2001年度 第14回 信號處理合同學術大會 論文集 第14卷 1號

시스톨릭 어레이를 이용한 고속 병렬처리 Reed-Solomon 복호기 설계

강진용, 선우명훈
아주대학교 전자공학부

Design of a High Speed and Parallel Reed-Solomon Decoder Using a Systolic Array

Jin Yong Kang and Myung Hoon Sunwoo
School of Electronics Engineering, Ajou University
E-mail : comang@madang.ajou.ac.kr

요약

본 논문에서는 연집 오류(burst error)에 우수한 정정 능력을 보이는 고속 RS(Reed-Solomon) 복호기를 제안한다. 제안된 RS 복호기는 RS(n, k, t); ($37 < n \leq 255$, $21 < k \leq 239$, $t = 8$)의 사양을 지원하며 수정 유클리드 알고리즘(modified Euclid's algorithm)을 이용한 시스톨릭 어레이(systolic array) 방식의 병렬처리 구조로 설계되었다. 고속 RS 복호기의 효율적인 VSLI 설계를 위하여 새로운 방식의 수정 유클리드 알고리즘 연산 회로를 제안한다. 제안된 수정 유클리드 알고리즘 회로는 $2t + 1$의 연산 지연 시간을 갖으며 기존 구조의 연산 지연 시간인 $3t + 37$에 비하여 $t = 8$ 인 경우 약 72%의 연산 지연이 감소하였다. 제안된 구조를 VHDL을 이용하여 설계하였으며 SAMSUNG 0.5㎛(KG80) 라이브러리를 이용하여 논리 합성과 타이밍 검증을 수행하였다. 합성된 RS 복호기의 총 게이트 수는 약 77,000 개이며 최대 80MHz의 동작 속도를 나타내었다.

1. 서론

FEC(Forward Error Correction) 기법 중에서 연집 오류 정정에 우수한 성능을 보이는 RS 부호는 위성 통신, 이동 통신, WATM(Wireless Asynchronous Transfer Mode) 망, 디지털 저장 장치, HDTV 등 다양한 응용 분야에서 사용되고 있다. 또한 최근 신뢰성 있는 고속 멀티미디어 데이터 전송을 위해 수백 Mbps 급의 전송 속도를 만족하는 RS 복호기의 필요성이 대두되었다. 따라서 고속 멀티미디어 데이터 전송이 가능하고 작은 면적으로 구현이 가능한 RS 복호기가 많이 발표되었다. RS 복호기는 부호기에 비해 복잡한 알고리즘과 연산 회로가 필요하기 때문에 작은 면적과 고속 구현이 가능한 효율적인 복호 알고리즘과 구조들이 연구되어 왔다. 대표적인 복호 알고리즘으로는 Berlekamp-Massey 알고리즘[1-2], 유클리드 알고리즘[3-4], 수정 유클리드 알고리즘[5-7]이 있다. 본 논문에서는 RS 복호 알고리즘 중에서 성능이 우수하고 하드웨어 구현이 용이한 수정 유클리드 알고리즘을 이용한 새로운 구조의 RS 복호기를 제안한다.

본 논문의 2 장에서는 제안된 RS 복호기의 구조와 동작에 대해 설명한다. 3 장에서는 시뮬레이션을 통하여 제안된 복호기의 동작을 검증하고 기존 구조와 비교를 통하여 성능을 평가한다. 마지막으로 4 장에서 결론을 맺는다.

2. 제안된 RS 복호기의 구조

그림 1에 제안된 RS 복호기의 전체 구조를 나타냈다. 제안된 복호기는 신드롬 다항식 계산 블록, 수정 유클리드 알고리즘 연산 블록, Chien Search 및 Forney 알고리즘 연산 블록, FIFO로 구성된다. 신드롬 계산 블록에서는 생성 다항식(generator polynomial)의 근을 이용하여 오류 패턴(error pattern)을 나타내는 신드롬 다항식(syndrome polynomial)을 구한다. 수정 유클리드 알고리즘 연산 블록은 오류 크기 다항식(error value polynomial)과 오류 위치 다항식(error locator polynomial)을 구하기 위한 키 방정식(key equation) 연산을 수행한다. Chien Search 및 Forney 알고리즘 연산 블록은 오류 크기 다항식과 오류 위치 다항식을 이용하여 데이터의 오류 크기와 오류 위치를 구한다. FIFO는 입력 값을 연산 시간동안 지연 시켜 수신 데이터의 오류를 정정하여

본 연구는 한국과학재단(No. 97-0100-1401-5)과 반도체 설계교육센터(IDEC)의 지원을 받아 수행되었음.

