1999 年 度 大 韓 電 子 工 學 會 夏 季 綜 合 學 術 大 會 論 文 集 第 22 卷 第 1 號 99/6

Pipeline (15,9) Reed-Solomon decoder의 VLSI 설계

김 기 욱, 송 인 채
숭실대학교 전자공학과
전화 : (02) 816-6073 / 팩스 : (02) 821-7653

A VLSI Design of a Pipeline (15,9) Reed-Solomon Decoder

Ki-Uk Kim, Inchae Song
School of Electronic Engineering, Soongsil University
E-mail : michael@hanul.soongsil.ac.kr

Abstract

In this paper, we designed a pipeline (15,9) Reed-solomon decoder. To compute the error locator polynomials, we used the Euclidean algorithm. This algorithm includes computation of inverse element. We avoided the inverse element calculation in this RS decoder by using ROMs. We designed this decoder using VHDL. Simulation results show that the designed decoder corrects three error symbols. We implemented this design through an Altera FPGA chip.

I. 서론

통신에서 정보전달에 따르는 오류를 정정하는 방식에는 FEC(Forward Error Correction)와 ARQ(Automatic Repeat Request)가 있는데 이중에서 FEC는 에러 정정 코드(error-correcting code)를 사용하여 통신 정보의 신뢰도를 높이는 방법이다[1].

에러 정정 코드는 코드워드(code word)간의 최소거리(minimum distance)가 클수록 에러 정정 능력이 커진다. nonbinary BCH (Bose Chadhuri Hocquenhem) 코드의 특수한 형태인 RS(Reed-Solomon) 코드는 최소거리가 가장 큰 코드이다[2].

전체적으로 성격이 다른 block들로 이루어져 있는 RS decoder는 pipeline 방식으로 설계하면 속도 면에서 많은 이점을 얻을 수 있다. 본 논문은 이런 pipeline 방식으로 구현하였다.

RS decoder의 가장 중요한 요소인 error-locator polynomial 계산은 Euclidean algorithm을 사용하였다. Euclidean algorithm을 사용할 경우 나눗셈 계산으로 회로가 복잡해지고 나눗셈 시간으로 인해 결과 값이 늦어지게 되는데 본 논문에서는 나눗셈을 사용하지 않고 ROM을 사용해 계산을 함으로써 간단하게 설계할 수 있었다.

설계의 방법으로는 VHDL을 사용하였고, Altera사의 Max2plus로 simulation을 하였고, synopsys를 이용해서 합성과 회로를 추출하였다.

II. Reed-Solomon Code

RS code는 nonbinary BCH 코드의 한 특수한 경우이다[1].

GF(q)상에서 정보 symbol의 길이가 k이고 부호 symbol 길이가 n 이고 코드 워드 안에서 오류 정정 능력이 t인 RS code를 (n,k) RS code라고 한다[2]. 본 논문에서 설계한 Reed-Solomon code의 규격은 GF(16)상에서 정의되는 9개의 정보 심볼에 6개의 redundancy를 붙여서 만든 오류 정정 능력 t=3 인

(15,9) RS decoder이다.
(15,9) RS code의 생성 다항식(generator polynomial) g(x)는 다음과 같다.

$$g(x) = (x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)(x+\alpha^5)(x+\alpha^6)$$

g(x)의 차수는 오류 정정 능력 t와 관계가 있는데 g(x)의 차수는 2t=6 이다. (15,9) RS code의 최소거리는 2t+1=n-k+1=7이다.[1]

1 symbol은 4bit로 구성되므로 (15,9) RS code 는 (60,36) binary 부호이다.

$$C= \underbrace{1100}_{c_0}\ \underbrace{0001}_{c_1}\ \underbrace{1000}_{c_2}\ \cdots\ \underbrace{1000}_{c_{14}}$$

Ⅲ. Reed-Solomon decoder의 설계

RS decoder의 방식에는 대표적으로 Algebraic decoding 방식과 Transform decoding 방식이 있다[2]. 본 논문에서는 Transform decoding 방식을 사용한다. Transform decoding 방식에는 Error locator polynomial을 계산하기 위해서 Berlekamp-Massey, Euclidean, continued-fraction algorithm등 많은 방법이 소개되어 있는데 본 논문에서는 VLSI 설계에 적합한 Euclidean algorithm을 이용해서 설계를 하였다.

