論文2003-40SD-6-10

새로운 DCME 알고리즘을 사용한 고속 Reed-Solomon 복호기

(High-Speed Reed-Solomon Decoder Using New Degree Computationless Modified Euclid's Algorithm)

白 在 鉉 *, 鮮 于 明 勳 *

(Baek Jae Hyun and Sunwoo Myung Hoon)

요 약

본 논문에서는 차수 연산이 필요 없는 새로운 DCME 알고리즘 (Degree Computationless Modified Euclid's algorithm)을 사용한 저비용 고속 RS (Reed-Solomon) 복호기를 제안한다. 제안하는 구조는 차수 연산 및 비교 회로가 필요 없어 기존 수정 유클리드 구조들에 비해 매우 낮은 하드웨어 복잡도를 갖는다. 시스톨릭 에레이 (systolic array)를 이용한 제안하는 구조는 키 방정식 (key equation) 연산을 위해서 초기 지연 없이 $2t$ 클록 사이클만을 필요로 한다. 또한, $3t+2$개의 기본 셀 (basic cell)을 사용하는 DCME 구조는 오직 하나의 PE (processing element)를 사용하므로 규칙성 (regularity) 및 비례성 (scalability)을 갖는다. 0.25㎛ Faraday 라이브러리를 사용하여 논리합성을 수행한 RS 복호기는 200MHz의 동작 주파수 및 1.6Gbps의 데이터 처리 속도를 갖는다. (255, 239, 8) RS 코드 복호를 수행하는 DCME 구조와 전체 RS 복호기의 게이트 수는 각각 21,760개와 42,213개이다. 제안하는 RS 복호기는 기존 RS 복호기들에 비해 23%의 게이트 수 절감 및 전체 지연 시간의 10%가 향상되었다.

Abstract

This paper proposes a novel low-cost and high-speed Reed-Solomon (RS) decoder based on a new degree computationless modified Euclid's (DCME) algorithm. This architecture has quite low hardware complexity compared with conventional modified Euclid's (ME) architectures, since it can remove completely the degree computation and comparison circuits. The architecture employing a systolic array requires only the latency of $2t$ clock cycles to solve the key equation without initial latency. In addition, the DCME architecture using $3t+2$ basic cells has regularity and scalability since it uses only one processing element. The RS decoder has been synthesized using the 0.25㎛ Faraday CMOS standard cell library and operates at 200MHz and its data rate supports up to 1.6Gbps. For the (255, 239, 8) RS code, the gate counts of the DCME architecture and the whole RS decoder excluding FIFO memory are only 21,760 and 42,213, respectively. The proposed RS decoder can reduce the total gate count at least 23% and the total latency at least 10% compared with conventional ME architectures.

Keywords : RS 부호, 오류 정정 코드, 차수 연산 회로, 시스톨릭 어레이, VLSI 설계

* 正會員, 亞洲大學校 電子工學部

(School of Electrical and Computer Eng., Ajou Univ.)

※ 연구는 과학기술부의 국가지정연구실(NRL) 사업과 반도체설계교육센터(IDEC)의 지원을 받아 이루어졌음.

接受日字:2003年4月14日, 수정완료일:2003年6月9日

Ⅰ. 서 론

전송한 데이터의 오류 정정을 위하여 통신 시스템에서 폭넓게 사용되는 오류 정정 부호는 컨볼루션(convolutional) 부호와 블록(block) 부호로 구분된다.[1] 랜덤

오류뿐만 아니라 연집 오류에 우수한 정정 능력을 보이는 RS 부호는 PLC(Power Line Communication)[2], DVB-T(Digital Video Broadcasting Terrestrial)[3], VSB(Vestigial Sideband)[4], 케이블 모뎀(cable modem)[5], 위성 및 이동 통신[6], 자기 저장 장치[7] 등 다양한 통신 시스템 및 디지털 저장 장치에 사용되고 있다.

일반적인 (n, k, t) RS 부호에서 k는 m 비트 정보 심벌의 개수를 나타내며 RS 부호화 후 $n=2^m-1$개의 심벌을 갖는다. $t(=\lfloor (n-k)/2 \rfloor)$는 RS 부호의 오류 정정 능력을 나타낸다[8]. RS 복호기는 신드롬(syndrome) 연산, 키 방정식 연산, Chien search 알고리즘, Forney's 알고리즘, 오류 정정의 5개 블록으로 구성된다. 이들 블록 중 오류 위치 다항식(error locator polynomial) 및 오류 크기 다항식(error value polynomial) 연산을 위한 키 방정식 블록은 가장 큰 하드웨어 복잡도 및 연산 시간을 필요로 한다. 베르캠프-메세이(Berlekamp-Massey) 알고리즘[1, 9-12], 유클리드(Euclid) 알고리즘[13, 14], 수정 유클리드(modified Euclid) 알고리즘[15-21]이 키 방정식 연산을 위해 사용된다.

베르캠프-메세이 알고리즘을 사용한 키 방정식 연산은 주파수 영역과 시간 영역에서 수행 가능하다[11]. 주파수 영역 알고리즘은 낮은 하드웨어 복잡도를 필요로 하는 시스템에 적합하지만, 하드웨어 구조가 불규칙적이며 t에 따라 다른 임계경로(critical path)를 갖는다[12]. 시간 영역 알고리즘은 기본 연산을 반복적으로 수행하므로 구현이 쉽고 하드웨어 구조가 규칙적이다[11]. 그러나 시간 영역 알고리즘은 주파수 영역 알고리즘에 비해서 더 많은 유한체(Galois filed) 곱셈기를 사용하므로 높은 하드웨어 복잡도를 갖는다. RiBM(Reformulated inversionless Berlekamp-Massey) 구조는 베르캠프-메세이 알고리즘을 사용하는 RS 복호기[1, 9-11]에 비해 임계 경로가 짧고 하드웨어 복잡도가 낮은 우수한 구조이다.

