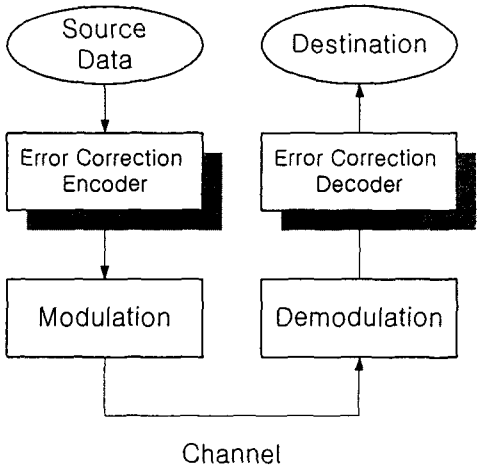

Fig. 1. Digital data transmission system.

The overall structure of Reed-Solomon Decoder is shown in Fig. 2. First, errors are detected through Syndrome generator, modified Euclid algorithm block, and Chien search block. Then, the exclusive OR operation will be performed on the detected errors and the delayed input data to correct errors in the Forney Algorithm & Error Correction block[9]~[10].

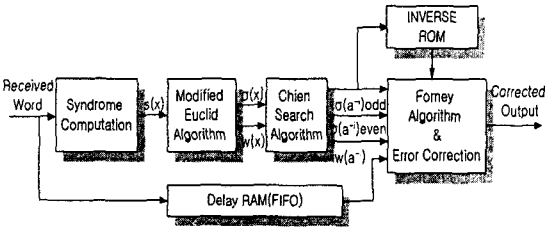

Fig. 2. Architecture of Reed-Solomon Decoder based on MEA.

1. Syndrome Calculation Block

To decode Reed-Solomon codes, syndromes of the received code word should be obtained first. Syndromes are an essential information to decode block codes.

Every code word can be divided by generation polynomial g(x) without remainder. If there is no

errors in the received data, all the syndromes are '0'. However, if there are errors in the received code word, a syndrome polynomial, which is one degree lower than the generation polynomial, is generated. Syndromes are calculated using Eq. (3) and (4) which can be implemented easily into hardware.

$$S_k = r(a^k) = \sum_{i=0}^{N-1} r_i (a^k)^i \qquad 0 \le k \le 2t-1$$
$$= r_0 + r_1 a^k + r_2 a^{2k} + \cdots + r_{n-1} a^{(n-1)k}$$
$$= r_0 + a^k(r_1 + a^k(r_2 + a^k(r_3 + \cdots + a^k(r_{n-2} + r_{n-1} a^k)\cdots))) \quad (3)$$

$$S(x) = \sum_{k=0}^{2t-1} S_k x^k \quad (4)$$

where r is received data and a^k is the root of generation polynomial.

Eq. (5) and (6) show expressions to calculate the syndromes of a (204,188) Reed-Solomon decoder. Syndromes, S_0, S_1, $\cdots$, S_{14}, S_{15} can be obtained simultaneously due to their parallel structure[3][4]. After 204 clocks of the first byte, 16 syndromes are generated at the same time. The basic hardware unit for syndrome calculation is shown in Fig. 3.

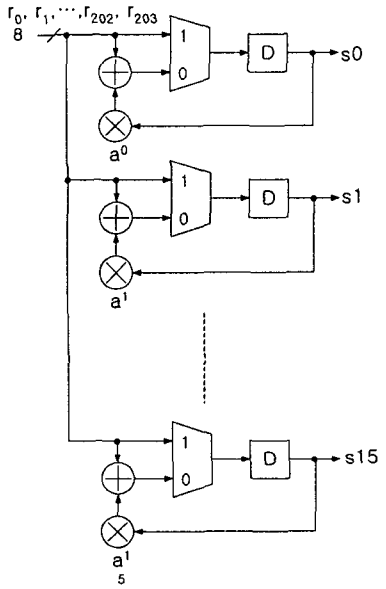

Fig. 3. Architecture of Syndrome Computation.