Euclidean algorithm을 이용한 RS decoder의 복호 과정을 다음의 5 단계로 표현 할 수 있다.

제 1 단계 : g(x) (generator polynomial)의 근을 이용한 syndrome S_k의 계산

$$S_k = \sum_{n=0}^{N-1} r_n \alpha^{nk}, \qquad 1 \le k \le 2t \qquad (1)$$

여기서 r_n $(0 \le n \le N-1)$ 은 입력으로 받은 신호이다. 그리고 $S_k = E_k$ $(1 \le k \le 2t)$ 이다. E_k 는 error pattern 의 transform 형태이다.

제 2 단계 : x^{2t}와 syndrome polynomial을 이용한 error locator polynomial $\sigma(x)$의 계산(Euclidean algorithm) [3],[4]

Euclidean algorithm은 다음과 같은 점화식으로 되어 있다.

$$\gamma_i(x)A(x) + \lambda_i(x)S(x) = R_i(x) \qquad (2)$$

(여기서 $A(x) = x^{2t}$, $S(x) = \sum_{k=1}^{2t} S_k x^{2t-k}$)

식(2)를 통해 다음 식

$$\lambda(x) = \lambda_0 x^t + \lambda_1 x^{t-1} + \cdots + \lambda_t \qquad (3)$$

을 구할 수 있고 여기서 polynomial $\lambda(x)$를 0이 아닌 첫째 계수 λ_0로 나누면 error locator polynomial

$$\sigma(x) = x^t + \sigma_1 x^{t-1} + \cdots + \sigma_t \qquad (4)$$
$$(\sigma_k = \lambda_k / \lambda_0 \ (\lambda_0 \neq 0) \ , 1 \le k \le t)$$

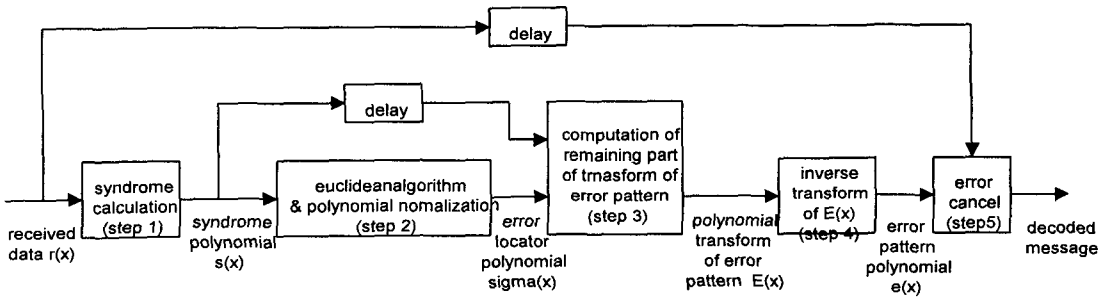
을 구할 수 있다.

제 3 단계 : σ_k를 이용한 나머지 transform error pattern (E_k)의 계산 [5]

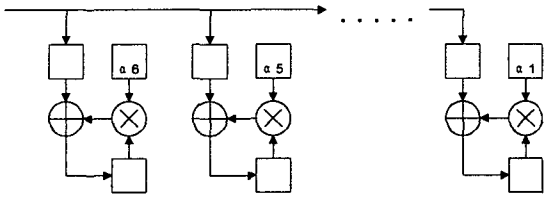
$$E_{2t+j} = \sum_{k=1}^{t} \sigma_k E_{2t-k+j}, \qquad j \ge 1 \qquad (5)$$

제 4 단계 : E_k 의 inverse trnasform을 통해 error pattern 의 추정

$$e_n = \sum_{k=0}^{N-1} E_k \alpha^{-nk}, \qquad 0 \le n \le N-1 \qquad (6)$$



[그림 1] (N,K) Reed-Solomon decoder의 block diagram



[그림 2] syndrome 계산 회로

제 5 단계 : Error 의 정정

입력받은 신호로부터 error pattern을 비교함으로써 error를 정정할 수 있다.