유클리드 알고리즘은 역수(inverse element) 연산을 위한 LUT(Look-Up Table) 또는 특정 회로를 필요로 한다. 그러나 수정 유클리드 알고리즘은 역수 연산을 수행하지 않으므로 연산 시간 및 하드웨어 복잡도를 줄일 수 있다. 수정 유클리드 알고리즘은 t에 관계없이 동일한 임계 경로를 가지므로 빠른 RS 복호 속도를 필요로 하는 응용 분야에 적합하며 하드웨어 구현이 쉽다. 그러나 수정 유클리드 알고리즘은 베르캠프-메세이 알고리즘에 비해 높은 하드웨어 복잡도를 갖는다.

본 논문에서는 차수 연산 및 비교 회로가 필요 없는 DCME 알고리즘을 사용한 RS 복호기를 제안한다. 제안하는 복호기는 $3t+2$개의 기본 셀만을 사용한다. 구현한 DCME 구조는 하나의 PE만을 사용하므로 VLSI 구현에 적합하며, t 또는 유한체의 변화에 따라 RS 복호기의 재설계가 용이하다. DCME 구조와 전체 RS 복호기의 게이트 수는 각각 21,760개와 42,213개이다. 제안하는 RS 복호기는 기존 수정 유클리드 알고리즘을 사용하는 RS 복호기[15-21]에 비해 23% 이상 하드웨어 복잡도를 감소시킨다.

또한, DCME 구조는 키 방정식 연산을 위해 초기 지연 시간 없이 $2t$ 사이클만을 필요로 한다. 반면에 수정 유클리드 알고리즘을 사용하는 기존 구조는 DCME 구조에 비해 더 많은 지연 시간을 필요로 한다. 제안하는 구조의 전체 RS 복호 시간은 adaptive RS 복호기[15]와 기존 RS 복호기[17]에 비해 각각 65%, 10% 향상되었다. 따라서 고속 RS 복호가 가능한 제안하는 RS 복호기는 DVB-T, VSB, DVD 등에 사용할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 DCME 알고리즘 및 기존 수정 유클리드 알고리즘에 대해 설명한다. 3장에서는 구현한 RS 복호기의 구조에 대해 기술하고, 4장에서 제안한 RS 복호기와 기존 복호기와의 성능 평가를 수행한다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 차수 연산이 필요 없는 DCME 알고리즘

본 절에서는 기존 수정 유클리드 알고리즘[19]과 제안하는 DCME 알고리즘의 연산 과정을 설명한다.

1. 수정 유클리드 알고리즘

식 (1)과 식 (2)는 키 방정식 연산을 위한 초기 조건(initial condition)을 나타낸다.

$$R_0(x) = x^{2t}Q_0 \quad (x) = S(x) = \sum_{i=0}^{2t-1} S_i x^i$$

$$(1)\lambda_0(x) = 0 \qquad \mu_0(x) = 1 \qquad (2)$$

식 (1)과 식 (2)의 $R(x)$와 $\lambda(x)$는 각각 오류 크기 다항식과 오류 위치 다항식을 의미하고, $S(x)$는 신드롬 다항식 (syndrome polynomial)이다.

수정 유클리드 알고리즘은 식 (3)~식 (6)의 다항식

연산을 반복적으로 수행한다.

$$R_{i+1}(x) = [\sigma_i b_i R_i(x) + \bar{\sigma}_i a_i Q_i(x)] - x^{l_i}[\sigma_i a_i Q_i(x) + \bar{\sigma}_i b_i R_i(x)] \quad (3)$$

$$\lambda_{i+1}(x) = [\sigma_i b_i \lambda_i(x) + \bar{\sigma}_i a_i \mu_i(x)] - x^{l_i}[\sigma_i a_i \mu_i(x) + \bar{\sigma}_i b_i \lambda_i(x)] \quad (4)$$

$$Q_{i+1}(x) = \sigma_i Q_i(x) + \bar{\sigma}_i R_i(x) \quad (5)$$

$$\mu_{i+1}(x) = \sigma_i \mu_i(x) + \bar{\sigma}_i \lambda_i(x) \quad (6)$$

a_i와 b_i는 각각 $R_i(x)$와 $Q_i(x)$의 최고차항 계수이고, i는 다항식 연산의 반복 횟수를 나타낸다. 식 (3)~식 (6)의 σ_i와 l_i는 식 (7)에 의해 주어진다.

$$l_i = deg(R_i(x)) - deg(Q_i(x)) \quad (7)$$
$$\sigma_i = 1 \text{ if } l_i \geq 0$$
$$\sigma_i = 0 \text{ otherwise}$$

식 (7)의 $deg(R_i(x))$와 $deg(Q_i(x))$는 다항식 $R_i(x)$와 $Q_i(x)$의 차수를 나타낸다. 식 (3)~식 (6)의 다항식 연산은 $deg(R_i(x)) < t$를 만족할 때까지 반복적으로 수행된다. 연산이 종료되면 다항식 $R(x)$와 $\lambda(x)$를 얻는다.

<그림 1>은 수정 유클리드 알고리즘[19]의 연산 흐름을 나타낸다. $R_i(x)$, $Q_i(x)$, $\lambda_i(x)$, $\mu_i(x)$, $deg(R_i(x))$, $deg(Q_i(x))$을 입력받아 키 방정식 연산을 수행한다. <그림 1>에서 보인 것처럼 수정 유클리드 알고리즘[19]은 다항식 $R_i(x)$, $Q_i(x)$의 차수와 t의 비교 연산이 필요하다. 따라서 각 기본 셀은 다항식의 차수 비교를 위해 차수 연산 및 비교 회로를 포함한다. 그 결과 수정 유클리드 알고리즘[19]은 키 방정식 연산을 위해서 많은 지연 시간을 필요로 하며, 다양한 다항식 연산을 위한 복잡한 제어 과정 및 하드웨어를 필요로 한다.