$$S_k = r(a^k) \qquad \text{for } k = 0,1,2,\cdots,14,15$$
$$= r_{203}(a^k)^{203} + r_{202}(a^k)^{202} + \cdots + r_1 a^k + r_0$$
$$= (\cdots((r_{203}a^k + r_{202})a^k + \cdots + r_1)a^k + r_0) \quad (5)$$

$$S(x) = \sum_{k=0}^{15} S_k x^k \quad (6)$$
$$= S_{15}x^{15} + S_{14}x^{14} + \cdots + S_1 x + S_0$$

2. Modified Euclid Algorithm Block

The modified Euclid algorithm is used in this paper because error location polynomial $\sigma(x)$ and error evaluation polynomial w(x) can be easily obtained without the calculation of inverse element. R(x) register, Q(x) register, start register, $\lambda(x)$ register, and $\mu(x)$ register are used for implementation of the modified Euclid algorithm. The algorithm is used to find the greatest common divisor between Eq. 7 and Eq. 8 for calculating error location polynomials.

$$A(x) = x^{2t} \quad (7)$$
$$S(x) = \sum_{i=0}^{2t-1} S_k x^k \quad (8)$$

Initial conditions of the algorithm are as follows.

$$R_0(x) = A(x) = x^{2t} = x^{16}$$
$$Q_0(x) = S(x)$$
$$\lambda_0(x) = 0$$
$$\mu_0(x) = 1 \quad (9)$$

Error value evaluator polynomial R(x) and error locator polynomial $\lambda(x)$ can be obtained by iterating the following equations with the previous initial condition,

$$R_i(x) = [\sigma_{i-1}\, b_{i-1}\, R_{i-1}(x) + \sigma_{i-1}'\, a_{i-1}\, Q_{i-1}(x)]$$
$$- x^{|l_i - l_{i-1}|}[\sigma_{i-1}\, a_{i-1}\, Q_{i-1}(x) + \sigma_{i-1}'\, b_{i-1}\, R_{i-1}(x)]$$
$$\lambda_i(x) = [\sigma_{i-1}\, b_{i-1}\, \lambda_{i-1}(x) + \sigma_{i-1}'\, a_{i-1}\, \mu_{i-1}(x)]$$
$$- x^{|l_i - l_{i-1}|}[\sigma_{i-1}\, a_{i-1}\, \mu_{i-1}(x) + \sigma_{i-1}'\, b_{i-1}\, \lambda_{i-1}(x)]$$
$$Q_i(x) = \sigma_{i-1}\, Q_{i-1}(x) + \sigma_{i-1}'\, R_{i-1}(x)$$
$$\mu_i(x) = \sigma_{i-1}\, \mu_{i-1}(x) + \sigma_{i-1}'\, \lambda_{i-1}(x) \quad (10)$$

where a_{i-1} and b_{i-1} are the leading coefficients of $R_{i-1}(x)$ and $Q_{i-1}(x)$, respectively, and l_{i-1} =

$\deg[R_{i-1}(x)] - \deg[Q_{i-1}(x)]$. In Eq.(10), σ_{i-1} must satisfy the following condition.

$$\sigma_{i-1} = \begin{cases} 1 & \text{if } l_{i-1} \geq 0 \\ 0 & \text{if } l_{i-1} < 0 \end{cases} \quad (11)$$

The iteration described in Eq. 10 stops when $\deg[R_i(x)] < \deg[\lambda_i(x)]$, and the resulting $\lambda_i(x)$ and $R_i(x)$ represent error locator polynomial $\sigma(x)$ and error value evaluator polynomial $w(x)$, respectively. The flow chart of the algorithm is shown in Fig. 4, and the architecture of the algorithm is also shown in Fig. 5[1]~[6].

