UWB 시스템 용 Reed-Solomon 복호기 설계

정회원 조 용 석*

Design of A Reed-Solomon Decoder for UWB Systems

Yong-suk Cho* *Regular Member*

요 약

본 논문에서는 오류정정 능력이 비교적 작은 경우에 매우 효율적인 직접복호법을 이용하여 기존의 복호기에 비해 하드웨어적으로 매우 간단한 UWB 용 (23, 17) Reed-Solomon 복호기의 설계 방법을 제안한다. 설계된 복호기는 오류위치다항식 및 오류평가다항식의 계산에 $GF(2^m)$ 상의 곱셈기가 9개만 사용되어, 기존의 복호기가 약 20여 개가 소요되는데 비해 매우 간단한 하드웨어로 구현할 수 있는 장점을 가지고 있다. 또한 제어회로도 매우 간단하고, 복호지연도 오증계산에 걸리는 한 블록만큼만 소요되므로 수신 시퀀스를 저장하는 버퍼 메모리를 절약할 수 있다.

Key Words : UWB, Reed-Solomon Decoder, Finite Field, Galois Field, Error Correcting Code

ABSTRACT

In this paper, we propose a design method of Reed-Solomon (23, 17) decoder for UWB using direct decoding method. The direct decoding algorithm is more efficient for the case of relatively small error correction capability. The proposed decoder requires only 9 $GF(2^m)$ multipliers in obtaining the error-locator polynomial and the error-evaluator polynomial, whereas other decoders need about 20 multipliers. Thus, the attractive feature of this decoder is its remarkable simplicity from the point of view of hardware implementation. Futhermore, the proposed decoder has very simple control circuit and short decoding delay. Therefore this decoder can be implemented by simple hardware and also save buffer memory which stores received sequence.

Ⅰ. 서 론

UWB (Ultra Wide Band)는 저전력의 넓은 주파수 대역폭을 사용하여 고속으로 데이터를 전송하는 무선통신 기술이다. 미국 FCC (Federal Communication Commission)는 UWB를 중심 주파수의 20% 이상의 점유 대역폭을 갖거나 500 MHz 이상의 점유 대역폭을 차지하는 무선 전송기술로 정의하고 있다. UWB 기술은 10 미터 내에서 100 Mbps 이상으로 데이터를 전송할 수 있다. 이 기술은 가정에서 고속으로 멀티미디어 데이터를 전송하는 응용에 적합하다. UWB는 IEEE 802.15.3a로 표준화가 진행되었다. MB-OFDM (Multi Band Orthogonal Frequency Division Multiplexing)과 DS-UWB (Direct Sequence UWB)가 표준기술로 경쟁하였다.

MB-OFDM UWB의 프레임(PPDU)은 크게 PLCP 프리앰블, PLCP 헤더, PSDU로 구성되며, PLCP 헤더는 PHY 헤더, MAC 헤더, HCS, tail bit, RS parity로 구성되고, Reed-Solomon (23, 17) 부호와 길쌈부호(convolutional codes)를 사용하여 PLCP 헤더를 보호하고 있다[1]. PHY 헤더와 MAC 헤더에는 송수신 시에 사용되는 중요한 정보들이 포함되어 있다. 특히 PHY 헤더에는 payload의 전송속도, 길이 등의 정보가 들어있기 때문에 복호지연이 짧아

* 영동대학교 정보통신사이버경찰학과(yscho@yd.ac.kr)
논문번호 : KICS2010-12-637, 접수일자 : 2010년 12월 30일, 최종논문접수일자 : 2011년 4월 4일

야 하고 고속으로 동작하여야 한다[2],[3].

Reed-Solomon (RS) 부호의 일반적인 복호 방법은 수신 시퀀스로부터 오증(syndrome)을 구하고, 그 오증으로부터 오류위치다항식(error-locator polynomial)과 오류평가다항식(error-evaluator polynomial)을 구한 다음, 이들을 이용하여 오류위치(error location)를 찾고 그 위치에 해당하는 오류값(error value)을 구하여 오류를 정정하는 것이다[4].