이와 같은 총 5단계를 block diagram으로 [그림1]에 나타내었다.

1. Clock generator

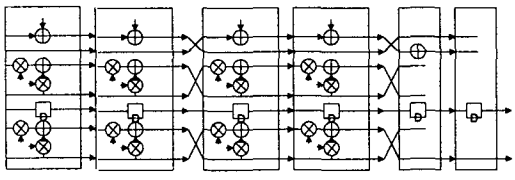
디지털 설계의 가장 중요한 것 중에 하나는 회로의 clock timing 문제이다. 그리고 pipeline 방식으로 설계를 하려면 각 회로들을 입력신호의 한 data의 크기인 60bit 단위로 제어해야 한다. 이를 위해 60bit one hot generator가 필요하게 된다.

2. Signal interleaver 회로

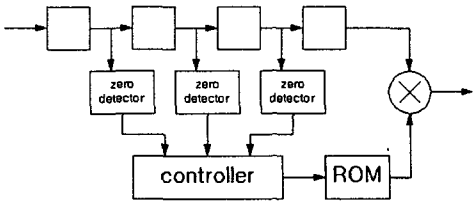
설계하는 (15,9) RS decoder 에서는 한 symbol은 4bit로 이루어져 있다. 그래서 decoder에서는 symbol (4bit) 단위로 계산을 하게 된다. 하지만 RS decoder에 들어오는 입력 data는 serial로 들어오게 됨으로 symbol 단위로 계산을 하기 위해서 입력 data를 4bit 씩 parallel로 바꿔 주어야 한다. 이를 위해 signal *interleaver*가 필요하게 된다.

3. Syndrome 계산회로

(15,9) RS decoder에서는 syndrome을 구하기 위해서 생성다항식의 근인 α, α^2, α^3, α^4, α^5, α^6 곱하면 된다. 이 회로의 회로도는 [그림 2]에서 보여진다. 여기서 곱셈이 사용되는데 곱셈은 GF(16)상의



[그림 3] error locator polynomial 회로



[그림 4] polynomial nomalization 회로

수 체계의 특성을 이용하여 다음과 같이 XOR 계산으로 바꿀 수 있다.

예를 들어 송신된 symbol r을

$$r = b_0 + b_1\alpha + b_2\alpha^2 + b_3\alpha^3$$

과 같이 표현할 수 있다. syndrome을 구하기 위해 α를 곱하는 것을 살펴보면

$$r \times \alpha = b_3 + (b_0+b_3)\alpha + b_1\alpha^2 + b_2\alpha^3$$

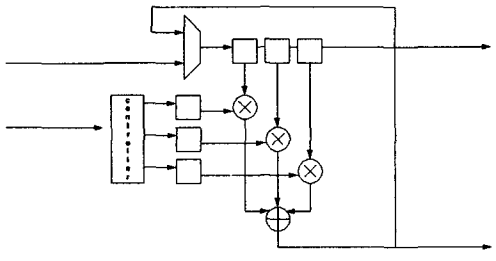
과 같이 바꿀 수 있다[2].

4. Error locator polynomial 회로 (Euclidean algorithm)

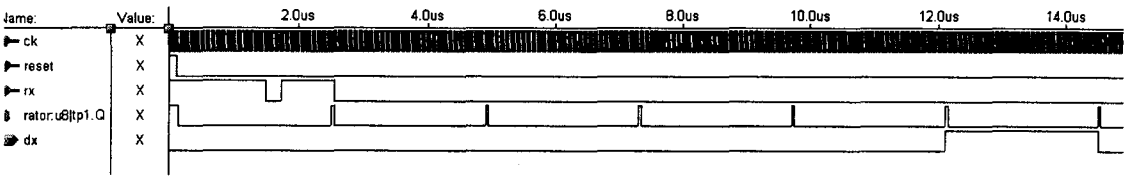
Error locator polynomial 회로는 (15,9) RS decoder의 한 pipeline 주기인 60 clock cycle 이상의 clock이 소요되므로 두 block으로 나누어서 설계하였다.