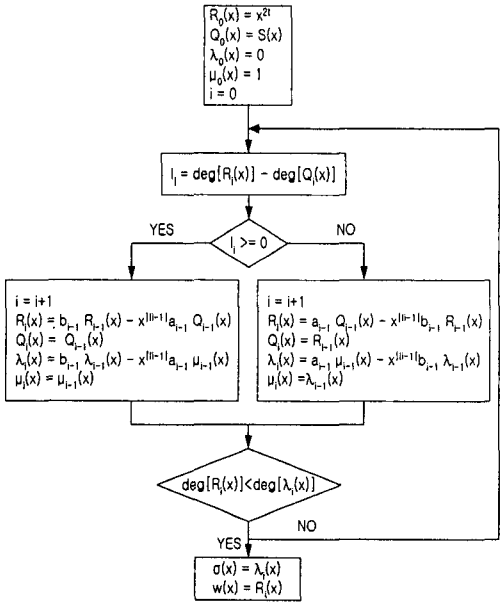

Fig. 4. Flow chart of the modified Euclid algorithm.

When the condition of $\deg[R_i(x)] < \deg[\lambda_i(x)]$ is satisfied, the values of $R_i(x)$ and $\lambda_i(x)$ registers are kept constant. Therefore, the modified Euclid algorithm block can cause the overall speed of the decoder much lower than expected. The problem can be avoided if a higher clock speed is used for the modified Euclid algorithm block[3]. In this work, the clock speed of the block is designed twice of other blocks to reduce latency and therefore to improve the overall speed of the designed decoder.

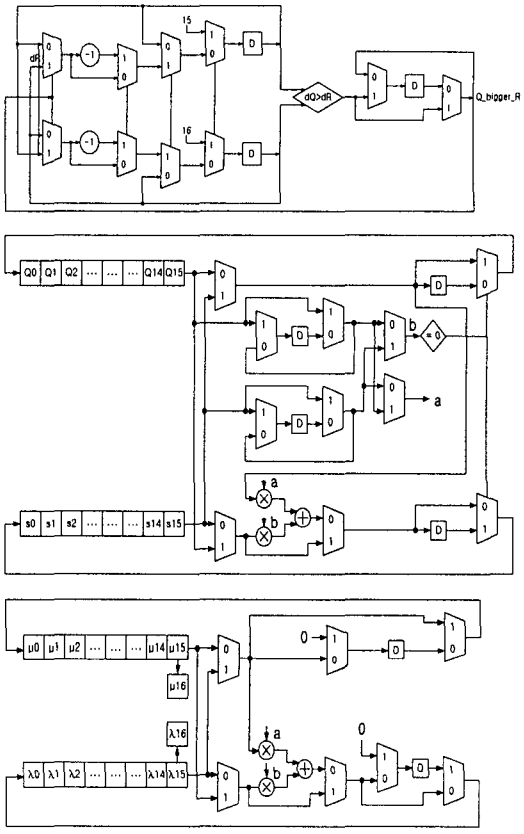

Fig. 5. Architecture of the modified Euclid algorithm.

3. Chien Search Block

Error locator polynomial $\sigma(x)$ and error value evaluator polynomial $w(x)$ calculated at the previous modified Euclid algorithm block will be used in next blocks. The Chien search block is to find the location of errors using $\sigma(x)$, and Forney algorithm block is to calculate the value of errors using $w(x)$.

First, the location of errors can be found by calculating the roots of error locator polynomial with Chien search algorithm. Eq. 12 represents the error locator polynomial $\sigma(x)$ of the (204,188) Reed -Solomon decoder.

$$\sigma(x) = \sigma_0 + \sigma_1 x + \cdots + \sigma_t x^t = \sum_{j=0}^{t} \sigma_j x^j \qquad 0 \leq j \leq t$$

$$= \sum_{j=0}^{8} \sigma_j x^j \qquad 0 \le j \le 8 \qquad (12)$$

With $x = a^i$, Eq. 12 can be expressed as Eq. (13). If $\sigma(x)$ is '0', then the inverse of a^i represents the location of errors. If a^{-i} is the root of $\sigma(x)$, and now a^i represents the location of errors.