이 과정 중에서 가장 핵심이 되는 부분이 오류위치다항식과 오류평가다항식을 구하는 것으로 이의 해법에 따라 Berlekamp-Massey 복호법, Euclid 복호법, PGZ (Peterson-Gorenstein-Zierler) 복호법, 직접복호법 등으로 분류하고 있다[5]. 앞의 2가지 복호법은 오류정정능력이 큰 부호에 효율적인 복호법으로 반복 계산에 의해 복호를 수행하므로 소프트웨어 적으로 복호기를 실현할 경우 적합한 복호법이며, 뒤의 2가지 복호법은 오류정정능력이 비교적 작은 부호에 대하여 하드웨어 적으로 보다 고속의 복호기를 실현하는 경우에 적합한 복호법이다.

직접복호법은 오증으로부터 오류위치다항식을 구한 다음 오류위치와 이에 해당하는 오류값을 구하는 일반적인 복호법과는 달리, 오증으로부터 직접 오류위치와 오류값을 구하여 오류를 정정하는 방법이다. RS 부호의 직접복호 알고리듬은 堀口敏男와 佐藤善彦[6]이 처음 제안하였으며 조용석과 박상규[7]는 직접복호법을 이용한 3중 오류정정 복호기를 제안하였다. 본 논문에서는 이를 개선하여 하드웨어적으로 매우 효율적인 UWB 용 (23, 17) RS 부호의 복호기를 설계한다.

기존의 PGZ 알고리듬으로 설계한 복호기[8]와 Euclid 알고리듬으로 설계한 복호기[3],[9]가 오류위치다항식 및 오류평가다항식의 계산에 $GF(2^m)$ 상의 곱셈기가 각각 24개와 20개가 소요되는데 비해, 설계된 복호기는 9개만 사용되어 매우 간단한 하드웨어로 구현할 수 있는 장점을 가지고 있다. 또한 제어회로도 매우 간단하고, 복호지연도 오증계산에 걸리는 한 블럭만큼의 지연만 소요되므로 매우 짧은 특징을 가지고 있다. 복호지연이 짧다는 것은 하드웨어적으로 보면 수신 시퀀스를 저장하는 버퍼 메모리가 그만큼 절약된다는 것을 의미한다.

본 논문의 구성은 다음과 같다. 먼저 Ⅱ장에서 Reed-Solomon 부호의 직접복호 알고리듬을 분석한다. Ⅲ장에서는 이를 토대로 기존의 복호기에 비해 하드웨어적으로 매우 간단한 UWB 용 (23, 17) RS 복호기의 설계 방법을 제안하고 이를 다른 복호법으로 설계한 기존의 복호기와 하드웨어 복잡도를 비교한다. 끝으로 Ⅳ장에서 결론을 맺는다.

Ⅱ. RS 부호의 직접복호 알고리듬

유한체 $GF(2^m)$의 원시원(primitive element)을 α라 할 때, t개 이하의 모든 오류를 정정할 수 있는 t중 오류정정 RS 부호에서, $u\,(1 \leq u \leq t)$개의 오류가 $i_1,\ i_2, \cdots\cdots, i_u\ (i_1 < i_2 < \cdots\cdots < i_u)$ 위치에서 발생하였다고 가정하고 오류위치를 $X_k\ (=\alpha^{i_k})$, 오류값을 Y_k라 하면 오증 s_j는

$$\begin{aligned} s_j &= \sum_{i=1}^{u} Y_i X_i^{\,j} \\ &= \sum_{i=1}^{u} (Y_i X_i) X_i^{\,j-1},\ \ j = 1, 2, \cdots\cdots, 2t \end{aligned} \tag{1}$$

가 된다. 여기에서 오증을 요소로 하는 $u \times u$ 행렬 $M_u(s)$를 다음과 같이 정의한다.