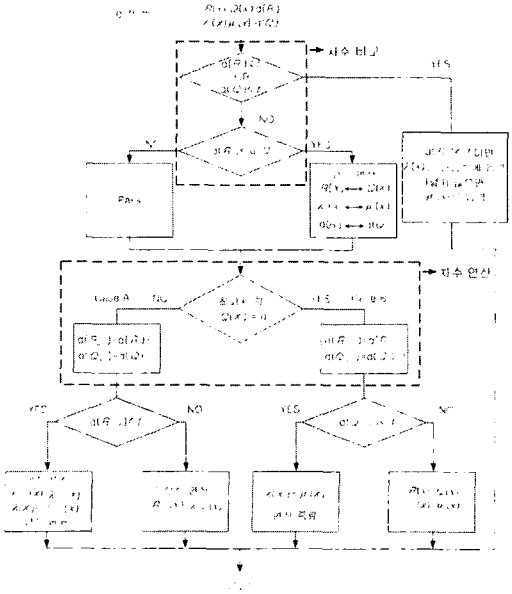

그림 1. 수정 유클리드 알고리즘의 연산 흐름
Fig. 1. Flowchart of the modified Euclid's algorithm.

2. DCME 알고리즘

제안하는 DCME 알고리즘[24]은 식 (1), (2)와 다른 초기 조건을 사용한다. $deg(R_0(x))$와 $deg(Q_0(x))$가 같도록 만들기 위해 $Q_0(x)$와 $\mu_0(x)$의 차수를 1씩 증가시킨다. 새로운 초기 조건은 식 (8), (9)와 같다.

$$R_0(x) = x^{2t} \quad Q_0(x) = xS(x) = \sum_{i=0}^{2t-1} S_i x^{i+1} \quad (8)$$

$$\lambda_0(x) = 0 \quad \mu_0(x) = x \quad (9)$$

위의 초기 조건에 의해 $deg(R_0(x))$와 $deg(Q_0(x))$는 $2t$로 같다. $deg(R_0(x))$와 $deg(Q_0(x))$가 같으므로 식 (7)의 l_0는 0이고 식 (3)은 식 (7)의 σ_0 값에 관계없이 $R_1(x) = b_0 R_0(x) + a_0 Q_0(x)$로 간단하게 표현할 수 있다. 즉, $l_i = 0$이고 $\sigma_i = 1$이면 식 (3)의 x^{l_i}가 1이므로 식 (3)은 $R_{i+1}(x) = b_i R_i(x) + a_i Q_i(x)$으로 표현할 수 있다. 또한, $l_i = 0$이고 $\sigma_i = 0$이면 식 (3)의 x^{l_i}는 1이므로 식 (3)은 $R_{i+1}(x) = a_i Q_i(x) + b_i R_i(x)$로 표현된다. 따라서 $l_i = 0$이면 σ_i 값에 관계없이 식 (3)과 식 (4)는 식 (10), (11)과 같이 표현 가능하다.

$$R_{i+1}(x) = b_i R_i(x) + a_i Q_i(x) \quad (10)$$

$$\lambda_{i+1}(x) = b_i \lambda_i(x) + a_i \mu_i(x) \quad (11)$$

식 (5)와 식 (6)의 교환 연산 역시 식 (12), (13)로 표현할 수 있다.

$$Q_{i+1}(x) = R_i(x) \quad (12)$$

$$\mu_{i+1}(x) = \lambda_i(x) \quad (13)$$

<그림 2>는 제안하는 DCME 알고리즘의 연산 흐름을 나타낸다. i, k, j는 각각 연산 반복 횟수, 상위 셀

의 위치, 하위 셀의 위치를 나타낸다. DCME 알고리즘은 시프트 연산(shift operation)에 의해 항상 $l_i = 0$을 갖는다. 다항식의 최고차항 계수가 0이면 시프트 연산은 다항식에 x를 곱하여 다항식의 차수를 1 증가시킨다. 시프트 연산은 최고차항의 계수가 0이 아닌 값을 갖질 때까지 반복 수행된다. 따라서 DCME 알고리즘은 식 (7)을 사용할 필요가 없으며 다항식 차수 연산 및 비교 회로를 제거할 수 있다. 즉, DCME 알고리즘은 식 (10)~식 (13)의 다항식 연산 및 교환 연산만을 반복 수행한다. 식 (10)~식 (13)은 다항식 차수 연산 및 비교 연산을 필요로 하지 않으므로 식 (3)~식 (6)에 비해 훨씬 단순하다. 결과적으로 DCME 알고리즘의 키 방정식 연산 블록은 기존 수정 유클리드 구조[15-21]에 비해 짧은 임계 경로 및 낮은 하드웨어 복잡도를 갖는다.

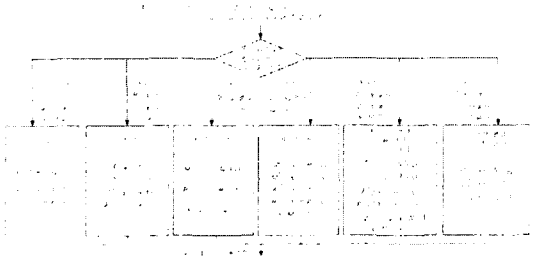

그림 2. DCME 알고리즘의 연산 흐름
Fig. 2. Flowchart of the proposed DCME algorithm.

*Control_Q(C_Q)*는 현재 수행중인 다항식 연산의 상태를 나타낸다. $C_Q = 0$이면 다항식 연산과 식 (12), (13)의 교환 연산이 함께 수행된다. 반면에 $C_Q \neq 0$이면 교환 연산 없이 다항식 연산만 수행된다. 즉, C_Q의 값에 따라 DCME 알고리즘은 교환 연산의 수행여부를 결정한다. *Control_R(C_R)*은 C_Q와 함께 다항식 연산을 제어한다. DCME 알고리즘은 $deg(Q_i(x))$가 $deg(R_i(x))$보다 크면 C_R을 1 증가시킨다. C_Q와 C_R의 초기 값은 각각 2와 0이다. $2t$ 번의 반복 연산 후 DCME 알고리즘은 종료되고 오류 크기 다항식 $R(x)$와 오류 위치 다항식 $\lambda(x)$를 얻는다.