$$\sigma(a^i) = \sigma_0 + \sigma_1 a^i + \sigma_2 (a^i)^2 \cdots + \sigma_t (a^i)^8$$

$$= \sum_{j=0}^{8} \sigma_j (a^i)^j \qquad 0 \le i \le 203 \qquad (13)$$

Conventionally, error locator polynomial $\sigma(x)$ is multiplied by 'a' consecutively. Then, the location of errors can be found by calculating the roots of $\sigma(x)$. Finally, errors in the received data can be corrected in next block by exclusive OR operation of w(x) and the received data.

In this paper, error locator polynomial $\sigma(x)$ is separated into even and odd terms as shown in Eq. (14) to increase the overall speed of the decoder. The location of errors can be decided when the sum of $\sigma(x)_{odd}$ 와 $\sigma(x)_{even}$ is '0', and then errors can be corrected as conventional method[3][4]. $\sigma(x)_{odd}$ will also be used as the address of the inverse ROM in Forney algorithm and error correction block to calculate the magnitude of error value.

$$\sigma(x)_{odd} = \sigma_7 x^7 + \sigma_5 x^5 + \sigma_3 x^3 + \sigma_1 x^1 \qquad (14)$$
$$\sigma(x)_{even} = \sigma_8 x^8 + \sigma_6 x^6 + \sigma_4 x^4 + \sigma_2 x^2 + \sigma_0$$

Error value evaluator polynomial w(x) can be obtained in the same way of error locator polynomial $\sigma(x)$ by using Eq. (15).

$$w(a^i) = w_0 + w_1 a^i + w_2 (a^i)^2 \cdots + w_t (a^i)^7$$

$$= \sum_{j=0}^{7} w_j (a^i)^j \qquad 0 \le i \le 203 \quad (15)$$

Fig. 6 shows the architecture of Chien search block.

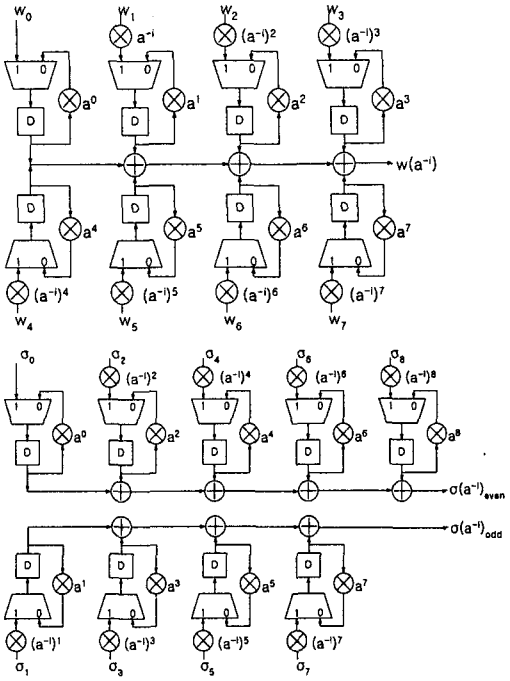

Fig. 6. Architecture of Chien Search Block.

4. Forney Algorithm & Error Correction Block

$\sigma(x)_{odd}$, $\sigma(x)_{even}$ and w(x), which are calculated at Chien search block, are the inputs to Forney algorithm and error correction block. First of all, error locations can be found when the sum of $\sigma(x)_{odd}$ and $\sigma(x)_{even}$ is '0', and then the error value at the error positions can be obtained from Forney algorithm expressed in Eq. (16)[3].

$$e(a^{-i}) = -a^i \frac{w(a^{-i})}{\sigma'(a^{-i})} \qquad 0 \le i \le 203$$

$$= -\frac{w(a^{-i})}{\sigma(a^{-i})_{odd}} \qquad (16)$$

To calculate error values by Forney algorithm, divisions should be done on $GF(2^8)$ as shown in Eq. (16). For easy and fast divisions, Inverse ROM is used to store the values of $1/\sigma(a^{-i})_{odd}$ with the address, $\sigma(a^{-i})_{odd}$. Finally, we can obtain error-corrected data by exclusive OR-ing between 342

clocks-delayed received data through Delay RAM and error values at the location of errors. Fig. 7 shows the architecture of Forney Algorithm & Error Correction block.