출력하기 위한 지연 버퍼로 사용된다.

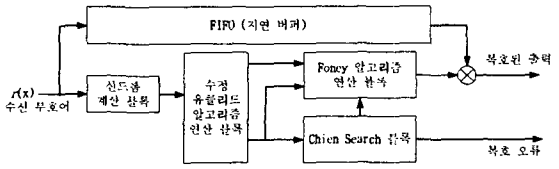

그림 1. RS 복호기의 전체 구조

수신 부호어 $r(x)$ 는 식 (1)과 같이 표현되며 $c(x)$ 와 $e(x)$ 는 각각 부호기의 출력인 송신 부호어와 채널 오류를 나타낸다.

$$r(x) = c(x) + e(x) \tag{1}$$

부호 다항식 $c(x)$ 는 정보 다항식 $m(x)$ 를 x^{n-k} 만큼 쉬프트시키고 정보 다항식 $g(x)$ 로 나눈 나머지 다항식 $p(x)$ 에 쉬프트시킨 정보 다항식을 더하여 생성하며 식 (2)와 같이 표현할 수 있다. $q(x)$ 는 몫 다항식을 나타낸다.

$$c(x) = x^{n-k}m(x) + p(x) = q(x)g(x) \tag{2}$$

2.1 신드롬 계산

그림 2는 신드롬 계산 블록을 나타낸다. 신드롬 계산은 RS 부호기에서 사용된 생성 다항식의 근 α^i 를 이용한다.

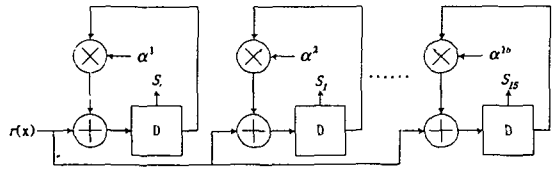

그림 2. 신드롬 계산 블록

송신 부호 다항식은 식 (2)와 같이 몫 다항식과 정보 다항식의 곱으로 표현이 되므로 생성 다항식의 근을 대입하면 부호 다항식은 0이 된다. 따라서 수신 부호어에 생성 다항식의 근을 대입하면 $r(x) = e(x)$ 가 된다. 식 (3)은 계산된 신드롬의 정의를 나타낸 것이다.

$$S_{i-1} = e(\alpha^i), \qquad i = 1, 2, \ldots 2t \tag{3}$$

채널에서 오류가 발생하지 않았을 경우 신드롬은 0이 되며 오류가 발생했을 경우에는 신드롬이 0이 아닌 값을 갖게 된다. 계산된 신드롬을 이용하여 신드롬 다항식을 식 (4)와 같이 정의한다.

$$S(x) = \sum_{i=0}^{2t-1} S_i x^i \tag{4}$$

생성된 신드롬 다항식은 수정 유클리드 알고리즘 연산 블록에서 오류 위치와 오류 크기를 계산하기 위해 사용된다.

2.2 수정 유클리드 알고리즘 연산

키 방정식을 풀기 위한 수정 유클리드 알고리즘은 유클리드 알고리즘 연산시 유한체 역원의 이용을 위해 필요한 LUT(Look Up Table)가 필요하지 않으므로 LUT 사용으로 발생하는 속도 지연을 줄일 수 있는 장점이 있다[5].

수정 유클리드 알고리즘을 이용하여 키 방정식을 풀기 위해 식 (5)와 같이 초기 값을 정하고 식 (6), 식 (7)의 연산을 최대 2t 번 반복 수행한다.

$$R_0(x) = x^{2t} \quad Q_0(x) = S(x) \tag{5-1}$$
$$\lambda_0(x) = 0 \quad \mu_0(x) = 1 \tag{5-2}$$

$$R_i(x) = [\sigma_{i-1}b_{i-1}R_{i-1}(x) + \bar{\sigma}_{i-1}a_{i-1}Q_{i-1}(x)] - x^{|l_{i-1}|}[\sigma_{i-1}a_{i-1}Q_{i-1}(x) + \bar{\sigma}_{i-1}b_{i-1}R_{i-1}(x)] \tag{6-1}$$

$$\lambda_i(x) = [\sigma_{i-1}b_{i-1}\lambda_{i-1}(x) + \bar{\sigma}_{i-1}a_{i-1}\mu_{i-1}(x)] - x^{|l_{i-1}|}[\sigma_{i-1}a_{i-1}\mu_{i-1}(x) + \bar{\sigma}_{i-1}b_{i-1}\lambda_{i-1}(x)] \tag{6-2}$$

$$Q_i(x) = \sigma_{i-1}Q_{i-1}(x) + \bar{\sigma}_{i-1}R_{i-1}(x) \tag{7-1}$$
$$\mu_i(x) = \sigma_{i-1}\mu_{i-1}(x) + \bar{\sigma}_{i-1}\lambda_{i-1}(x) \tag{7-2}$$

a_{i-1} 와 b_{i-1} 는 각각, 의 최고차항의 계수이다.

