논문 2009-46TC-4-5

리드 솔로몬 복호기의 에러값을 구하기 위한 새로운 고속의 경제적 산술논리 연산장치의 설계에 대해

(New and Efficient Arithmatic Logic Unit Design For Calculating Error Values of Reed-Solomon Decoder)

안 형 근*

(Hyeong-Keon An)

요 약

본 논문에선 리드솔로몬 디코더의 오류위치 탐색장치와 오류치 계산장치중 오류치 계산기의 효율적 설계에 대해 서술한다. 오류치계산은 오류위치가 결정이 되면 선형 연립방정식의 해를 구하면 되나 갈로이스 장상에서 승산장치, 제산장치등의 회로가 구성되져야 한다. 본 논문은 이들 연산회로의 효율적 설계법에 대해 기술하고 있다. 오류위치 계산장치의 설계법은 이미 많은 학자및 기술자들에 의해 연구가 진행되어 여기서는 오류값 계산장치에 대해 주로 연구를 진행 하였다.

Abstract

In This Paper, New Efficient Arithmatic Logic Unit Design for Calculating Error Values of Reed Solomon Decoder is described. Error Values are solved by solving Linear system of Equations ,So called Newtonian set of identity equations. Here We Need Galois Multiplier, Adder, Divider on GF(2^8) field. We prove how the Hardware circuits are improved better than the classical circuits. The method to find error location is not covered here, since many other researchers have already deeply studied it.

Keywords : Reed-Solomon, Decoder, GF(2^8), Error Locator, Arithmatic Logic Unit, Multiplier, Divider, Exor, Linear Systems of Equation

I. Introduction

Reed Solomon coding theory is very Famous well known Nonbinary Error Correction Method For Electronic Devices (Consumer and Communic ation Products)[3, 5].

In this paper, new RS(Reed Solomon) Decoder, which finds out Error Values when its location is already found by using Chien search or any other methods. Normally Error Locations are found by solving non Linear Systems of Equations. I and Many other Researchers studied to solve the Nonlinear Equations to get the Error Locations.

In this Paper we focus on how to get Error Valus when its location is already found by using any methods.

In chapter I Introduction is written to introduce the whole paper. In chapter II, We briefly described how the Newtonian identities are formed to get the Error Values when their locations are already known. These equations are linear system of Equations. In Chapter III, we propose New and Efficient Arithmatic Logic Unit Circuit which generates Error

* 정회원 동명대학교 정보통신공학과
(Tong Myung University Department of Information and Telecommunication Engineering)
접수일자: 2009년2월2일, 수정완료일: 2009년4월16일

values Very quickly and Economically and explain About the circuit. In chapter IV, we show more detail explanation of Divider circuit for ALU(Aritnmatic Logic Unit) using Subfield theory.

In chapter V future works which will be taken by us, and Comparisons Between Old design and New design are described[8].

II. Error Value Generation when its Location is Known

The RS(Reed Solomon) codes are based on finite fields, often called Galois fields.

In CDP, RSC(32,28), on GF(2^8) field, code is used and upto 2 symbol errors can be corrected[7].

An RS code with 8bit symbols will use a Galois field GF(2^8), consisting of 256 symbols. In decoding Reed-Solomon code, we should calculate the Syndromes as in equation 1.

Let

$$C(X)=\sum_{j=0}^{n-1} C_j X^j$$

Be the Transmitted polynomial, and Let

$$R(X)=\sum_{j=0}^{n-1} r_j\ X^j$$

Be the Received polynomial. Then Error pattern of the channel is

$$E(X)=\sum_{j=0}^{n-1} E_j\ X^j$$

Where E_j (j=0 to n-1) are Error values. Here Syndromes are defined as

$$S_i=E(\alpha^i)\ (i=0,1,\cdots,\ 2t-1) \tag{1}$$

for t Error Correcting Code. Here Syndrome Calculator is described in the Book[6].

Now if there are t errors, whose values are E_n (n=0,1,2 ⋯, t-1) and positions are

$$\alpha^{j_n}=\beta_j\ (J=0,1,2\ \cdots\ ,t-1)$$

Then Error Locator Polynomial is defined as

$$\delta(X)=(X-\beta_0)(X-\beta_1)\cdots(X-\beta_{t-1})$$
$$=\sum_{k=0}^{t} X^k\delta_{t-k} \tag{2}$$

Now Newton's Identities are following set of equations.

$$\sum_{j=1}^{t} S_{t-j+v}\delta_j=\ S_{v+t}$$

where v=0,1,2 ⋯, t-1 (3)

From Equations (2) & (3), we can get Error Positions β_i(i=0,1, ⋯, t-1), but it is not easy because they are non linear system of Equations. To find out the Solutions of Equations (2) & (3), there are many ways including Author's Method[2, 7].