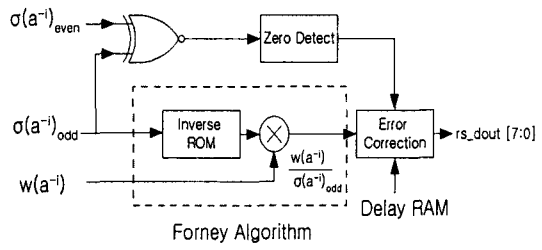

Fig. 7. Architecture of Forney Algorithm & Error Correction.

III. Implementation of Reed-Solomon Decoder

1. Design Procedure

The function of the (204,188) Reed-Solomon decoder is verified first by C++ programs. The hardware implementation of the decoder is achieved by Verilog-HDL, and then the decoder is synthesized for a higher data throughput with a 0.25μm CMOS TLM library. Gate level simulations are performed to check its proper working. Finally, placement & routing is finished to prepare fabrication of the decoder.

2. Architecture

The Reed-Solomon decoder is consisted of 6 blocks as shown in Fig. 8

gen_syn block and *delay_ram* block will have 8 bits input data simultaneously. *gen_syn* block is to generate syndromes, and *delay_ram* block is to delay the input data by 342 clocks. To check the function of the *delay_ram* block, FIFO is used first, and then SRAM(416X8) is used to reduce its size.

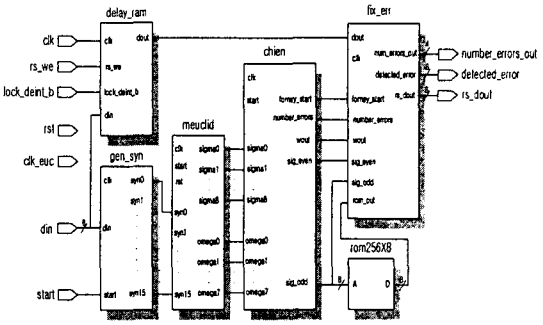

Fig. 8. Architecture of Reed-Solomon Decoder Verilog-HDL.

Generated syndromes will be used to produce error locator polynomial σ(x) and error value evaluator polynomial w(x) through *meuclid* block which performs the modified Euclid Algorithm. *Chien* block for Chien search algorithm generates *sig_odd, sig_even, wout* and *forney_start* signals. The *forney_start* signal pinpoints the starting point of the received code word to correct them, and finally *fix_err* block generates error-corrected data (*rs_dout*) using the outputs of *Chien* block.

Additionally, there are 8-bit output signal (*detected_error*) indicating the error values and 4-bit output signal (*number_errors_out*) indicating the number of errors. Input ports (14 bits) and output ports (20 bits) are summarized in table 1.

Port name	I/O	bits	Function
rst	I	1	reset (active low)
clk	I	1	Main clock
clk_euc	I	1	clock of MEA block
rs_we	I	1	write enable of delay_ram
lock_deint_b	I	1	de-interleaver lock signal (active low)
start	I	1	start signal of reed solomon block
din	I	8	input data
number_errors_out	O	4	error number (up to 8)
detected_error	O	8	error value
rs_dout	O	8	decoded output

Table 1. Input & Output Port.

3. Synthesis

RTL code verified through simulations is synthesized with a 0.25μm CMOS TLM library. Design emphasis is given to high operating speed of the chip.

Care should be taken in designing the modified Euclid algorithm block because its clock speed is designed twice higher than in other blocks. By replacing long delay time gates in the critical path with shorter delay time gates, we obtain satisfactory results. The gate count of each blocks is as follows.