$$M_u(s) = \begin{bmatrix} s_1 & s_2 & \cdots & s_u \\ s_2 & s_3 & \cdots & s_{u+1} \\ s_3 & s_4 & \cdots & s_{u+2} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ s_u & s_{u+1} & \cdots & s_{2u-1} \end{bmatrix} \tag{2}$$

식 (1)을 대입하여 이 행렬의 행렬식(determinant) $|M_u(s)|$를 전개하면 다음과 같이 된다[6].

$$|M_u(s)| = \prod_{i=1}^{u} Y_i X_i \prod_{i > j \geq 1}^{u} (X_i + X_j)^2 \tag{3}$$

이 행렬식 $|M_u(s)|$는 오류가 정확하게 u개 발생하였을 때는 0이 아니고 u개 보다 적게 발생하였을 경우에는 0이 된다[10].

여기에서 오증에 관한 다음과 같은 함수를 정의한다.

$$S_j(x) = s_j x^j, \qquad j = 1, 2, \cdots\cdots, 2t \tag{4}$$

$$\begin{aligned} A_j(x) = &\ S_j(x) + S_{j+1}(x)\ , \\ &\ j = 1, 2, \cdots\cdots, 2t-1 \end{aligned} \tag{5}$$

$$B_j(x) = S_j(x) + S_{j+2}(x), \quad j = 1,2,\cdots\cdots,2t-2 \tag{6}$$

식 (4)~(6)에 식 (1)을 대입하여 정리하면 다음과 같이 쓸 수 있다.

$$\begin{aligned} S_j(x) &= \sum_{i=1}^{u} Y_i X_i^j x^j \\ &= \sum_{i=1}^{u} Y_i X_i x (X_i x)^{j-1} \end{aligned} \tag{7}$$

$$A_j(x) = \sum_{i=1}^{u} [Y_i X_i x (1 + X_i x)] (X_i x)^{j-1} \tag{8}$$

$$B_j(x) = \sum_{i=1}^{u} [Y_i X_i x (1 + X_i x)^2] (X_i x)^{j-1} \tag{9}$$

여기에서 $A_j(x)$를 요소로 하는 식 (2)와 동일한 형태를 갖는 행렬의 행렬식 $|M_u(A(x))|$를 식 (3)과 같은 방법으로 전개하면 다음과 같이 쓸 수 있다.

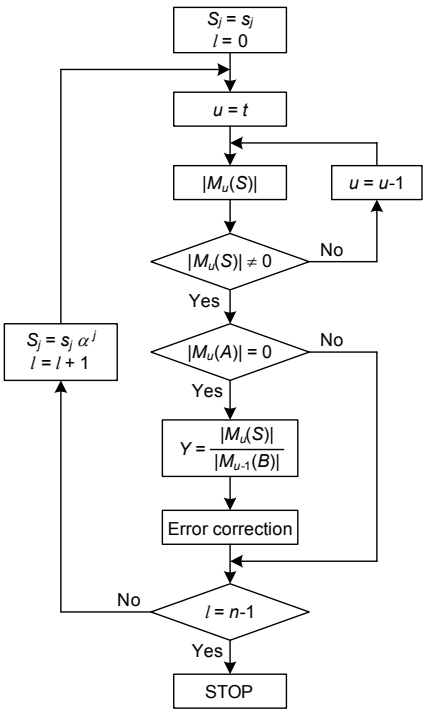

그림 1. 직접복호 알고리듬의 순서도

$$\begin{aligned} |M_u(A(x))| &= \prod_{i=1}^{u} Y_i X_i x (1 + X_i x) \\ &\quad \cdot \prod_{i > j \geq 1}^{u} (X_i x + X_j x)^2 \\ &= x^{u^2} \cdot |M_u(s)| \cdot \prod_{i=1}^{u} (1 + X_i x) \end{aligned} \tag{10}$$

식 (10)을 살펴보면 행렬식 $|M_u(s)|$는 u개의 오류가 발생하였을 경우 0이 아니므로 $|M_u(A(x))| = 0$를 만족하는 근이 오류위치가 된다. 즉 행렬식 $|M_u(A(x))|$를 오류위치다항식으로 이용할 수 있다. 또한 식 (7)과 식 (9)를 이용하면 오류위치 X_k^{-1}에 대응하는 오류값 Y_k를 다음과 같이 구할 수 있다.

$$Y_k = \frac{|M_u(S(X_k^{-1}))|}{|M_{u-1}(B(X_k^{-1}))|}, \quad k = 1, 2, \cdots\cdots, u \tag{11}$$

따라서 식 (10)과 식 (11)을 이용하면 그림 1과 같이 직접복호법의 복호 절차를 정리할 수 있다.