After getting the positions, from eq.(1),

$$\begin{pmatrix} E_0 \\ E_1 \\ \bullet \\ \bullet \\ E_{t-1} \end{pmatrix}=\begin{pmatrix} 1 & 1 & .. & 1 \\ \alpha^{j1} & \alpha^{j2} & .. & \alpha^{j(t-1)} \\ & & \cdot & \\ & & \cdot & \\ (\alpha^{j1})^{t-1} & & \cdot\ \cdot & (\alpha^{j(t-1)})^{(t-1)} \end{pmatrix}^{-1}\begin{pmatrix} S_0 \\ S_1 \\ \bullet \\ \bullet \\ S_{t-1} \end{pmatrix} \tag{4}$$

Where, α^{jn}(n=0,1,..,t-1)'s and S_i's are all known Error Positions and Syndromes.

So

$$E_i=\ (\sum_{L=0}^{t-1} A(i)_L S_L)\ /\ (\sum_{k=0}^{N} B(i)_k \alpha^k)$$
$$(i=0,1..,t-1) \tag{5}$$

Here All $A(i)_L$, $B(i)_k$'s are all already found from Equation (4). Since Equation 5 contains only the GF(Galois field) Adding, Multiplying and Dividing operations, The Arithmatic Logic Unit for Computing Eq.(5) (Error Values) needs only circuits for calculating those there operations (Adding, Multiplying and Dividinging)[4, 6].

In Fig. 1 We see Block Diagram of Arithmatic Logic Unit(ALU). It has two inputs and Generates the Error Values. The ALU operation is determined by the OPCODE(2 bit allocated for the ALU operation from the N bit Micro code of CPU.

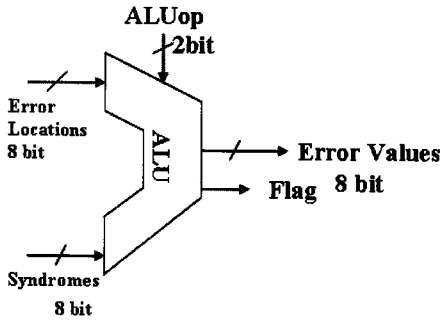


그림 1. 덧셈, 곱셈, 나눗셈회로만을 사용해 식 (4)와 (5)를 계산하는 산술논리연산장치

Fig. 1. ALU which calculates Equations (4) & (5) and uses only Adding, Multip-lying and Dividing Circuits.

In Next section, we describes more About the ALU shown in Fig. 1.

III. Structure of Error Value Generating Arithmatic Logic Unit for Reed Solomon Decoder

The first operation of the ALU is Addition. In $GF(2^8)$ field, Adding is done by 8 bit EXOR gate. The 2nd opertion is Multiplying and is reallized by Multiplier in the ALU shown in Fig. 2, and the Multiplier output goes to ALU output, when Opcode is 01 and the each bit of the Opcode selects proper one of 2 MUX inputs.

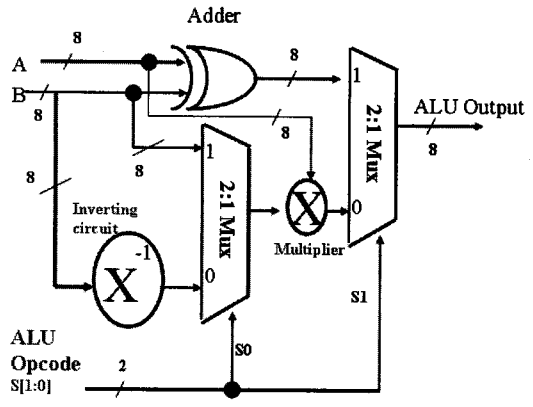


그림 2. 덧셈기, 승산기 그리고 제산기회로를 포함한 산술논리연산장치의 상세 블록도

Fig. 2. Detailed Block Diagram for ALU which contains 3 Operations of Adding, Multiplying and Dividing Circuits.

표 1. 그림 2의 산술논리연산장치를 위한 산술논리연산 장치용 동작코드표

Table 1. ALU Opcode table for the ALU of Fig.2.

