論文99-36C-1-4

Reed-Solomon부호의 복호를 위한 수정 유클리드 알고리즘의 효율적인 반복 셀 구조

An Efficient Recursive Cell Architecture for Modified Euclidean Algorithm to Decode Reed-Solomon Code

金 右 鉉 *, 李 商 卨 **, 宋 文 圭 **

(Woo-Hyun Kim, Sang-Seol Lee, and Moon-Kyou Song)

요 약

Reed-Solomon(RS) 부호는 CD-ROM, HDTV, ATM 그리고 디지털 VCR 등 여러 분야에서 연집(burst) 오류를 정정하기 위해 적용되어왔다. RS 부호를 복호하기 위해서는 Berlekamp-Massey 알고리즘, 유클리드 알고리즘 그리고 수정 유클리드 알고리즘(MEA)이 개발되었다. 최근에는 이들 중에서도 MEA가 가장 자주 사용되었다. 본 논문은 부호의 복호에 사용되는 MEA을 위한 효율적인 반복 셀 구조를 제안한다. 제안된 구조의 두 가지 주된 특징은 다음과 같다. 첫째, MEA의 수행에 있어 기존의 방법[1] 보다 약 25% 적은 수의 클럭 사이클을 이용한다. 둘째, MEA 수행에 소비되는 클럭 사이클 수가 부호의 길이 n보다 큰 경우 MEA 셀의 개수를 줄일 수 있었으며, 수신된 워드를 위한 버퍼 요구량 또한 줄일 수 있었다. 예로써 (128, 124) RS 부호에 대한 MEA 회로가 VHDL을 통하여 기술되고 검증된다.

Abstract

Reed-Solomon(RS) codes have been employed to correct burst errors in applications such as CD-ROM, HDTV, ATM and digital VCRs. For the decoding RS codes, the Berlekamp-Massey algorithm, Euclidean algorithm and modified Euclidean algorithm(MEA) have been developed among which the MEA becomes the most popular decoding scheme. We propose an efficient recursive cell architecture suitable for the MEA. The advantages of the proposed scheme are twofold. First, the proposed architecture uses about 25% less clock cycles required in the MEA operation than [1]. Second, the number of recursive MEA cells can be reduced, when the number of clock cycles spent in the MEA operation is larger than code word length n. thereby buffer requirement for the received words can be reduced. For demonstration, the MEA circuitry for (128, 124) RS code have been described and the MEA operation is verified through VHDL.

* 學生會員, 圓光大學校 電氣工學科

(Department of Electrical Engineering, Wonkwang University)

** 正會員, 圓光大學校 電氣電子工學部

(Division of Electrical and Electronic Engineering, Wonkwang University)

※ 이 논문은 1998년도 원광대학교의 교비지원에 의해서 연구됨

接受日字:1998年8月20日, 수정완료일:1998年12月24日

Ⅰ. 서 론

Reed-Solomon(RS)부호는 연집(burst) 오류에 강한 특성을 갖는 오류 정정 부호이다. 이러한 특성으로 RS부호는 지금까지 CD-ROM, DVD, HDTV, Digital VCR, ATM, 위성 통신과 같은 여러 분야에서 FEC(Forward Error Correction)를 위하여 사용되어 왔다[3, 4, 5].

RS부호는 보통(n, k)로 표현된다. 여기서 n은 부호

의 길이를 나타내며, k는 정보의 길이를 나타낸다. 또한 정정 가능한 오류의 최대 개수 t는 $(n-k/2)$로 정의된다. RS부호의 복호에 삭제(erasure) 위치 정보를 이용하면 $(n-k)\leq(2e+s)$를 만족하는 개수의 오류와 삭제를 정정할 수 있다. 여기서 e와 s는 각각 오류와 삭제의 개수이다[3, 4].

RS 부호를 복호하는 과정에서, 오류의 위치와 값을 판별하기 위해서 보통 유클리드 알고리즘(EA)과 Berlekamp-Massey 알고리즘(BMA) 그리고 수정 유클리드 알고리즘(MEA)이 사용되었다. EA는 일반적으로 BMA에 비하여 회로의 복잡도에 있어 비효율적이지만 이해와 적용이 쉽다는 장점을 지니고 있다. 유클리드 알고리즘의 수행에 요구되는 역 연산은 회로의 복잡도를 높인다. MEA는 역 역산을 피하도록 개발되어서 BMA 만큼 회로의 복잡도에 있어 효율적이고 이해와 구현이 쉽다는 장점을 가지고 있다[1, 2, 4, 5].