$$l_{i-1} = \deg(R_{i-1}(x)) - \deg(Q_{i-1}(x))$$
$$\sigma_{i-1} = 1 \quad \text{if } l_{i-1} \geq 0$$
$$\sigma_{i-1} = 1 \quad \text{if } l_{i-1} < 0$$

$\deg(R_{i-1}(x)) < t$ 이면 다항식 연산을 중지한다. 연산이 완료되면 $\lambda(x)$ 와 $R(x)$ 가 각각 오류 위치 다항식과 오류 크기 다항식이 된다.

그림 3 은 제안한 수정 유클리드 알고리즘 연산 회로의 전체 블록을 나타낸 것이다. 다항식 $R(x)$ 와 $Q(x)$ 를 연산하는 17 개의 상위 셀과 다항식 $\lambda(x)$ 와 $\mu(x)$ 의 연산을 수행하는 18 개의 하위 셀로 이루어져 있다. 총 35 개의 기본 셀이 시스톨릭 어레이로 구성되어 있으며 다항식 연산을 위한 기본 셀의 제어는 하나의 독립된 제어 블록에서 이루어진다. 제어 블록에서는

기본 셀의 레지스터 제어 신호인 로드 0, 로드 1 과 다항식 연산 제어 신호인 지시신호를 생성한다. $R_{i-1}(x)$ 과 $Q_{i-1}(x)$ 의 최고차항 계수가 0 인지 아닌지를 판별하고 그 결과에 따라 4 가지 상태의 지시신호를 생성한다. 각 셀의 번호는 초기 입력되는 다항식의 차수와 대응되며 기본 셀 16 에 항상 최고차항의 계수가 위치하도록 다항식 연산과 계수의 이동을 반복한다.

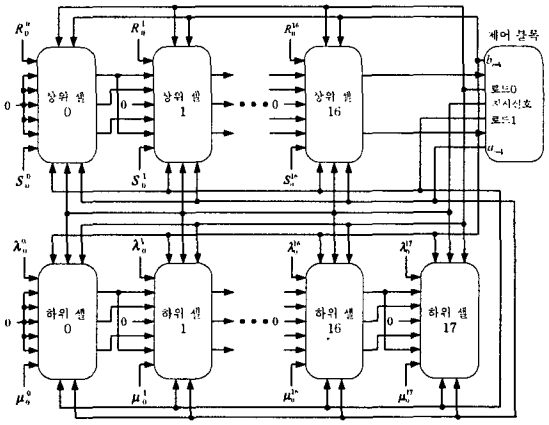

그림 3. 전체 수정 유클리드 알고리즘 블록

제안한 구조는 기본 셀이 차수 계산 회로와 다항식 연산 제어 회로를 포함하지 않으므로 기본 셀이 매우 단순한 구조를 갖는다. 그림 4 는 기본 셀의 구조를 나타낸 것이다. 상위 셀과 하위 셀의 구조는 동일하며 기본 셀의 입력과 출력에 따라 구분된다. 괄호 밖의 문자는 상위 셀의 입출력이며 괄호 안의 문자는 하위 셀의 입출력을 나타낸다. k 는 셀의 번호를 나타낸다.

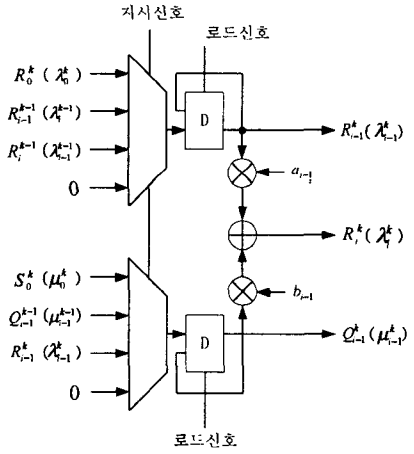

그림 4. 수정 유클리드 알고리즘 기본 셀 구조

제안한 수정 유클리드 알고리즘 연산 구조는 기존 병렬처리 구조[5,6]를 보완하여 오류 패턴에 관계 없이 반복 회수를 2t + 1 로 고정하였다. 최종 결과인 오류 위치 다항식과 오류 크기 다항식을 얻기 위해 2t + 1 의 클럭만이 필요하므로 고속 복호가 가능한 구조이다.