<표 1>은 (7, 5, 1) RS 부호에서 제안하는 DCME 알고리즘의 키 방정식 연산 과정을 보여준다. DCME 알고리즘은 식 (1), (2) 대신 식 (8), (9)의 새로운 초기 조건을 사용하므로 $deg(R_0(x))$과 $deg(Q_0(x))$는 2로 같다. 따라서 $i=0$에서 DCME 알고리즘은 $Q_0(x) = \alpha^3 x^2 + \alpha^2 x$, $\mu_0(x) = x$를 초기 조건으로 사용한다. $i=1$에서 $R_0(x)$와 $Q_0(x)$의 최고차항 계수 a_0와 b_0는 각각 1과 α^3이고 C_Q가 2이므로 Case 6을 수행한다. 다항식 및 시프트 연산 후 $R_1(x) = \alpha^2 x^2$, $Q_1(x) = \alpha^3 x^2 + \alpha^2 x$, $\lambda_1(x) = x^2$, $\mu_1(x) = x$을 얻고, $deg(R_1(x))$과 $deg(Q_1(x))$는 2이고 C_Q는 1이다. $i=2$에서 $R_1(x)$와 $Q_1(x)$의 최고차항 계수 a_1와 b_1가 각각 α^2과 α^3이고 C_Q는 1이므로 Case 6을 다시 수행한다. 따라서 $R_2(x) = \alpha^4 x^2$, $Q_2(x) = \alpha^3 x^2 + \alpha^2 x$, $\lambda_2(x) = \alpha^3 x^2 + \alpha^2 x$, $\mu_2(x) = x$을 얻고 $2t=2$이므로 DCME 알고리즘은 종료된다.

표 1. (7, 5, 1) RS 코드의 DCME 알고리즘 예
Table 1. Example of the DCME algorithm for the (7, 5, 1) RS code.

i	C_Q	$R_i(x)$	$Q_i(x)$	$\lambda_i(x)$	$\mu_i(x)$	a_{i-1}	b_{i-1}
0	2	x^2	$\alpha^3 x^2 + \alpha^2 x$	0	x	-	
1	1	$(\alpha^3 x^2 + 1 \cdot (\alpha^3 x^2 + \alpha^2 x)$ $\alpha^2 x$ 시프트 연산 : αx^2	$\alpha^3 x^2 + \alpha^2 x$	$0 + x = x$ 시프트 연산 : x^2	x	1	α^3
2	0	$[\alpha^3 \alpha^2 x^2] - [\alpha^2 \cdot (\alpha^3 x^2 + \alpha^2 x)]$ $\alpha^3 x^2$ 시프트 연산 : $\alpha^3 x^3 = x^3(\alpha^3)$ 연산 종료($2t=2$)	$\alpha^3 x^2 + \alpha^2 x$	$[\alpha^3 x^2] \cdot [\alpha^2 x]$ $\alpha^5 x^3 + \alpha^4 x$ 시프트 연산 : $\alpha^5 x^4 + \alpha^4 x^3$ $= x^3(\alpha^5 x + \alpha^4)$	x	α^2	α^3

수정 유클리드 알고리즘[19]은 DCME 알고리즘과 같은 반복 횟수를 갖지만 다항식의 차수 및 비교 연산으로 인하여 각 다항식 연산 시 더 긴 지연시간 및 임계 경로를 필요로 한다. 반면에 제안하는 DCME 알고리즘은 <그림 2>와 같이 6가지 Case들 중 오직 하나만 수행하며 다항식의 차수 연산 및 비교 연산을 제거하였다. 그 결과 DCME 알고리즘은 수정 유클리드 알고리즘에 비해 짧은 임계 경로를 가지므로 각 다항식 연산을 1 클록 사이클 안에 수행할 수 있어 키 방정식 연산을 위해 2t 사이클만을 필요로 한다. 따라서 DCME 알고리즘을 사용하는 키 방정식 블록은 기존 수정 유클리드 블록[15-21]에 비해 단순하며 저비용 고속을 요하는 응용 분야에 사용 가능하다.

DCME 알고리즘 검증을 위하여 C++를 사용하여 시뮬레이션을 수행하였다. (7, 5, 1), (37, 33, 2), (41, 21, 10), (255, 235, 10), (255, 239, 8), (255, 251. 2) 등의 다

양한 (n, k, t) RS 부호에 대해서 시뮬레이션을 수행하였으며 DCME 알고리즘의 결과와 기존 수정 유클리드 알고리즘의 결과가 동일함을 확인하였다.

Ⅲ. 제안하는 RS 복호기의 구조

본 절에서는 제안하는 RS 복호기에 대해서 기술하고 구현한 신드롬 연산 블록, DCME 블록, Chien Search 블록, Forney's 블록을 분석한다.

1. 전체 RS 구조

복호기에 입력되는 수신 다항식(received polynomial) $r(x)$는 식 (14)와 같다.

$$r(x) = c(x) + e(x) \tag{14}$$

$c(x)$와 $e(x)$는 각각 송신 다항식(transmitted polynomial) 및 오류 다항식(error polynomial)이다. $c(x)$는 정보 다항식(message polynomial)과 생성 다항식에 의해서 식 (15)와 같이 표현된다.

$$\begin{aligned} c(x) &= x^{n-k} m(x) + [x^{n-k} m(x) \bmod g(x)] \\ &= q(x) g(x) \end{aligned} \tag{15}$$

$q(x)$는 몫 다항식(quotient polynomial)이다. RS 복호기는 수신 다항식 $r(x)$로부터 $e(x)$을 찾고 오류를 정정한다.