OP Code (S[1:0])	Operation
00	A/B (Divide)
01	AB(Multiply)
10	A+B (Add : EXOR)
11	A+B (Add : EXOR)

The 3rd final operation is Dividing and is reallized by multiplying A with Inversed B. In this case ALU Opcode becomes 00 to select outputs of Inverting circuit and Multiplier of the ALU in Fig. 2.

Table 1 shows the Opcode pattern for each 3 ALU operations.

In Next chapter, we show how we implement the various Circuits of the ALU in Fig. 2.

IV. Detailed Explanation of Divider Circuit for ALU using Galois Subfield.

Because Divider contains Multiplier, we only Describe the operation of Divider. As shown in Fig. 3, Divider contains Inverting Circuit and Multiplier Circuit, so to reduce the gate couts of the Divider, we need to reduce the gate counts of the Multiplier and Inverting circuits.

If we use subfield theory[2], we can greately reduce the gate counts of them. Also length of the propagation path is shortened, so speed of the circuits becomes faster than the other case which doesn't use the Subfield theory.

(1) Field Converting Circuit

This circuit contains only EXOR gate and Logic Equation is as follows[2].

IF $A = A0 + A = A0 + \beta A1$, where A, $\beta \in GF(2^8)$ And A0, A1 $\in GF(2^4)$ then, IF A = (b0, b1, b2, b3, b4, b5, b6, b7) and A0 = (Z0, Z1, Z2, Z3), A1 = (Z4, Z5, Z6, Z7)

(i) $GF(2^8)$ to $GF(2^4)$ Converter Logic

$Z0 = b0 + b1 + b5$

$$Z1 = b1 + b3 + b5$$
$$Z2 = b2 + b3 + b6$$
$$Z3 = b1 + b3 + b4 + b6$$
$$Z4 = b1 + b2 + b3 + b5 + b6 + b7$$
$$Z5 = b2 + b5 + b6$$
$$Z6 = b1 + b2 + b3 + b4 + b5 + b6$$
$$Z7 = b1 + b3 + b4 + b5$$

(ii) GF(2^4) to GF(2^8) Converter Logic

$$b0 = Z0 + Z1 + Z2 + Z6 + Z7$$
$$b1 = Z1 + Z2 + Z5$$
$$b2 = Z5 + Z3 + Z7$$
$$b3 = Z2 + Z7 + Z6$$
$$b4 = Z1 + Z7$$
$$b5 = Z7 + Z5 + Z6$$
$$b6 = Z3 + Z5 + Z6$$
$$b7 = Z1 + Z6 + Z4 + Z7$$

(2) Divider Logic Equation using Multiplier and Inverting Circuit.

(i) Multiplier Logic in Fig. 3

Suppose that the element C is the product of the elements A and B.

$$C = A \cdot B$$
$$=(a0 + a1\beta) \cdot (b0 + b1\beta)$$
$$=c0 + c1\beta$$

where a0, a1, b0, b1, c0, c1 $\in$ GF(2^4)

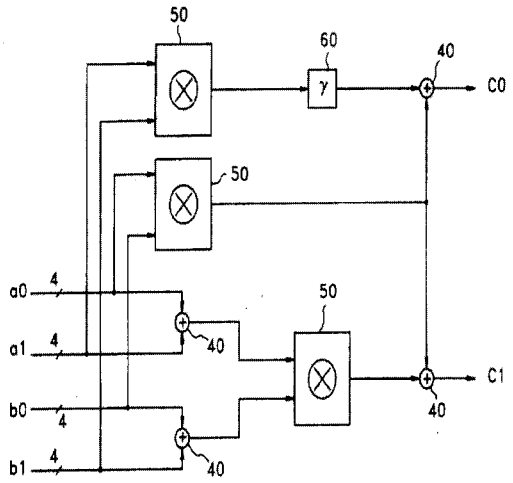


그림 4. GF(2^8)제산기 회로의 부분구성회로로 사용된 GF(2^4)상에서의 곱셈회로

Fig. 4. Mutiplier Circuit in GF(2^4) used as a one part of the GF(2^8) Divider Circuit.
*40: Adder over GF(2^4)
*50: Multiplier over GF(2^4)
*60: Υ Multiplier over GF(2^4)

Then

$$c0 = a0b0 + a1b1\Upsilon$$
$$c1 = a0b1 + a1b0 + a1b1 \quad (6)$$

Fig. 4 shows a block diagram for implementing equations (6)