최근에 RS 부호의 복호에 MEA가 자주 사용되었는데, 이 알고리즘을 이용하면 수정 신드롬 다항식(modified syndrome polynomial) $T(x)$와 삭제 위치자 다항식(erasure locator polynomial) $A(x)$로부터 오류의 위치와 값에 대한 정보를 가지고 있는 오류/삭제 위치자 다항식(error/erasure locator polynomial) $\sigma(x)$와 오류/삭제 평가자 다항식(error/erasure evaluator polynomial) $\omega(x)$를 얻을 수 있다[1, 2].

[6]에서는 full systolic array 구조로 MEA를 구현하였기 때문에 다수의 GF(Galois Field) 곱셈기가 요구된다. 더욱이 GF 곱셈기의 요구량은 $2t=(n-k)$가 클수록 많아지기 때문에 회로의 복잡도 또한 증가하게 된다. [2]는 [6]에서 사용된 MEA 셀을 반복 셀(recursive cell) 구조로 수정하였으며, $(n-k)^2>n$일 때 $\lceil (n-k)^2/n \rceil$ 개의 MEA 셀에 대하여 multiplexing 기법을 적용하여 구현한다. 여기서 $(n-k)^2$은 한 개의 MEA 셀이 MEA를 수행하는데 소비하는 클럭 사이클 수이다. multiplexing 기법을 사용할 경우 요구되는 MEA셀의 개수는 단일 MEA 수행에 소비되는 클럭 사이클 수를 줄이면 감소될 수 있다. MEA는 종결 조건을 만족할 때까지 GF 곱셈과 뺄셈을 반복하므로 MEA를 수행하는데 걸리는 시간은 이 곱셈과 뺄셈 연산 회수에 의해서 결정된다. 본 논문에는 곱셈과 뺄셈 연산 회수를 줄임으로써 MEA를 수행하는 데 요구되는 시간(클럭 사이클 수)을 줄일 수 있는 방법을 제안하고자 한다. Ⅱ장에서는 MEA를 이용한 RS복호기의 구조에 대하여 논하며, Ⅲ장에서는 MEA의 효율적인 반복적인 셀 구조의 유도 과정과 MEA 수행에 소비되는 클럭 사이클 수가 n보다 큰 경우의 RS 복호기에 그의 적용을 논하였다. 그리고 Ⅳ장을 결론으로 맺는다.

Ⅱ. MEA를 이용한 RS 복호기의 구조

그림 1은 MEA를 이용한 RS복호기의 구조를 보인다. 그림 1의 첫 번째 단에서 $r(x)$는 수신된 부호 다항식을 의미하며, Erasure Locations는 $r(x)$와 함께 동기 되어서 입력되는 2진 시퀀스로써 삭제 위치 정보이다. $r(x)$에 생성자 다항식(generator polynomial) $g(x)$의 해를 대입하면 신드롬 다항식(syndrome polynomial) $S(x)$가 얻어지며, 이와 동시에 $r(x)$와 함께 입력된 삭제 위치 정보로부터 삭제 위치자 다항식의 근 α^{-i}를 추출한다. 두 번째 단에서는 신드롬 다항식 $S(x)$와 삭제 위치자 다항식의 근을 이용하여 수정 신드롬 다항식 $T(x)$가 얻어지고, 삭제 위치자 다항식 $A(x)$는 삭제 위치자 다항식의 근을 다항식 확장하여 얻어진다. 세 번째 단은 MEA 수행부로써 $T(x)$와 $A(x)$를 입력 받아서 오류/삭제 위치자 다항식 $\sigma(x)$와 오류/삭제 평가자 다항식 $\omega(x)$를 출력한다. 네 번째 단에서는 $\sigma(x)$와 $\omega(x)$를 이용하여 오류의 위치와 값을 평가한다.[1, 2]

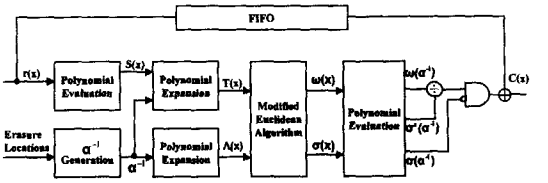


그림 1. MEA를 이용한 RS 복호기의 구조
Fig. 1. A RS decoder using MEA.