<그림 3>은 제안하는 RS 복호기의 구조이다. 신드롬 연산 블록은 $g(x)$의 근을 사용하여 신드롬 다항식 $S(x)$을 구한다. 키 방정식 연산 블록은 $R(x)$와 $\lambda(x)$을 얻기 위해 다항식 연산을 수행한다. Chien Search 블록 및 Forney's 블록은 $R(x)$와 $\lambda(x)$을 사용하여 오류 위치 및 크기를 연산한다. FIFO 메모리는 RS 복호 시간동안 수신 다항식을 지연시킨다. RS 복호기는 FIFO 메모리의 출력 값에 오류를 더하여 수신된 오류를 정정한다.

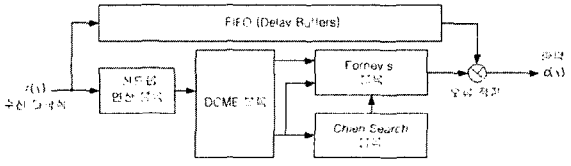

그림 3. 제안하는 RS 복호기
Fig. 3. Proposed RS decoder.

2. 신드롬 연산 블록

식 (16)은 신드롬 다항식 계수의 정의를 나타낸 것이다[23].

$$\begin{aligned} &S_{i-1} = r(\alpha^i) = c(\alpha^i) + e(\alpha^i) = e(\alpha^i), \\ &i = 1, 2, \cdots, 2t \end{aligned} \tag{16}$$

$c(x)$는 식 (15)와 같이 $g(x)$와 $q(x)$의 곱으로 표현할 수 있으므로 x에 생성 다항식의 근을 대입하면 $c(x)$는 0이다. 따라서 $r(x)$와 $e(x)$는 같다. 계산된 신드롬 다항식 계수를 사용하여 신드롬 다항식은 식 (17)과 같이 정의한다.

$$S(x) = \sum_{i=0}^{2t-1} S_i x^i \tag{17}$$

수신 다항식에 오류가 포함되지 않았을 경우 $S(x)$는 0이다.

<그림 4>는 신드롬 연산 블록을 나타낸다. $S(x)$를 연산하기 위해 생성 다항식의 근 $\alpha^i (i = 1, 2, \cdots, 2t)$를 사용한다. 계산된 $S(x)$는 수신 다항식 $r(x)$의 오류 형태를 나타내며 키 방정식 연산을 위해 사용된다. 신드롬 연산 블록의 전체 셀 개수는 오류 정정 능력 t에 따라 바뀌며 t의 두 배이다. 구현한 신드롬 연산 블록의 게이트 수는 6,460개이다.

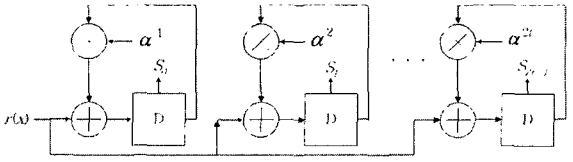

그림 4. 신드롬 연산 블록
Fig. 4. Syndrome computation block.

3. 제안하는 DCME 구조

<그림 5>는 $3t+2$개의 기본 셀과 제어 블록으로 구성된 DCME 블록의 구조를 나타낸다. $R(x)$와 $Q(x)$의 반복 연산을 수행하는 상위 셀의 개수는 $2t+1$개고, $\lambda(x)$와 $\mu(x)$의 반복 연산을 수행하는 하위 셀의 개수는 $t+1$개이다. 상위 셀과 하위 셀의 구조는 동일하다. 따라서 DCME 블록은 단순히 $3t+2$개의 기본 셀만으로 구성되며 같은 구조의 기본 셀을 사용하므로 규칙성과 비례성을 갖는다. 각 기본 셀은 $R(x)$, $Q(x)$, $\lambda(x)$,

$\mu(x)$의 계수를 저장하고 있으며 각 셀의 위치는 다항식의 차수를 의미한다.

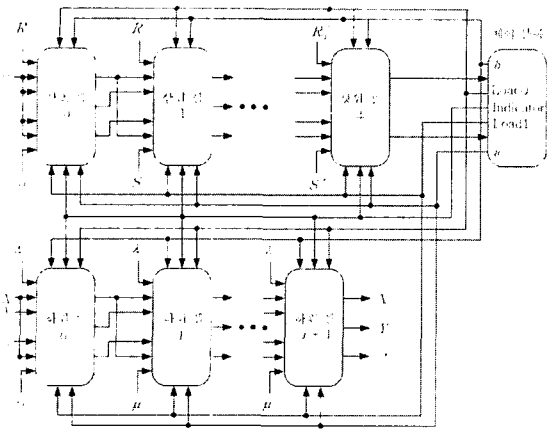

그림 5. 제안하는 DCME 구조
Fig. 5. Proposed DCME architecture.

<그림 6>은 제안하는 기본 셀의 구조이다. 괄호 밖의 문자는 상위 셀의 입출력이며 괄호 안의 문자는 하위 셀의 입출력을 나타낸다. 제안하는 기본 셀은 차수 연산 및 비교 회로를 포함하지 않으므로 두개의 레지스터, 6개의 2-입력 멀티플렉서, 2개의 유한체 곱셈기, 하나의 유한체 덧셈기로 구성된다.

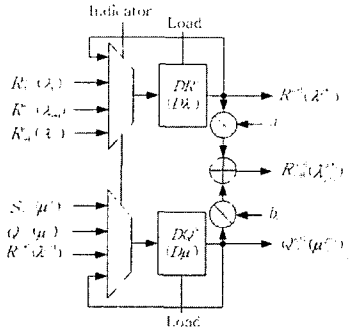

그림 6. 제안하는 기본 셀
Fig. 6. Proposed basic cell.