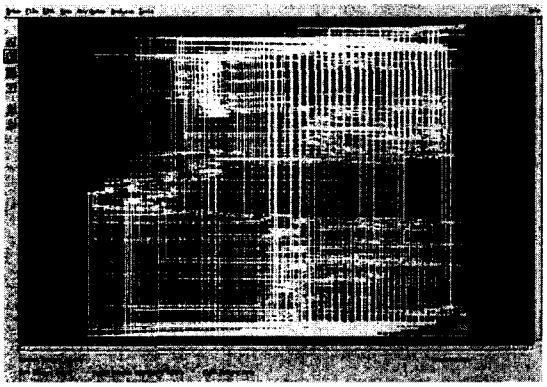

Fig. 9 Synthesis of Reed-Solomon Decoder

- gen_syn block : 1834.00 gates
- meuclid block : 7899.75 gates
- chien block : 2809.50 gates
- fix_err block : 659.00 gates

The total gate number of the Reed-Solomon decoder except memory blocks is about 14,000 in terms of equivalent NAND gate. Fig. 9 shows the Synthesis of Reed-Solomon Decoder.

4. Gate Level Simulation

After synthesis, functioning and timing of each blocks are verified by Model-Sim of Mentor Graphics. Intrinsic delay time, fanout and interconnect delay time caused by routing wires are considered in simulations.

The delay times included in the cell library and the delay factors calculated from the synthesized netlist should be stored as the Standard Delay Format(SDF). The performance of the designed decoder can be simulated at the worst, standard and best conditions using the SDF.

Fig. 10 shows the results of gate level simulation. Among 204 bytes in the code word, 8 bytes are set to '0' from the 101st to 108th bytes. The simulation result shows that those errors are corrected such that the received data is recovered to the original code word.

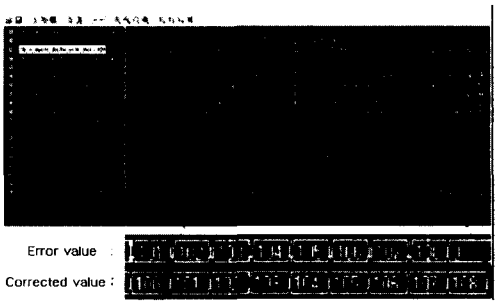

Fig. 10. The results of gate level simulation

Fig. 11 shows the timing diagram of the (204,188) Reed-Solomon decoder.

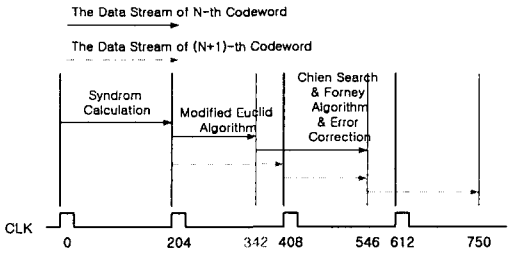

Fig. 11. Timing Diagram of the Reed-Solomon Decoder.

5. Placement & Routing

Since the number of gates in the decoder circuit is relatively small, flat placement & routing is done using Apollo P&R tool. The skew problem among clock trees and the long delay path problem, which often occur in a larger chip, are not of concern in the designed decoder circuit due to its small gate number.

The chip size, pad limited, is 1392.540 x 1320.910

μm^2 with a CMOS 0.25μm TLM process. Fig. 12 shows the layout result of the Reed-Solomon chip.

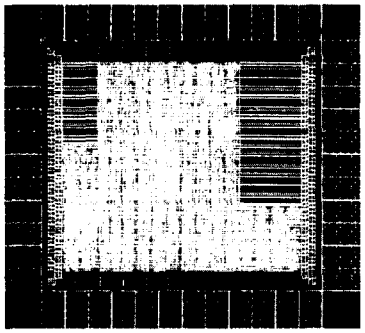

Fig. 12. Chip Layout.