Ⅲ. MEA의 효율적인 반복 셀 구조

1. MEA의 효율적인 반복 셀 구조의 유도

MEA는 다음과 같이 세 단계로 표현될 수 있다. 이 알고리즘을 수행하기 $\deg(A(x))>\deg(T(x))$전에 이면 오류가 발생하지 않은 경우이며, $\omega(x)=T(x)$, $\sigma(x)=A(x)$로 결정하고 다음 단계들을 수행하지 않는

다. 여기서 $\deg(z(x))$는 임의의 다항식 $z(x)$의 차수(degree)를 의미한다. 하지만 $\deg(A(x)) \le \deg(T(x))$이면 다음 단계들을 반복 수행함으로써 $\omega(x)$와 $\sigma(x)$를 얻을 수 있다[1, 2].

단계 1) 식 (1)에 와 를 대입하여 초기값을 구한다.

$$R_0 = (x)^{2t},\quad Q_0(x) = T(x),\quad \lambda_0(x) = 0,\quad \mu_0(x) = A(x) \quad (1)$$

단계 2) 식 (2)~(5)를 이용하여 $R_i(x)$, $Q_i(x)$, $\lambda_i(x)$, $\mu_i(x)$를 구한다.

$$R_i(x) = [\rho_{i-1} b_{i-1} R_{i-1}(x) + \overline{\rho_{i-1}} a_{i-1} Q_{i-1}(x)] - x^{|l_{i-1}|}[\rho_{i-1} a_{i-1} Q_{i-1}(x) + \overline{\rho_{i-1}} b_{i-1} R_{i-1}(x)] \quad (2)$$

$$Q_i(x) = \rho_{i-1} Q_{i-1}(x) + \overline{\rho_{i-1}} R_{i-1}(x) \quad (3)$$

$$\lambda_i(x) + [\rho_{i-1} b_{i-1} \lambda_{i-1}(x) + \overline{\rho_{i-1}} a_{i-1} \mu_{i-1}(x)] - x^{|l_{i-1}|}[\rho_{i-1} a_{i-1} \mu_{i-1}(x) + \overline{\rho_{i-1}} b_{i-1} \lambda_{i-1}(x)] \quad (4)$$

$$\mu_i(x) = \rho_{i-1} \mu_{i-1}(x) + \overline{\rho_{i-1}} \lambda_{i-1}(x) \quad (5)$$

$$\rho_{i-1} = \begin{cases} 1, & l_{i-1} \ge 0 \\ 0, & l_{i-1} < 0 \end{cases},\ \text{where}$$

$$l_{i-1} = \deg(R_{i-1}(x)) - \deg(Q_{i-1}(x)) \quad (6)$$

여기서, a_{i-1}와 b_{i-1}는 각각 $R_{i-1}(x)$와 $Q_{i-1}(x)$의 최고차 항의 계수이다.

단계 3) 종결 조건 $\deg(\lambda_i(x)) > \deg(R_i(x))$을 만족하면 $\omega(x) = R_i(x)$, $\sigma(x) = \lambda_i(x)$이 되고, 알고리즘은 종료된다. 그렇지 않으면 단계 2로 간다.

식 (2), (4)를 고려해 보자. [1]의 방법은 $R_i(x)$, $\lambda_i(x)$를 연산하기 위해 두 개의 곱셈기와 한 개의 덧셈기를 가진 연산 셀 두 개를 사용하였다. 이 경우 MEA는 최악의 경우 식 (2), (4)를 $2t$번 반복 수행해야 하며, 식 (2), (4)를 한번 수행하기 위해서는 $R_0(x)$의 차수가 $2t$이기 때문에 상수 항을 포함해서 총 $2t+1$개의 항에 대해 계수의 곱셈과 덧셈을 행해야 했다. 즉, MEA 수행에 총 $2t(2t+1)$ 개의 클럭 사이클이 소비되었다. 그러나 식 (2), (4)가 반복 수행 되면 될 수록 $R_{i-1}(x)$와 $\lambda_{i-1}(x)$의 최고차항은 서로 소거되기 때문에 $R_i(x)$와 $\lambda_i(x)$의 차수는 감소된다. 다시 말하자면, 식 (2), (4)가 반복될수록 식 (2), (4)를 한번 수행하기 위해 곱셈과 덧셈을 연산하여야 할 계수항의 개수가 감소한다. 따라서, 식 (2), (4)는 다음 식 (2′), (4′)과 같이 다시 쓸 수 있다.