Ⅲ. UWB 용 (23, 17) RS 복호기

UWB 시스템에서 규정하고 있는 RS 부호는 원시다항식(primitive polynomial)이 식 (12)와 같은 유한체 $GF(2^8)$ 상의 3중 오류정정 부호인 (255, 249) RS 부호를 232 심벌만큼 단축시킨 (23, 17) RS 부호이다.

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \tag{12}$$

본 논문에서는 (23, 17) RS 복호기를 그림 2와 같이 오증 계산 및 치환부, 오류위치 및 오류값 계산부, 오류정정부의 3부분으로 나누어 설계한다.

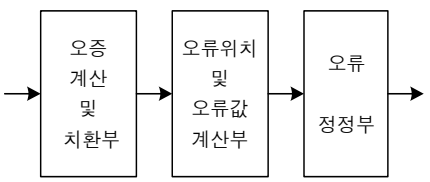

그림 2. (23, 17) RS 복호기의 블럭도

3.1 오증 계산 및 치환부

(23, 17) RS 부호는 (255, 249) RS 부호를 232 심벌만큼 단축시킨 부호이므로 S_j는

$$S_j(x) = s_j(\alpha^{232})^j, \quad j=1,2,\cdots\cdots,6 \qquad (13)$$

가 된다. 여기에서 α는 식 (12)와 같은 원시다항식의 근이다.

식 (13)을 이용하면 그림 3과 같이 오증 계산 및 치환부를 설계할 수 있다.

그림 3에서 굵은 선은 m비트 버스를 표시한 것이며 점선은 계산된 오증을 병렬 로드(load)하는 것을 나타내고 있다. ⊕는 $GF(2^m)$ 상의 덧셈기로 m개의 2입력 Exclusive OR 게이트로 구현할 수 있다. 또한 □는 m비트 레지스터를, ⓐ는 α^i를 곱하는 상수 곱셈기를 나타내고 있다.

3.2 오류위치 및 오류값 계산부

한 블럭에서 발생하는 3개 이하의 모든 오류를 정정하는 3중 오류정정 부호는 오류가 1개, 2개, 3개 발생한 경우를 모두 고려하여야 한다. 먼저 오류

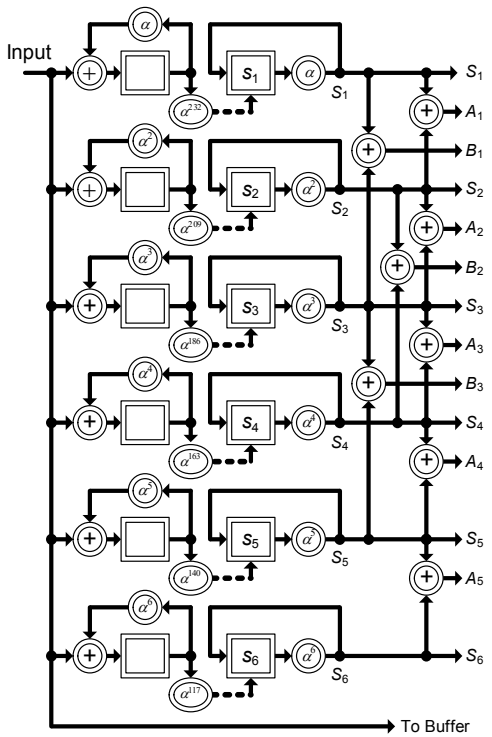

그림 3. 오증 계산 및 치환부

위치다항식 $|M_1(A)|$, $|M_2(A)|$, $|M_3(A)|$는 다음과 같이 구할 수 있다.