IV. Conclusions

A high speed (204,188) Reed-Solomon decoder, correctable up to 8 bytes, is designed using the modified Euclid algorithm. The decoder is designed suitable for applications of high speed data networks which require stable data transmissions and high bandwidth efficiency.

The functionalities of the decoder are verified through C++ programs first, and then it is designed in pipeline architecture using the systolic array pattern to increase speed. The function and timing are successfully verified through gate level simulations. Finally, P&R is done for fabrications of the decoder.

The decoding speed is increased, and the latency is reduced by using the parallel architecture of syndrome generator and faster clock speed of the modified Euclid algorithm block, and by separating error locator polynomial into even and odd terms in Chien search block. The clock speed of the modified Euclid algorithm block is 250MHz, and 125MHz for other blocks. Its latency shows 342 clocks(2.736μs).

The total number of gates is about 14,000 except memory blocks and the chip size is 1392.540 x 1320.910 μm^2. The designed chip has the maximum data rate, 1Gbps.

References

[1] Howard M. Shao. T. K. Truong "A VLSI Design of a Pipeline Reed-Solomon Decoder", IEEE Trans. Computers, Vol. C-34, NO. 5, pp. 393~403, 1985

[2] Howard M. Shao. Irving S. Reed "On the VLSI Design of a Pipeline Reed-Solomon Decoder Using Systolic Arrays" IEEE Trans. Computers, Vol. 37, NO. 10, pp. 1273~1280, 1988

[3] Hanho Lee " A VLSI Design of A High Speed Reed-Solomon Decoder" IEEE Trans Computer, 0-7803-6741-3/901, pp. 316~320, 2001

[4] Hanho Lee, Meng-Lin Yu, Leilei Song "VLSI Design of Reed-Solomon Decoder Architectures" ISCAS 2000-IEEE, May 28-31 2000, pp. V-705~708

[5] Young-Jin Lim, Moon-ho Lee "A Minimized Modified Euclid Architecture" IEEE, 0-7803-6323-X/00, pp. 177~178, 2000

[6] Dong-Sun Kim, Jong-Chan Choi, Duck-Jin Chung "Implementation of High Speed Reed-Solomon Decoder" IEEE Trans, 0-7803-5491-5/99, pp. 808~812, 1999

[7] Sung-Won Hong, "Performance Analysis of RS/Convolutional codes and Turbo code using Semi Random Interleaver over the Radio Communication Channel", The J. of the Korean Institute of Maritime Information & Communication Sciences, pp. 861~868, Sept. 2001

[8] Hsie-Chia Chang, Chih-Yu Cheng, Shu-Hui Tsai, and Chen-Yi Lee "A (204,188) Reed-Solomon Decoder using Decomposed Euclidean Algorithm" IEEE, 0-7803-6475-9/00, pp. 262~265, Aug 8-11 2000

[9] Shu Lin, Daniel J. Costello, Jr. "Error Control Coding-Fundamentals and Applications" 1983 by Prentice-Hall.

[10] M. Y. Lee, "BCH code and Reed-Solomon code" March, 1990

저자 소개

최영식(Young-Shig Choi)

1982년 2월 경북대학교 전자공학과 (공학사)
1986년 12월 Texas A&M Univ. 전기공학과(공학석사)
1993년 5월 아리조나 주립대 전기공학과 (공학박사)
1987년 2월 ~ 1999년 2월 현대전자 시스템 IC 연구소
1993년 3월 ~ 2002년 2월 동의대학교 전자공학과
2003년 3월 ~ 현재 부경대학교 조교수 전자컴퓨터정보통신공학부

최혁환(Heyk-Hwan Choi)

경북대학교 전자공학과 (공학사)
아리조나 주립대 전기공학과 (공학석사)
아리조나 주립대 전기공학과 (공학박사)
1994년 ~ 현재 부경대학교 부교수
전자컴퓨터정보통신공학부
※ 관심분야 : RF 집적회로 설계, 아날로그 IC 설계