첫 번째 block은 바로 Euclidean algorithm을 이용해서 syndrome polynomial S(x)를 λ(x)까지 구하는 회로이다. Algorithm은 점화식 형태로 i의 값을 증가시켜가며 식(2)을 계속 증가 시켜 나가다가 $R_i(x)$의 차수가 오류 정정 능력 t 보다 작게 될 때 멈추게 되고 그때의 λ(x)의 값을 구할 수 있게 된다[3],[4]. (15,9) RS decoder의 경우에는 6번 진행시키게 된다.

이 회로의 회로도는 [그림 3]에 나타나 있다.



[그림 5] Error pattern calculate 회로



[그림 6] 전체회로의 simulation 결과

5. Polynomial nomalization 회로

식(4)에서 살펴 본 것과 같이 error locator polynomial $\sigma(x)$를 구하기 위해서는 $\lambda(x)$의 0이 아닌 첫째 계수로 나누어야 하는 회로가 필요하다[3],[4].

식(4)에서 보듯이 여기서 나눗셈 회로가 필요하게 되는데 나눗셈 계산으로 인해 회로가 복잡해지고 시간이 늦어지므로 나눗셈을 피하기 위해 먼저 λ_0^{-1} 값들을 ROM에 저장했다가 λ_1 과 λ_2를 곱해서 σ_0와 σ_1을 구한다. 이 회로의 회로도는 [그림 4]에 나타나 있다.

6. Error pattern calculate 회로

이 회로는 error locator polynomial $\sigma(x)$과 syndrome polynomial S(x)을 이용해서 나머지 transformed error pattern E(x)를 구하는 회로이다[5].

식(5)을 이용해서 구할 수가 있다. (15,9) RS decoder에서는 t= 3 이다. 이 회로의 회로도는 [그림 5]에 나타나 있다.

7. Error pattern generation 회로

전 block에서 구하여진 transformed error pattern을 inverse transform을 이용해서 error pattern을 구하는 회로이다. 회로의 모습은 [그림 2]의 syndrome 계산회로와 같다. 다만 α^{15} 까지 곱하는 회로가 같이 쓰인다[2].

8. Error cancel 회로

구하여진 error pattern과 처음에 입력으로 받은 received pattern을 XOR 계산을 통해 error를 cancel 시키는 회로이다.

Ⅳ. Simulation 및 검토

앞에서 살펴본 모든 block들을 합성한 전체적인 회로의 simulation 결과는 [그림 6]에 나타나 있다.

Simulation을 위한 data는 다음과 같다. encoded word는 c = ($\alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}$) 이고 error pattern은 e = (0,0,0,0,0,0,0,0,0, $\alpha^0, \alpha^{12}, \alpha^3$,0,0,0) 이다. 따라서 입력값은 c+e = ($\alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{12}, \alpha^{11}$,0, $\alpha^5, \alpha^{12}, \alpha^{12}, \alpha^{12}$) 이다. [그림 6]에서 data를 입력 한 후에 올바르게 decode된 값이 출력됨을 알 수 있다.

Ⅴ. 결론

본 논문에서는 pipeline 방식으로 euclidean algorithm을 이용해서 (15,9) RS decoder를 설계하였다. ROM을 사용해서 나눗셈 계산을 간단하게 처리 하였다. simulation 결과 3개의 오류 data를 정정하는 것을 확인 할 수 있었다. Altera 사의 EPF10K70RC240-4 chip을 이용해서 FPGA로 구현하였다.

참고문헌

[1] Peter Sweeney, "ERROR CONTROL CODING", Prentice Hall, 1991.

[2] Stepphen B.Wicker "Reed-Solomon Code and Their Applications", IEEE PRESS

[3] Howard M. Shao, "On the VLSI Design of a Pipeline Reed-Solomon Decoder Using Systolic Arrays", *IEEE Trans. on computer* , vol 37 NO.10, pp 1273-1280, October 1988

[4] Howard M. Shao, and Irving S. Reed, " A VLSI Design of a Pipeline Reed-Solomon Decoder", *IEEE Trans. Inform. Theory*, vol C-34 NO.5, pp 393-402 May 1985

[5] Irving S. Reed, "The Fast Decoding of Reed-Solomon Code Using Fermat Theoretic Transforms and Continued Fractions", *IEEE Trans on Information Theory*, vol IT-24, No 1, pp 100-104, January 1978