<그림 5>의 제어 블록은 각 기본 셀의 제어 신호를 출력한다. 제어 신호에는 *Load0, Load1, Indicator*가 있다. *Load0*와 *Load1*은 기본 셀의 레지스터에 사용되며 *Indicator*는 다항식 연산을 위한 신호이다. 제어 블록은 $R_i(x)$와 $Q_i(x)$의 최고차항 계수가 0인지 아닌지를 검색하여 *Indicator* 신호를 출력한다. *Indicator* 신호에 따라 각 기본 셀은 초기 값을 로드, 시프트 연산, 다항식 계수의 유지, 다항식 연산 중 하나의 동작을 수행한다. 시프트 연산은 $R_i(x)$의 최고차항 계수 a_i와 $Q_i(x)$의 최고차항 계수 b_i에 따라 수행된다. $a_i=0$, $b_i\neq 0$이면 상위 $k+1$ 셀의 DR^{k+1} 레지스터는 k 셀의 DR^k 레지스터로부터 R_i^k의 계수 값을 입력받으며 DQ 레지스터의 값은 변화가 없다. k는 0부터 $2t-1$의 값을 갖는다. $a_i\neq 0$, $b_i=0$이면 DR 레지스터의 값은 변화가 없으며 DQ^{k+1} 레지스터는 DQ^k 레지스터로부터 Q_i^k를 입력받는다. 또한, $a_i=0$, $b_i=0$이면 DR^k 레지스터와 Q 레지스터의 계수는 각각 DR^{k+1} 레지스터와 DQ^{k+1} 레지스터로 입력된다. 반면에 $a_i\neq 0$, $b_i\neq 0$이면 각 기본 셀은 <표 2>에서 보인 것처럼 다항식 연산 수행 후 시프트 연산을 수행한다.

결과적으로 상위 $2t$ 셀은 다항식 연산의 반복 후 항상 $R(x)$와 $Q(x)$의 최고차항 계수를 갖는다. 같은 연산이 $\lambda(x)$와 $\mu(x)$에 적용되고 연산 종료 후 하위 t 셀 역시 최고차항의 계수를 갖는다. DCME 블록에 사용되는 모든 멀티플렉서는 동일한 제어 신호를 받으며 각 기본 셀은 각 사이클마다 동일한 연산을 수행한다. 따라서 오직 $2t$ 사이클의 지연 시간 후 상위 $t+j$ 셀은 오류 크기 다항식 $R(x)$의 계수 R_j를 갖으며, 하위 셀 j는 오류 위치 다항식 $\lambda(x)$의 계수 λ_j를 갖는다. j는 0부터 t 사이의 값을 갖는다. 제안하는 DCME 블록의 게이트 수는 21,760개이다.

4. Chien Search 및 Forney's 블록

오류 크기는 식 (18)의 Forney's 알고리즘을 사용하여 계산할 수 있다.

$$e_{ik}=\frac{-\Omega(X_k^{-1})}{\Lambda'_k(X_k^{-1})} \tag{18}$$

$\Omega(x)$는 오류 크기 다항식이고 $\Lambda'(x)$는 오류 위치 다항식의 미분 다항식이다. X_k^{-1}는 오류 위치 다항식 $\lambda(x)$의 근이며, 근의 역수 X_k는 오류의 위치를 나타낸다.

(255, 239, 8) RS 부호에서 $\lambda(x)$의 근은 식 (19)를 사용하여 구할 수 있다.

$$\lambda(\alpha^i)=\lambda_8(\alpha^i)^8+\lambda_7(\alpha^i)^7+\cdots+\lambda_1(\alpha^i)+\lambda_0 \tag{19}$$

$\lambda(x)$의 계수는 <그림 5>의 하위 셀로부터 얻는다.

식 (19)를 식 (20)과 같이 재배열하면 하드웨어 구현이 용이하다[19]. 따라서 오류 위치 다항식의 근을 구하기 위한 Chien search 알고리즘은 식 (20)을 사용하여 구현한다.

$$\lambda(\alpha^i) = \alpha^i(\alpha^i(\cdots(\alpha^i(\lambda_8\alpha^i + \lambda_7) + \lambda_6)\cdots) + \lambda_1) + \lambda_0 \tag{20}$$

<그림 7>은 $\alpha^i\ (i=0,1,\cdots,n-2,n-1)$을 입력받아 오류 위치를 연산하는 Chien Search 블록의 구조를 나타내며, S'가 0일 때의 출력 X_k^{-1}가 $\lambda(x)$의 근이다.

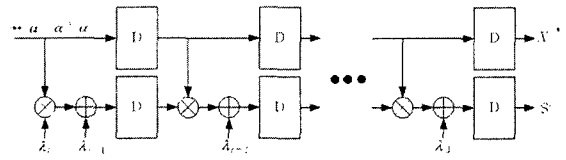

그림 7. Chien search 블록
Fig. 7. Chien search block.

<그림 8>은 오류 크기를 연산하는 Forney's 블록을 나타낸다. 역수(inverse) ROM 테이블은 식 (18) $\Lambda'(X_k^{-1})$의 역수 연산을 위해 사용된다. Forney's 알고리즘을 사용하는 기존 RS 복호기[17, 19] 역시 역수 ROM 테이블을 사용한다. Chien search 및 Forney's 블록은 t 사이클의 지연 시간을 갖으며, 게이트 수는 각각 4,530개와 7,980개이다.

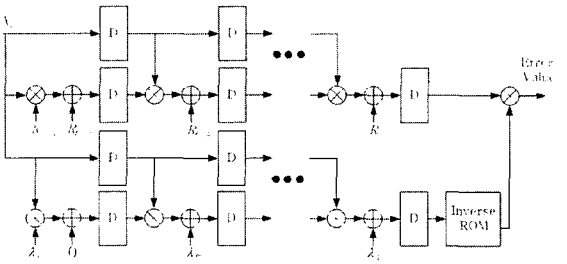

그림 8. Forney's 블록
Fig. 8. Forney's block.

Ⅳ. 성능 평가

제안하는 DCME 복호기는 VHDL을 사용하여 설계하였으며 Faraday 0.25㎛ 라이브러리와 SYNOPSYS™을 사용하여 논리 합성을 수행하였다. 또한 CADENCE™ Verilog-XL을 사용하여 VHDL 코드의 시뮬레이션을 수행하였다. 구현한 RS 복호기는 200MHz의 동작 속도와 42,213개의 게이트를 갖으며 전체 RS 복호 지연 시간은 288 클록 사이클이다. <표 2>는 (255, 239, 8) RS 부호에서 기존 수정 유클리드 RS 복호기[15, 17]와 DCME 복호기와의 성능 비교를 나타낸다.