$$R_i(x) = \rho_{i-1} \sum_{k=\deg(R_{i-1}(x))}^{0} [b_{i-1} \cdot co^{(k)}(R_{i-1}(x)) - a_{i-1} \cdot co^{(k-|l_{i-1}|)}(Q_{i-1}(x))] x^k + \overline{\rho_{i-1}} \sum_{k=\deg(Q_{i-1}(x))}^{0} [a_{i-1} \cdot co^{(k)}(Q_{i-1}(x)) - b_{i-1} \cdot co^{(k-|l_{i-1}|)}(R_{i-1}(x))] x^k \quad (2')$$

$$\lambda_i(x) = \rho_{i-1} \sum_{k=\deg(R_{i-1}(x))}^{0} [b_{i-1} \cdot co^{(k)}(\lambda_{i-1}(x)) - a_{i-1} \cdot co^{(k-|l_{i-1}|)}(\mu_{i-1}(x))] x^k + \overline{\rho_{i-1}} \sum_{k=\deg(Q_{i-1}(x))}^{0} [a_{i-1} \cdot co^{(k)}(\mu_{i-1}(x)) - b_{i-1} \cdot co^{(k-|l_{i-1}|)}(\lambda_{i-1}(x))] x^k \quad (4')$$

여기서, $co^{(k)}(z(x))$는 임의의 다항식 $z(x)$의 k차 항의 계수를 의미한다.

식 (2′)와 (4′)를 통해서 $R_i(x)$와 $\lambda_i(x)$를 계산할 때, $\rho_{i-1}=1$이면 $\deg(R_{i-1}(x))+1$ 만큼의 시간이 소요되고, $\rho_{i-1}=0$이면 $\deg(Q_{i-1}(x))+1$만큼의 시간이 소요된다는 것을 알 수 있다. 따라서, 식 (2)와 (4)가 수행될 때 요구되는 시간 간격을 $\deg(R_{i-1}(x))$과 $\deg(Q_{i-1}(x))$에 따라 가변적으로 조정하면 MEA가 수행되는 데 걸리는 시간을 줄일 수 있다.

제안된 방법을 사용하여 MEA의 수행에 걸리는 시간이 최대인 최악의 경우를 고려해 보자. 최악의 경우의 $R_i(x)$, $Q_i(x)$, $\lambda_i(x)$, $\mu_i(x)$를 $R_i(x)^*$, $Q_i(x)^*$, $\lambda_i(x)^*$, $\mu_i(x)^*$라고 하자. 식 (2)와 (4)가 반복됨에 따른 이들의 차수 변화는 표 1과 같다.

표 1. 최악의 경우에 식 (2′)와 (4′)의 반복에 따른 $R_i(x)^*$, $Q_i(x)^*$, $\lambda_i(x)^*$, $\mu_i(x)^*$의 차수 변화

Table 1. A degree change of $R_i(x)^*$, $Q_i(x)^*$, $\lambda_i(x)^*$, and $\mu_i(x)^*$ while equation (2′) and (4′) are repeated in the worst case.

i	0	1	2	3	4	5	6	…
$\deg(R_i(x)^*)$	2t	2t-1	2t-2	2t-2	2t-3	2t-3	2t-4	…
$\deg(Q_i(x)^*)$	2t-1	2t-1	2t-1	2t-2	2t-2	2t-3	2t-3	…
$\deg(\lambda_i(x)^*)$	0	1	1	2	2	3	3	…
$\deg(\mu_i(x)^*)$	0	0	0	1	1	2	2	…

표 1을 정리하면 $\deg(R_i(x)^*)$, $\deg(Q_i(x)^*)$, $\deg(\lambda_i(x)^*)$, $\deg(\mu_i(x)^*)$를 다음 식 (7)~(10)과 같

이 표현할 수 있다.

$$\deg(R_i(x)^*) = 2t - \lceil (i+1)/2 \rceil, \text{ for } i = 1, 2, 3, \ldots \quad (7)$$

$$\deg(Q_i(x)^*) = 2t - \lfloor (i+1)/2 \rfloor, \text{ for } i = 1, 2, 3, \ldots \quad (8)$$

$$\deg(\lambda_i(x)^*) = \lfloor (i+1)/2 \rfloor, \text{ for } i = 1, 2, 3, \ldots \quad (9)$$

$$\deg(\mu_i(x)^*) = \lfloor (i+1)/2 \rfloor - 1, \quad \text{for} \quad i = 1, 2, 3, \ldots \quad (10)$$