$$|M_1(A)| = A_1 \qquad (14)$$

$$|M_2(A)| = \begin{vmatrix} A_1 & A_2 \\ A_2 & A_3 \end{vmatrix} = A_2^2 + A_1A_3 \qquad (15)$$

$$|M_3(A)| = \begin{vmatrix} A_1 & A_2 & A_3 \\ A_2 & A_3 & A_4 \\ A_3 & A_4 & A_5 \end{vmatrix} = A_3^3 + A_1A_4^2 + A_5|M_2(A)| \qquad (16)$$

같은 방법으로 $|M_1(S)|$, $|M_2(S)|$, $|M_3(S)|$, $|M_1(B)|$, $|M_2(B)|$를 구하면 다음과 같이 된다.

$$|M_1(S)| = S_1 \qquad (17)$$

$$|M_2(S)| = S_2^2 + S_1S_3 \qquad (18)$$

$$|M_3(S)| = S_3^3 + S_1S_4^2 + S_5|M_2(S)| \qquad (19)$$

$$|M_1(B)| = B_1 = S_1 + S_3 \qquad (20)$$

$$|M_2(B)| = B_2^2 + B_1B_3 \qquad (21)$$

또 식 (11)과 같은 오류값 계산식은 실제 발생한 오류의 개수 u가 1, 2, 3일 경우 각각 다르므로 이들을 각각 $Y^{(1)}$, $Y^{(2)}$, $Y^{(3)}$라 하면 다음과 같이 쓸 수 있다.

$$Y^{(1)} = \frac{|M_1(S)|}{|M_0(B)|} = S_1, \ (|M_0(B)|=1) \qquad (22)$$

$$Y^{(2)} = \frac{|M_2(S)|}{|M_1(B)|} = \frac{S_2^2 + S_1S_3}{B_1} \qquad (23)$$

$$Y^{(3)} = \frac{|M_3(S)|}{|M_2(B)|} = \frac{|M_3(S)|}{B_2^2 + B_1B_3} \qquad (24)$$

따라서 식 (14)~(24)을 이용하면 그림 4와 같이 오류위치 및 오류값 계산부를 설계할 수 있다.

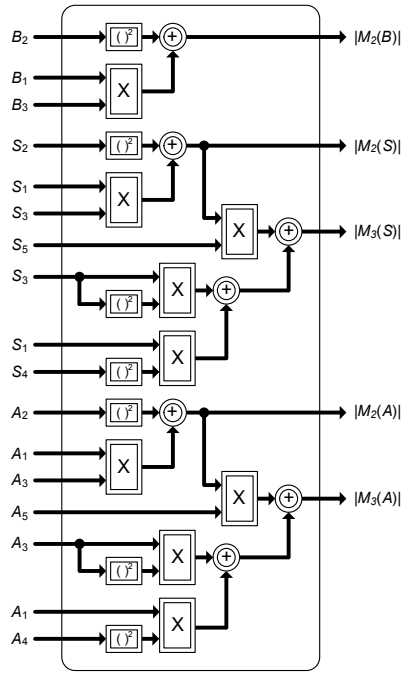

그림 4. 오류위치 및 오류값 계산회로

그림 4에서 [×]은 $GF(2^m)$ 상의 곱셈기를, [( )²]은 $GF(2^m)$ 상의 제곱기를 나타내고 있다.

3.3 오류 정정부

버퍼 레지스터의 출력과 더해지는 오류값 Y는 다음과 같이 정리할 수 있다.

$$Y=\begin{cases} Y^{(3)}, \text{ if } |M_3(S)|\neq 0 \text{ and } |M_3(A)|=0 \\ Y^{(2)}, \text{ if } |M_3(S)|=0 \text{ and } |M_2(S)|\neq 0 \\ \qquad\qquad \text{and } |M_2(A)|=0 \\ Y^{(1)}, \text{ if } |M_3(S)|=0 \text{ and } |M_2(S)|=0 \\ \qquad\qquad \text{and } S_1\neq 0 \text{ and } A_1=0 \end{cases} \quad (25)$$

따라서 식 (25)를 이용하면 그림 5와 같은 오류 정정부를 설계할 수 있다.