2.3 Chien Search 및 Forney 알고리즘 연산

오류 위치 다항식과 오류 크기 다항식을 이용하여 식 (8)과 같이 오류 값을 계산한다.

$$e_{i_k} = \frac{-\Omega(X_k^{-1})}{\Lambda_k'(X_k^{-1})} \tag{8}$$

$\Omega(x)$ 는 오류 크기 다항식이며 $\Lambda'(x)$ 는 오류 위치 다항식의 미분 값을 나타낸다. 오류 위치 다항식의 근이 X_k^{-1} 이며 이 값의 역수인 X_k 가 오류 위치가 된다. 식 (9)를 이용하면 오류 위치 다항식의 근을 구하기 위한 회로의 하드웨어 구현이 용이하다.

$$\lambda(\alpha^i) = \alpha^i(\alpha^i(\cdots(\alpha^i(\lambda_8\alpha^i + \lambda_7) + \lambda_6)\cdots) + \lambda_1) + \lambda_0 \tag{9}$$

그림 5 는 Chien Search 구조를 나타낸 것이다. $\alpha^i(2^m - n \le i \le n-1)$ 를 차례로 대입하여 S^i 가 0 인 경우의 X_k^{-1} 가 오류 위치 다항식의 근이 된다.

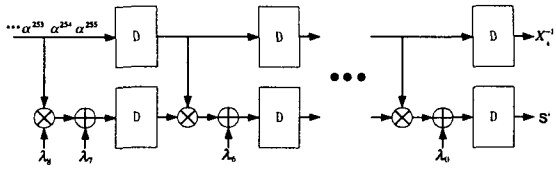

그림 5. Chien Search 구조

그림 6 은 Forney 알고리즘 연산 블록을 나타낸 것으로 식 (8)의 분모 항의 계산을 위하여 유한체 역원을 저장하기 위한 Inverse ROM 테이블이 사용되었다. Chien Search 및 Forney 알고리즘 연산 구조는 파이프라인 기법이 적용되어 고속 연산이 가능하며. 각각 t 의 지연 시간을 갖는다.

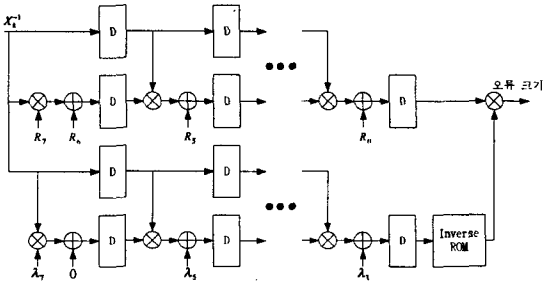

그림 6. Forney 알고리즘 연산 구조

3. 검증 및 성능 비교

제안된 RS 복호기를 VHDL을 이용하여 설계하였으며 SAMSUNG 0.5㎛(KG80) 라이브러리를 이용하여 논리 합성을 수행하였다. 논리 합성은 SYNOPSYS®의 Design Compiler™를 이용하였으며 CADENCE® 의 Verilog-XL™을 이용하여 타이밍 검증을 수행하였다.

그림 7은 설계된 RS 복호기의 시뮬레이션 파형을 나타낸 것이다. 원으로 표시된 부분이 오류가 정정되는 것을 나타낸다. FIFO에 저장된 수신 신호(fifoout) 0x80에 0x01의 오류 값(err_val)이 검출되어 정정 결과인 0x81 값이 복호기 출력(RSout)으로 되는 것을 알 수 있다. 이 값은 오류가 없는 송신된 신호(tx)와 같다.

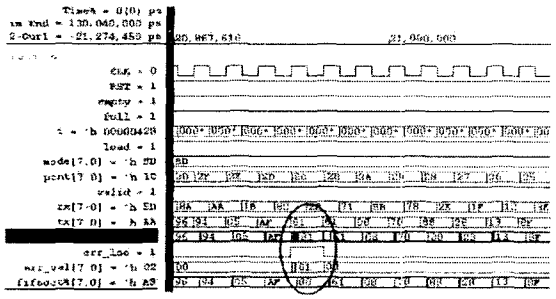

그림 7. RS 복호기의 시뮬레이션 파형

합성된 RS 복호기의 총 게이트 수는 약 77,000 개이며 최대 80MHz의 동작 속도를 나타내었다.