식 (7)~(10)을 이용하여 종결점의 i를 구하여 보자. 이면 MEA의 수행이 종료되므로, 이 때 i의 값은 식 (11)와 같이 표현된다.

$$\begin{aligned} 2t - \lceil (i+1)/2 \rceil &< \lfloor (i+1)/2 \rfloor \\ 2t &< \lfloor (i+1)/2 \rfloor + \lceil (i+1)/2 \rceil \end{aligned} \quad (11)$$

식 (11)을 간단히 하기위해 i가 짝수인 경우와 홀수인 경우로 나누어 고려하면 식 (11)은 식 (12)와 (13)으로 표현된다.

$$2t < [(i+1)/2 + (i+1)/2]_{i=\text{odd}} \quad (12)$$

$$2t < [i/2 + i/2 + 1]_{i=\text{even}} \quad (13)$$

그리고 식 (12)와 (13)으로부터 식 (14)와 같은 결과를 얻을 수 있다.

$$2t < i+1 \quad (14)$$

식 (14)를 만족하는 i의 최소값은 $2t$이다. 즉, 종결조건이 나타날 때까지 식 (2'), (3), (4'), (5)을 $2t$번 반복하여야 한다는 것이다.

표 1과 식 (7), (8)을 이용하면, 최악의 경우에 t의 값에 따라 요구되는 총 클럭 사이클 수 T를 구할 수 있다. 식 (2')와 (4')에서 알 수 있듯이 $R_i(x)$와 $\lambda_i(x)$를 한번 계산할 때 요구되는 클럭 사이클 수는 $\deg(R_{i-1}(x))+1$과 $\deg(Q_{i-1}(x))+1$중에서 큰 값이어야 한다. 따라서 MEA를 수행하는 데 소요되는 시간(클럭 사이클 수) T는 다음 식 (15)와 같이 표현된다.

$$\begin{aligned} T &= \sum_{i=0}^{2t-1} [\max\{\deg(R_i(x)^*), \deg(Q_i(x)^*)\} + 1] \\ &= 2t + \sum_{i=0}^{2t-1} \max\{\deg(R_i(x)^*), \deg(Q_i(x)^*)\} \\ &\quad 2t + \deg(R_0(x)^*) + \sum_{o=1}^{2t-1} \deg(Q_i(x)^*) \end{aligned} \quad (15)$$

식 (15)는 표 1로부터 유도되고, 식 (7), (8)을 대입하면 식 (16)이 된다.

$$T = 2t + 2t + \sum_{i=1}^{2(t-1)} \left[2t - \lfloor \frac{i+1}{2} \rfloor \right] \quad (16)$$

편의상 식 (16)의 합산항의 범위를 $2t$까지 조정하면 식 (17)이 된다.

$$T = 4t + 2t(2t-1) - \sum_{i=1}^{2t} \left[\lfloor \frac{i+1}{2} \rfloor - \lfloor \frac{2t+1}{2} \rfloor \right] \quad (17)$$

식 (17)에서 $\lfloor (i+1)/2 \rfloor$는 같은 값이 두 번씩 반복되고, $2t$가 짝수이므로 $\lfloor (2t+1)/2 \rfloor = t$가 되어서 식 (18)이 된다.

$$\begin{aligned} T &= 2t(2t+1) - \sum_{i=1}^{t} 2j + t \\ &= 2t(2t+1) + t - 2 \cdot \frac{t(t+1)}{2} \\ &= 2t(2t+1) - t^2 \end{aligned} \quad (18)$$

[1]의 방법은 $2t(2t+l)$ 만큼의 클럭 사이클 수가 소비되므로 제안된 방법을 적용할 경우 t^2만큼의 클럭 사이클 수가 절약된다는 것을 식 (18)로부터 알 수 있다. 식 (19)에 의하여 [1]의 방법에 비하여 절약되는 클럭 사이클 수의 비율 R_{sc}는 약 25%이다.

$$R_{sc} = \frac{t^2}{2t(2t+1)} \cong \left(\frac{t^2}{4t^2} \right)_{t \gg 1} = \frac{1}{4} = 0.25 \quad (19)$$

2. (128, 124) RS 부호에 제안된 방법을 적용한 구조와 시뮬레이션 결과

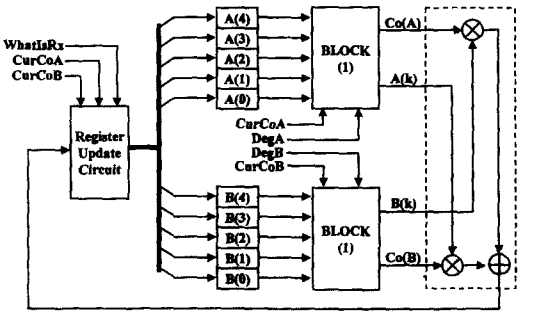


그림 2. $R_i(x)$와 $Q_i(x)$연산부
Fig. 2. Computation block for $R_i(x)$ and $Q_i(x)$.