그림 5에서 [INV]는 $GF(2^m)$ 상의 역원기를, [MUX]는 m개의 2×1 멀티플렉서를 표시한 것이다. [= 0]은 입력 비트가 모두 0일 때 0을 출력하는 전영검출기(all zero detector)로 $m-1$개의 2입력 OR 게이트로 구현할 수 있다. 또 [OR]는 비트 별로 OR하는 회로로 m개의 2입력 OR 게이트로 구현할 수 있으며, [gate]는 선택(select) 입력이 1일 때에만 입력을 출력으로 통과시키는 회로로 m개의 2입력 AND 게이트로 구현할 수 있다.

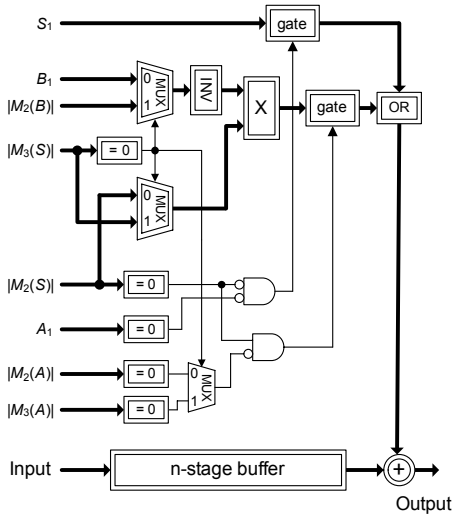

그림 5. 오류 정정부

그림 3~5와 같이 설계된 곱셈기의 동작을 검증하기 위하여 ALTERA의 MAX+PLUS II 툴을 사용하여 VHDL로 구현하고 시뮬레이션 하였다.

그림 6은 모두 0인 부호어를 전송하여 위치 22, 21, 20에서 α^{98}(0x43), α^{4}(0x10), α^{210}(0x59)의 오류가 발생한 경우의 시뮬레이션 결과이다. 한 프레임 후에 동일한 위치에 동일한 오류값이 출력되는 것을 볼 수 있다.

설계된 복호기와 기존의 복호기의 하드웨어 복잡도를 비교할 때 그림 2와 같은 구성에서 오증 계산 및 치환부와 오류 정정부의 하드웨어 복잡도는 비슷하다고 가정하면, 하드웨어 복잡도에 결정적으로 영향을 미치는 것은 오류위치 및 오류값 계산부에 사용되는 유한체 곱셈기의 개수가 된다. 표 1에 기존의 복호기와의 유한체 연산기의 수를 비교하였다.

표 1에서 보듯이 기존의 PGZ 알고리듬으로 설계한 복호기[8]와 변형 Euclid 알고리듬으로 설계한 복호기[3]가 각각 24개와 20개의 곱셈기를 사용한 반면에 본 논문에서 설계한 복호기는 그림 4와 같이 9개의 곱셈기만을 사용하므로 하드웨어 복잡도 면에서 기존의 복호기에 비해 훨씬 우수함을 알 수

표 1. 오류위치 및 오류값 계산에 소요되는 유한체 연산기

$GF(2^m)$ 상의 연산기	문헌 [8]의 복호기	문헌 [3]의 복호기	본 논문의 복호기
곱 셈 기	24 개	20 개	9 개
덧 셈 기	15 개	10 개	15 개
제 곱 기	4 개	–	7 개