표 2. (255, 239, 8) RS 부호 상의 성능 비교
Table 2. Performance comparison for the (255, 239, 8) RS code.

		[15]	[16]	[17]	DCME 복호기
설계 공정		-	-	0.25㎛	0.25㎛
동작 주파수		30MHz	-	75MHz	200MHz
수정 유클리드 블록	레지스터	80	$8t+4$	$50t$	$6t+4$
	멀티플렉서	-	$16t+8$	$38t$	$18t+12$
	곱셈기	4	$8t+4$	$8t$	$6t+4$
	덧셈기	4	$4t+2$	$4t$	$3t+2$
수정 유클리드 블록 게이트 수		-	-	43,100개	21,760개
전체 게이트 수		211,296개	-	55,200개	42,213개
수정 유클리드 블록 지연 시간		$(4t^2+t+20)pr$	$2t+1$	$3t+37$	$2t$
전체 지연 시간		835 클록	-	321 클록	288 클록

두 개의 기본 셀을 사용하는 adaptive RS 복호기[15]는 80개의 레지스터, 4개의 유한체 곱셈기, 4개의 유한체 덧셈기, 제어 회로로 구성된다. 두 개의 기본 셀은 수정 유클리드 알고리즘을 반복 수행한다. adaptive 복호기는 기존 RS 복호기와 같이 다항식 차수 연산 및 비교 회로를 사용하고 두 개의 신드롬 연산 블록을 가지므로 매우 높은 하드웨어 복잡도를 갖는다. adaptive RS 복호기의 게이트 수는 FIFO 메모리를 제외하고 211,296개이다[15].

EPE(embedded polynomial expansion) 복호기[16]는 XEA(extended Euclid's algorithm) 알고리즘을 사용한다. XEA 알고리즘은 수정 유클리드 알고리즘에 비해 단순하지만 여전히 다항식 차수 연산 및 비교 회로가 필요하고 제어 회로가 복잡하다. 또한, EPE 구조는 전체 기본 셀 중 사용하는 셀의 개수를 나타내는 셀 이용률(utilization)이 높다. 그러나 제안하는 DCME 구조는 사용하는 셀 개수는 동일하고 $3t+2$개의 기본 셀만으로 구성되어 EPE 구조에 비해 이용률이 더 높은 효율적인 구조이다. $4t+2$개의 기본 셀을 사용하는 EPE 구조는

키 방정식 연산을 위해서 $8t+4$ 레지스터, $16t+82$-입력 멀티플렉서, $8t+4$ 유한체 곱셈기, $4t+2$ 유한체 덧셈기를 사용한다.

$2t$ 개의 기본 셀을 사용하는 ME(modified Euclid) 복호기[17]는 $50t$ 레지스터, $38t$ 2-입력 멀티플렉서, $8t$ 유한체 곱셈기, $4t$ 유한체 덧셈기를 사용한다. 각 기본 셀은 레지스터 25개, 2-입력 멀티플렉서 19개, 유한체 곱셈기 4개, 유한체 덧셈기 2개로 구성된다. FIFO 메모리를 제외한 전체 RS 복호기의 게이트 수는 55,200개이며[17] 수정 유클리드 블록의 게이트 수는 43,100개이다[18].

$3t+2$ 개의 기본 셀로 구성된 제안하는 DCME 구조는 $6t+4$ 레지스터, $18t+12$ 2-입력 멀티플렉서, $6t+4$ 유한체 곱셈기, $3t+2$ 유한체 덧셈기를 사용한다. DCME 구조는 EPE 구조[16]에 비해 비슷한 크기의 기본 셀을 더 적은 개수만큼 사용한다. ME 구조[17]는 사용하는 기본 셀의 개수가 $2t$개로 DCME 구조의 $3t+2$개보다 적지만 제안하는 DCME의 기본 셀은 단순한 구조를 갖는다. 따라서 제안한 RS 복호기는 기존 복호기[16, 17]에 비해 작은 게이트를 사용하여 구현 가능하다. DCME 구조와 FIFO 메모리를 제외한 전체 RS 복호기의 게이트 수는 각각 21,760개와 42,213개이다. 따라서 제안한 RS 복호기는 adaptive 복호기[15]에 비해 80%, ME 복호기[17]에 비해 23%의 하드웨어 면적이 향상되었다.

DCME 구조는 키 방정식 연산을 위해 $2t$ 사이클을 필요로 한다. 반면에 adaptive 복호기[15]는 $(4t^2+t\ t+20)pr$ 사이클을 필요로 한다. p는 기본 셀의 개수이며 r은 수정 유클리드 블록과 전체 복호기의 동작 주파수 비율이다. 또한, EPE 구조[16]는 $2t+1$, ME 구조[17]는 $3t+37$ 사이클의 지연 시간을 갖는다. 따라서 제안한 구조는 adaptive RS 복호기[15]에 비해 65%, ME 복호기[17]에 비해 10% 속도가 향상되었다. adaptive 복호기[15]의 동작 주파수는 30MHz이며, 0.25㎛ 라이브러리를 사용하는 ME 복호기[17]의 동작 주파수는 75MHz이다. ME 복호기[17]의 키 방정식 블록만을 0.16㎛ 라이브러리를 사용하여 구현하면 300MHz의 동작 주파수를 갖는다[18]. 그러나 0.25㎛ 라이브러리를 사용하는 제안하는 RS 복호기는 다항식의 차수 연산 및 비교 회로가 없어 200MHz의 동작 주파수를 갖으며, [18]과 같이 0.16㎛ 라이브러리를 사용할 경우 200MHz보다 훨씬 높은 동작 주파수를 가질 것이다.