(128, 124) RS 부호에 제안된 방법을 적용하면, 그림 2, 3, 4와 같다. 제안된 구조는 [6]에서 제안된 구조와 유사하며, BLOCK(1)과 Register Update Circuit에 제어 신호들이 더 첨가되었다는 것이 가장

큰 차이점이다. 그림 2는 $R_i(x)$와 $Q_i(x)$ 연산부를 보이며, 식 (2′)와 (3)에 해당하는 연산을 수행한다. 레지스터 A(j)와 B(j)는 각각 식 (1)의 $R_0(x)$와 $Q_i(x)$ 다항식의 계수들로 초기화 되며 연산이 반복됨에 따라 $R_i(x)$와 $Q_i(x)$의 다항식의 계수들을 가지게 된다. BLOCK(1)은 DegA의 값에 따라 A(j)레지스터의 최고차 항의 계수 Co(A)를 출력하고, CurCoA의 값에 따라 연산할 계수 A(k)를 매 클럭 마다 출력한다. WhatIsRx는 $R_i(x)$가 저장 될 위치를 나타낸다. 이 값이 0이면 레지스터 A(j)가 갱신되며, 레지스터 B(j)는 이전 값을 유지한다. WhatIsRx가 1이면 레지스터 B(j)가 갱신되고, 레지스터 A(j)는 이전 값을 유지한다. 갱신된 레지스터는 $R_i(x)$의 계수들이 되고, 이전 값을 유지한 레지스터는 $Q_i(x)$의 계수들이 된다. 이렇게 함으로써 식 (3)과 (5)에서 요구되는 스와핑을 대신하며, 스와핑에 클럭 사이클이 요구되지 않는다. DegA와 DegB를 위한 블록도는 그림 4와 같으며, 각각 레지스터 A(j)와 B(j)에 저장된 다항식 계수들의 차수를 의미한다. CurCoA는 그림 4에서 알 수 있듯이 쉬프트 레지스터로 구성되어 있으며, 각 비트는 연산 되어야 할 레지스터의 위치를 가르킨다. 예를 들어 두 번째 비트가 1이면 A(1) 레지스터가 연산된다. CurCoA는 식 (2′), (3), (4′), (5)의 i가 증가할 때마다 초기화된다. CurCoB는 CurCoA와 같은 구조를 가진다. 그림 3은 $\lambda_i(x)$, $\mu_i(x)$을 위한 블록도이며, 식 (4′)와 (5)에 해당하는 연산을 수행하며 $R_i(x)$와 $Q_i(x)$ 연산부와 유사한 구조를 갖는다. 그림 4의 차수 비교기는 종결 조건을 체크하기 위해 필요한 블록이며, 종결 조건이 만족되면 SW가 활성화되어 $\omega(x)$와 $\sigma(x)$가 출력 된다.

그림 5는 그림 2, 3, 4에 보인 회로의 시뮬레이션 결과이다. VHDL로 모델링 하였으며, Synopsys사의 VHDL debugger를 이용하여 시뮬레이션을 수행하였다. 시뮬레이션에 이용된 $T(x)$와 $A(x)$는 각각 16진수 표현법으로 (95, BA, 03, B7)과 (00, 00, 00, 00, 01)이다. 그림 5에서 MEASTART 신호가 1일 때 MEA연산이 시작되며, 이때 BX3~BX0와 DX4~DX0의 값은 각각 $T(x)$와 $A(x)$인 (95, BA, 03, B7)와 (00, 00, 00, 00, 01)로 초기화되며, AX4~AX0와 CX4~CX0는 각각 (01, 00, 00, 00, 00)과 (00, 00, 00, 00, 00, 00)으로 초기화된다. 그림 5에서 SX4~SX0와 WX3~WX0는 각각 $\sigma(x)$와 $\omega(x)$를 의미한다. EVLSTART가 1로 출력되는 시점에서 MEA 연산이 종료되며, 이 때 $\sigma(x)$와 $\omega(x)$의 값은 각각 (00, 00, CF, 13, 93)과 (00, 00, BF, 09)이다.