(ii) Inverting Circuit Logic

Assumming that the inverse of Z is Z^{-1}, and $Z = x0 + x1\beta$ where x0, x1 $\in$ GF(2^4), and $Z^{-1} = y0 + y1\beta$ where y0, y1 $\in$ GF(2^4), then

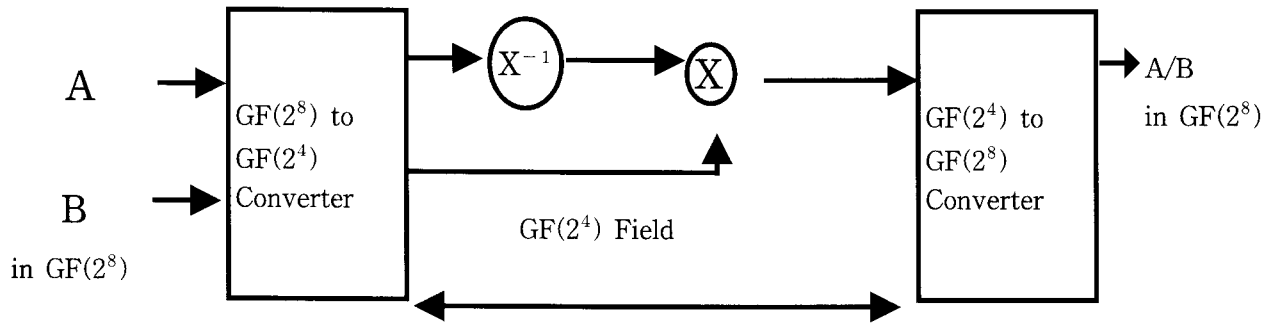
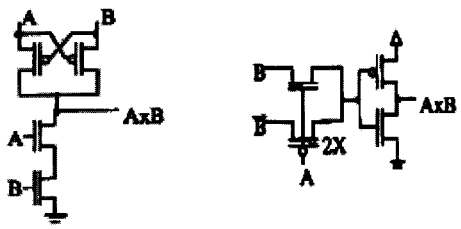


그림 3. 그림 2의 산술논리 연산장치를 위한 제산기회로의 블록도

Fig. 3. Divider Circuit Block Diagram for ALU shown in Fig. 2.



A type B type

그림 5. 상보합 모스형 2종류의 이엑스오알 게이트
Fig. 5. Two types of CMOS EXOR gate.

$$Z \cdot Z^{-1} = 1 \tag{7}$$

From equation (7)

$$y0 = \frac{xo + x1}{x0(x0 + x1) + \gamma(x1^2)}$$

$$y1 = \frac{x1}{x0(x0 + x1) + \gamma(x1^2)} \tag{8}$$

The inverse Circuit represented by (8) is much simpler and faster than inverting Circuit consisted by $GF(2^8)$ elements[4].

Finally for Adding operation of ALU, we use a CMOS EXOR Circuit type A as in Fig. 5[1]. This gate is simpler than the conventional CMOS EXOR gate B type resulting in smaller number of total gate Count.

In Fig. 5, B type EXOR gate needs 6 Transistors including one CMOS inverter, while Type A needs only 4 Transistors.

Operating Example of Dividing Circuit :

Let's find A/B, when $A=\alpha^3$, $B=\alpha^5$ where A, B $\in$ $GF(2^8)$.

Solution: $B = \alpha^5 \in GF(2^8) = \alpha^{12} + \alpha^6\beta \in GF(2^4)$ from the Conversion Logic equation. From Equation (8),

$$y0 = \alpha^{14} / (\alpha^{13} + \alpha^{12}\alpha^{14}) = \alpha^9.$$

Here

$x1 = (\ = (z0,z1,z2,z3) = (1,1,1,1) = \alpha^6$,

and

$$\gamma x1^2 = \gamma\ (z0 + z1\gamma + z2\gamma^2 + z3\gamma^3)^2$$
$$= \gamma\ (z0 + z1\gamma^2 + z2\gamma^4 + z3\gamma^6)$$
$$= \gamma\ (z0 + z1\gamma^2 + z2(1+\gamma^3) + z3\gamma^2(1+\gamma^3))$$
$$= (0,\ 1,\ 1,\ 0) = \alpha^{13} \in GF(2^4).$$

Also $y1 = 1/(\alpha^5 + \alpha^7) = \alpha^{-14} = \alpha \in GF(2^4)$. TheAefore $B^{-1} = y0 + y1\beta = \alpha^9 + \alpha\beta$. Also $A = \alpha^3 \in GF(2^8) = \alpha^8 + \alpha^{11}\beta$ in $GF(2^4)$. So $A/B = A \cdot B^{-1} = (\alpha^8 + \alpha^{11}\beta)(\alpha^9 + \alpha\beta)$.