V. 결 론

본 논문에서는 키 방정식 연산을 위한 새로운 DCME 알고리즘을 제안한다. DCME 알고리즘은 다항식 차수 연산 및 비교 회로가 필요 없어 기존 수정 유클리드 알고리즘에 비해 낮은 하드웨어 복잡도를 갖는다. 따라서 제안한 DCME 복호기는 다양한 (n, k, t) RS 부호의 저비용 고속 RS 복호가 가능하다. (255, 239, 8) RS 부호에서 DCME 복호기는 42,213개의 게이트를 사용하며, 기존 RS 복호기에 비해 최소한 23%의 면적 감소 및 10% 이상 연산 속도가 향상되었다. 또한, 0.25㎛ 라이브러리를 사용하는 제안한 복호기는 200MHz의 동작 주파수 및 1.6Gbps의 데이터 처리 속도를 갖는다. 따라서 제안한 구조는 짧은 지연 시간 및 낮은 하드웨어 복잡도를 가지므로 PLC, DVB-T, VSB, cable modem, WATM, 위성 통신, 이동 통신, DVD 등 다양한 응용 분야에 활용 가능하다.

참 고 문 헌

[1] A. Raghupathy and K. J. R. Liu, "Algorithm-based low-power/high-speed Reed-Solomon decoder design," IEEE Trans. Circuit Syst. II, vol. 47, pp. 1254~1270, Nov. 2000.

[2] HomePlug Powerline Alliance, Medium Interface Specification Release 0.5, Nov. 2000.

[3] DVB, Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television, ETSI EN 300 744 v1.4.1, Jan. 2001.

[4] ATSC, ATSC Digital Television Standard, ATSC standard A/53B, Aug. 2001.

[5] DAVIC 1.4 Specification Part 08, Lower Layer Protocols and Physical Interface, 1998.

[6] A. M. Michelson and A. H. Levesque, Error-Control Techniques for Digital Communication. New York : Wiley, 1985.

[7] T. R. N. Rao and E. Fujiwara, Error Control Coding for Computer Systems. Englewood Cliffs, NJ : Prentice-Hall, 1989.

[8] S. B. Wicker and V. K. Bhargava, Reed-Solomon Codes and Their Applications. IEEE

Press, 1994.

[9] 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. Commun., vol. 47, pp. 1488~1494, Oct. 1999.

[10] H. J. Kang and I. C. Park, "A high-speed and low-latency Reed-Solomon decoder based on a dual-line structure," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS' 2002), May 2002, pp. 3180~3183.

[11] 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. Video Technol., vol. 7, pp. 864~871, Dec. 1997.

[12] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," IEEE Trans. VLSI Syst., vol. 9, pp. 641~655, Oct. 2001.

[13] M. A. A.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 Techn. Conf. (VTC'99), vol. 1, 1999, pp. 92~96.

[14] T. K. Matsushima, T. Matsushima, and S. Hirasawa, "Parallel architecture for high-speed Reed-Solomon codec," in Proc. IEEE Int. Telecommun. Symp. (ITS'98), vol. 2, 1998, pp. 468~473.

[15] M. K Song, E. B. Kim, H. S. Won and M. H. Kong, "Architecture for decoding adaptive Reed-Solomon codes with variable block length," IEEE Trans. Consumer Elec., vol. 48, pp. 631~637, Aug. 2002.

[16] C. T. Huang and C. W. Wu, "VLSI design of a high speed pipelined Reed-Solomon CODEC," in Proc. Int. Symp. Multi-Technology inform. Processing (ISMIP), Dec. 1996, pp. 517~522.

[17] H. H. Lee, M. L. Yu and L. Song, "VLSI design of Reed-Solomon decoder architectures," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS' 2000), vol. 5, May 2000, pp. 705~708.

[18] H. H. Lee, "Modified Euclidean algorithm block for high-speed Reed-Solomon decoder," IEE Electronics Letters, vol. 37, pp. 903~904, July 2001.

[19] 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.

[20] 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.

[21] X. Yuanxin, X. Fang, Y. Qingdong, Q. Peiliang, and W. Kuang, "A new VLSI design for decoding Reed-Solomon codes based on ASIP," in Proc. Int. ASIC Conference, 2001, pp. 448~451.

[22] K. Iwamura, Y. Dohi, and H. Imai, "A design of Reed-Solomon decoder with systolic-array structure," IEEE Trans. Comput., vol. 44, pp. 118~122, Jan. 1995.

[23] M. Martina, G. Masera, G. Piccinini, F. Vacca, and M. Zamboni, "VLSI Reed Solomon decoder architecture for networked multimedia applications," in Proc. IEEE Int. ASIC/SOC Conference, 2001, pp. 347~351.

[24] J. H. Baek, J. Y. Kang and Myung H. Sunwoo, "Design of a high-speed Reed-Solomon decoder," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS' 2002), May 2002, pp. 793~796.

저 자 소 개

白 在 鉉(正會員)

2002년 2월 : 아주대학교 전자공학 학사. 2003년 2월 아주대학교 전자공학 석사. 2003년 2월~현재 : 아주대학교 전자공학과, 박사과정 재학 중. <주관심분야 : 통신 및 신호처리용 ASIC 설계>

鮮 于 明 勳(正會員)

1980년 2월 : 서강대학교 전자공학 학사. 1982년 2월 : 한국과학기술원 전 자 공학 석사. 1982년 3월~1985년 8월 : 한국전자통신연구소(ETRI) 연구원. 1985년 9월~1990년 8월 : Univ. of Texas at Austin 전자공학 박사. 1990년 8월~1992년 8월 : Motorola, DSP Chip Division (미국). 1992년 8월~1996년 10월 : 아주대학교 전기전자공학부 조교수. 1996년 10월~2001년 9월 : 아주대학교 전자공학부 부교수. 2001년 10월~현재 : 아주대학교 전자공학부 교수. <주관심분야 : VLSI 및 Parallel Architecture, 통신 멀티미디어용 DSP 칩 및 ASIC 설계>