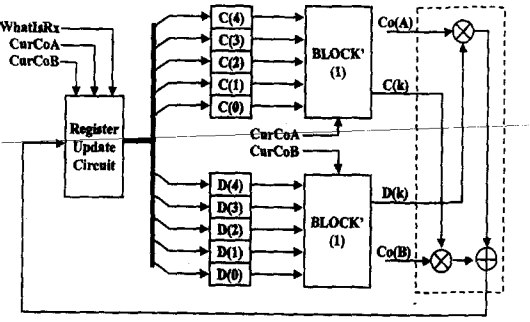


그림 3. $\lambda_i(x)$와 $\mu_i(x)$ 연산부
Fig. 3. Computation block for $\lambda_i(x)$ and $\mu_i(x)$.

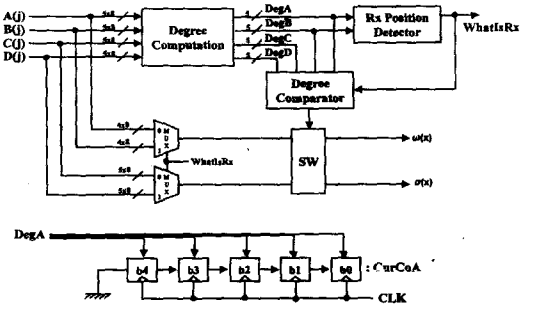


그림 4. $\sigma(x)$와 $\omega(x)$ 결정부 그리고 차수 연산부
Fig. 4. Determination block of $\sigma(x)$ and $\omega(x)$, and degree computation block.

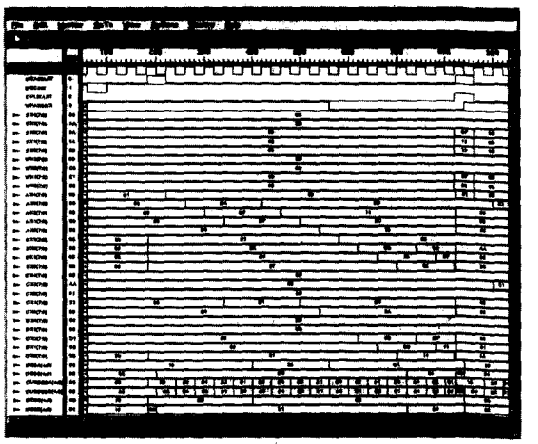


그림 5. MEA 수행부의 VHDL 시뮬레이션 결과
Fig. 5. VHDL simulation result of the MEA operation block.

3. 제안된 반복 셀 구조의 Multiplexing 기법 적용

MEA의 수행에 요구되는 클럭 사이클 수를 #(MEA)라고 하자. #(MEA)가 n 보다 큰 경우에는 여러 개의 MEA 셀을 multiplexing 하여 사용하여야 한다[2]. 이 경우에 요구되는 MEA 셀의 개수는 $\lceil \#(MEA)/n \rceil$이다. 따라서, 제안된 구조에 [2]에서 제안한 multiplexing 기법을 적용하면 $\lceil (2t(2t+1)t^2)/n \rceil$개의 셀이 요구된다. 표 2는 [2], [6] 그리고 제안한 방식의 MEA 수행에 요구되는 셀 개수 비교를 보인다. 표 2에서 알 수 있듯이 #(MEA)가 n보다 큰 경우에 제안된 구조에 multiplexing 기법을 적용하면 MEA의 수행에 [2] 보다 더 적은 개수의 MEA 셀이 요구된다.

표 2. MEA 수행에 요구되는 셀 개수의 비교

Table 2. The comparison of the number of cells required in the MEA computation.

RS 부호	Full systolic Array [6]	Multiplexing technique [2]	Proposed
(15, 9)	6	3	3
(31, 15)	16	9	7
(255, 223)	32	5	4

표 3. MEA 연산 동안 요구되는 수신 워드 지연용 버퍼의 개수 비교

Table 3. The comparison of the number of buffers for delaying received word during MEA computation.