Now Apply equation (6) which is implemented as the Circuit in Fig. 4.

The result is $c0 = \alpha^2 + \gamma\alpha^{12}$ ($\because \beta^2 = \beta + \gamma$) $= \alpha \in GF(2^4)$. Similarlily, $c1 = \alpha^{12} + \alpha^9 + \alpha^{20} = \alpha^{11} \in GF(2^4)$. So $C = A/B = A \cdot B^{-1} = c0 + c1\beta = \alpha + \beta\alpha^{11} GF(2^4) = \alpha^{253} \in GF(2^8) = \alpha^{-2} \in GF(2^8)$, using $GF(2^4)$ to $GF(2^8)$ transformation. This is Correct !!!($\because$ $A/B = \alpha^3/\alpha^5 = \alpha^{-2} \in GF(2^8)$).

V. Conclusion and Discussion

Because Divider of ALU contains Multiplier as a Sub circuit, we only compare the gate Counts of the Divider of the proposed method and that of classical Method. The divider is consist of Multiplier and Inversing circuit. As we see in Table 2 and 3[2], current method is much better in reducing gate counts Needed. This means Silicon size required for ALU is shrinked to about 1/3.5 of the size for the classical method[6].

For Speed, Because Current proposed method in this paper uses mostly $GF(2^4)$ field operations and uses just once transfer funtion between $GF(2^8)$ and $GF(2^4)$ field, So even more faster ALU operation is made comparing with that of calssical ALU.

표 2. $GF(2^8)$장에서 나누기회로에 필요한 게이트 갯수
Table 2. Number of gates needed for Divider of the $GF(2^8)$ field.

	AND gates	EXOR gates	OR gates	Sub total
Multiplier	64	73		137
Inverse Circuit	304		494	798

Total Number of gates : 935

표 3. 새로운 방법을 이용할 때 나누기회로에 필요한 게이트 개수

Table 3. Number of gates needed for Divider using Present Method

	AND gates	EXOR gates	OR gates	Sub Total
$GF(2^8)$ to $GF(2^4)$		13		13
Multiplier in $GF(2^4)$	48	62		110
Inverse Circuit in $GF(2^4)$	64	56	10	130
$GF(2^4)$ to $GF(2^8)$		13		13

Total Number of gates : 266

In our future work, we will discuss about the more efficient Reed-Solomon Encoder design method.

References

[1] Dajen Huang, "On CMOS EXCLUSIVE OR DESIGN," IEEE Xplore, pp 829-832, 1990.

[2] 안형근, "갈로이스 부분장 변환을 이용한 새로운 고속의 경제적 치엔탐색기의 설계에 대해," 대한전자공학회논문지 TC편, 제44집 3호, pp. 67-73, April 2007.

[3] Hyeong-Keon An, "The New RS ECC Codec for Digital Audio and Video, "IEEE CES Conference paper, pp. 112-115, Lasvegas USA, May 1992.

[4] Lee Man Young, "BCH coding and Reed-Solomon Coding theory," 민음사, 대우아카데믹 프레스, 1990.

[5] 안형근, "디지탈 오디오/비디오,통신용 전자기기를 위한 Reed-Solomon Codec설계에 대해", pp13~18, TC편 제 42호, 2005년 11월호, 대한전자공학회지

[6] Hsu,I.K., Reed," The VLSI Implementation of a Reed-Solomon Encoder Using Berlekamp's Bit-Serial Multiplier Algorithm", IEEE Trans. On Computer ,Vol.C-33,No.10, pp.906-911, 1984.

[7] Hyeong-Keon An, "2 Error Correcting RS Decoder Design," IDEC Conference Paper, 2004.

[8] Shu Lin, Daniel J. Costello Jr, " Error Control Coding," Prentice-Hall , PP 240~261(2004).

저 자 소 개

안 형 근(정회원)

1979년 서울대학교 전기공학과 학사 졸업.

1981년 KAIST 전기및전자 공학과 석사 졸업.

1988년 뉴욕주립대학교 전기공학과 박사 졸업.

1988~1998 삼성전자 수석연구원

1998~1999 Telson전자 이사

2000~현재 동명대학교 정보통신공학과 교수

<주관심분야 : 통신, 신호처리, 반도체, OLED/LED display 조명장치