표 1은 제안된 RS 복호기의 수정 유클리드 알고리즘 구조의 하드웨어 복잡도를 기존 구조[7]와 비교한 것이다. 제안된 구조는 수정 유클리드 알고리즘 연산에 2t + 1의 클럭 지연이 발생하며 Chien Search 및 Forney 알고리즘 연산에 2t의 클럭 지연이 발생하여 신드롬 출력 이후 총 4t + 1의 클럭 지연이 발생한다. 기존 구조의 경우 3t + 41의 클럭 지연이 발생한다. t = 8 일 경우에 제안한 구조는 기존 구조에 비해 약 50%의 속도 향상 효과를 나타냈다. 전체 연산 지연 시간의 경우는 기존 구조에 비해 약 10% 속도가 향상되었다.

표 1. 기존 구조와의 성능 비교

	제안한 구조	기존 구조[7]
설계 공정	0. 5 ㎛	0.25 ㎛
동작 속도	80MHz	75MHz
동작 전압	3.3V	2.5V
게이트 수	약 77,000 개	약 55,000 개
처리 지연 시간 (신드롬 출력 이 후)	33 clocks	65 clocks
전체 처리 지연 시간	289 clocks	321 clocks
데이터 처리 속도	640Mbps	600Mbps

4. 결론

제안된 구조는 $GF(2^m, m = 8)$의 유한체 구조를 사용하고 RS(n, k, t); $(37 < n \leq 255, 21 < k \leq 239, t = 8)$의 사양을 지원한다. 제안된 수정 유클리드 알고리즘 연산 블록은 시스톨릭 어레이 방식의 고속 병렬처리 구조를 갖는다.

제안된 복호기의 총 게이트 수는 약 77,000 개이고 최대 80MHz의 동작 속도를 나타내었으며 최대 640Mbps의 데이터 처리 속도를 갖는다. 이는 WATM이 요구하는 155Mbps 성능을 충분히 만족시킬 수 있다. 기존 구조에 비해 전체 복호 과정에서 약 10%의 연산 속도가 향상되었으며 수정 유클리드 알고리즘 블록의 경우는 약 50%의 속도가 향상되었다. 제안된 복호기는 WATM, WLAN(Wireless Local Area Network), HomeRF 등의 신뢰성있는 고속 멀티미디어 데이터 통신 응용 분야에 기여할 것이다.

참고 문헌

[1] H. M. Hsu and C. L. Wang, "An Area-Efficient Pipelined VLSI Architecture for Decoding of Reed-Solomon Codes Based on a Time-Domain Algorithm," *IEEE Trans. Circuits Syst. for Video Technology*, vol. 7, no. 6, Dec. 1997.

[2] J. H. Jeng and T. K. Truong, "On Decoding of Both Errors and Erasures of a Reed-Solomon Code Using an Inverse-Free Berlekamp-Massey Algorithm," *IEEE Trans. Communications*, vol. 47, pp. 1488-1494, Oct. 1999.

[3] M. A. Attia Ali, A. Abou-El-Azm, and M. F. Marie, "Error Rates for Non-Coherent Demodulation FCMA with Reed-Solomon Codes in Fading Satellite Channel," in *Proc. IEEE Vehicular Technology Conference(VTC'99)*, vol. 1, 1999, pp. 92-96.

[4] T. K. Matsushima, T. Matsushima, and S. Hirasawa, "Parallel Architecture for High-Speed Reed-Solomon Codec," in *Proc. IEEE International Telecommunications Symposium(ITS'98)*, vol 2, 1998, pp. 468-473.

[5] H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Yuen and I. S. Reed, "A VLSI Design of a Pipeline Reed-Solomon Decoder," *IEEE Trans. Comput.*, vol. C-34, pp. 393-403, May 1985.

[6] H. M. Shao and I. S. Reed, "On VLSI Design of a Pipeline Reed-Solomon Decoder Using Systolic Arrays," *IEEE Trans. Comput.*, vol. 37, pp. 1273-1279, Oct. 1988.

[7] H. H. Lee, M. L. Yu and L. Song, "VLSI Design of Reed-Solomon Decoder Architectures," in *Proc. IEEE International Symposium on Circuits and Systems(ISCAS 2000)*, Geneva, Switzerland, vol 5, May 2000, pp. 705-708.