RS 부호	Multiplexing technique [2]	Proposed
(15, 9)	15×3	15×3
(31, 15)	31×9	31×7
(255, 223)	255×5	255×4

MEA의 수행에 요구되는 MEA 셀의 개수를 #(*Mea cells*)라고 하자. MEA가 수행되는 동안에 수신된 워드를 오류 정정 부까지 지연시키는 버퍼의 개수는 $n\times$#(*MEA cells*)이다. 따라서, 제안된 방법을 이용하면 [2]의 방법 보다 n개 이상의 버퍼를 줄일 수 있다. (31, 15) RS 부호의 경우 2×31=62 개의 버퍼가, 그리고 (255, 223) RS 부호의 경우에는 255 개의 버퍼가 절약 될 수 있다. 표 3은 MEA의 연산 동안에 수신된 워드를 지연하는데 요구되는 버퍼수의 비교를 보였다.

Ⅳ. 결 론

본 논문에서 우리는 MEA의 수행에 포함되어 있는 불필요한 연산을 제거함으로써, MEA 수행에 요구되는 클럭 사이클 수를 줄이는 방법과 그 방법을 적용한 하드웨어를 제안하였다. 제안된 방법의 특징은 다음과 같다. 첫째, 기존의 방법 [1] 에 비해 t^2만큼의 클럭 사이클 수를 줄일 수 있었으며, 이는 기존의 방법에 비해 약 25%정도 감소시킨 결과임을 보였다. 둘째, MEA의 수행시간이 n보다 큰 경우 제안된 구조에 multiplexing 기법을 적용하여 기존의 방법 [2] 보다 소요되는 MEA 셀의 개수를 줄였고 수신된 워드를 지연시키는 버퍼의 개수도 줄일 수 있었다. 셋째, 차수 감산 기능을 감산기 대신 쉬프트 레지스터를 이용함으로써 복잡도를 줄였고 차수 비교기 또한 간단한 구조를 가진다. 넷째, $R_i(x)$, $Q_i(x)$, $\lambda_i(x)$, $\mu_i(x)$의 저장을 위한 레지스터의 데이터 경로를 가변적으로 둠으로써 $R_i(x)$와 $Q_i(x)$ 그리고 $\lambda_i(x)$와 $\mu_i(x)$를 서로 스와핑하는데 클럭 사이클이 요구되지 않는다.

참 고 문 헌

[1] G. S. Choi, H. S. Choi and Y. H. Kim, "RS Decoder using Modified Euclidean Algorithm for DVD/CD", *Proceedings of the ICSPAT*, vol. 1, pp. 208~212, 1997.

[2] H. M. Shao and I. 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] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall Publication, 1995.

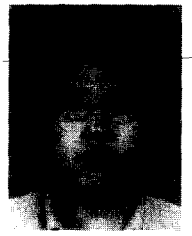
[4] M. H. Lee and S. B. Choi and J. S. Chang, "A High Speed Reed-Solomon Decoder", *IEEE Trans. Consumer Electronics*, vol. 41 no. 4, pp. 1142~1149, 1995.

[5] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applica-*

tions, IEEE Press, 1994.

[6] H. M. Shao and I. S. Reed, "A VLSI Design of a Pipeline Reed-Solomon Decoder", *IEEE Trans. Computers*, vol C-34, no. 5, pp. 393~402, 1985.

저 자 소 개



金 右 鉉(學生會員)

1997년 2월 원광대학교 전기공학과 (공학사). 1997년 3월 ~ 현재 원광대학교 대학원 전기공학과 석사 과정. 1997년 4월 ~ 11월 한국전자통신연구원 위촉연구원. 주관심분야는 VLSI 설계, 채널 부호, 마이크로프로세서, 암호화 등

李 商 卨(正會員)

1984년 2월 고려대학교 전자공학과 (공학사). 1989년 2월 한국과학기술원 전기 및 전자공학과(공학석사). 1994년 2월 한국과학기술원 전기 및 전자공학과(공학박사). 1994년 3월 ~ 현재 원광대학교 공과대학 전기전자공학부 조교수. 주관심분야는 컴퓨터 구조, 통신 및 응용 VLSI 설계, 테스팅, 영상 처리 등

宋 文 圭(正會員)

1988년 2월 고려대학교 전자공학과 (공학사). 1990년 2월 고려대학교 대학원 전자공학과(공학석사). 1994년 2월 고려대학교 대학원 전자공학과 (공학박사). 1994년 3월 ~ 현재 원광대학교 공과대학 전기전자공학부 조교수. 1997년 10월 ~ 현재 한국전자통신연구원 초빙연구원. 주관심분야는 스펙트럼확산 통신, 무선이동통신, 채널코딩, 통신 회로 VLSI설계