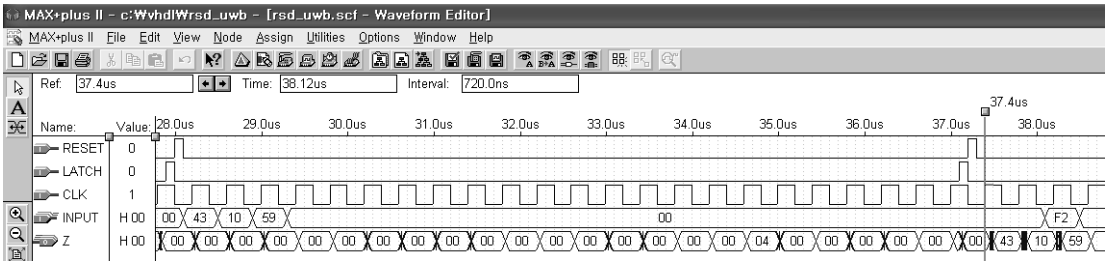

그림 6. 설계된 복호기의 시뮬레이션 결과

있다. 또한 그림 5와 같이 오류 정정을 위한 제어 회로도 본 논문에서 설계한 복호기가 하드웨어적으로 훨씬 더 간단한 장점을 가지고 있다.

Ⅳ. 결 론

본 논문에서는 RS 부호의 복호법 중에서 오류정정능력이 비교적 작을 경우 효율적인 직접복호법을 이용하여 하드웨어적으로 매우 간단한 UWB 용 (23, 17) RS 복호기를 설계하였다. 설계된 복호기는 VHDL로 구현하여 그 동작을 검증하였다.

설계된 (23, 17) RS 복호기는 오류위치와 오류값 계산에 $GF(2^m)$ 상의 곱셈기가 9개만 소요되어 기존의 복호기가 약 20여개가 소요되는데 비해 훨씬 간단한 하드웨어로 구현할 수 있다. 또한 제어회로도 매우 간단하며, 복호지연도 오증계산에 걸리는 부호길이만큼만 소요되므로 수신 시퀀스를 저장하는 버퍼 메모리를 절약할 수 있는 장점을 가지고 있다.

참 고 문 헌

〔1〕 한국정보통신기술협회, "WiMedia UWB 멀티밴드 직교주파수 분할 물리계층", *정보통신단체표준*, TTAE.OT-06.0026, 2008.

〔2〕 S. W. Choi, S. S. Choi, H. Lee, "RS Decoder Architecture for UWB," *IEEE ICACT 2006*, pp.805-808, 2006.

〔3〕 강성진, 김한종, "UWB 시스템을 위한 RS (23, 17) 복호기 최적 설계", *한국통신학회논문지*, Vol.33, No.8, pp.821-828, 2008.

〔4〕 M. Y. Rhee, *Error Correcting Coding Theory*, McGraw-Hill, New York, 1989.

〔5〕 S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications*, IEEE Press, 1994.

〔6〕 堀口 敏男, 佐藤 善彦, "*GF*(2^m)の 上の 擴大 Reed-Solomon 符號の 一復號方式", *電子通信學會論文誌*, Vol.J66-A, pp.97-98, 1983.

〔7〕 조용석, 박상규, "Reed-Solomon 부호의 직접복호법을 이용한 3중 오류정정 복호기 설계", *한국통신학회 논문지*, Vol.24, No.8A, pp.1238-1244, 1999. 8.

〔8〕 A. M. Patel, "On-the-fly Decoder for Multiple Byte Errors," *IBM J. RES. DEVELOP.*, Vol.30, No.3, pp.259-269, May, 1986.

〔9〕 H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Joseph, and I. S. Reed, "A VLSI Design of a Pipeline Reed-Solomon Decoder," *IEEE Trans. Computers*, Vol.C-34, No.5, pp.393-403, May, 1985.

〔10〕 W. W. Peterson and E. J. Weldon, *Error-Correcting Codes*, MIT Press, Cambridge, Mass., 1972.

조 용 석 (Yong-suk Cho) 정회원

1986년 2월 한양대학교 전자통신공학과 학사

1988년 2월 한양대학교 전자통신공학과 석사

1998년 8월 한양대학교 전자통신공학과 박사

1989년 4월~1996년 2월 한국전기통신공사 연구개발단 전임연구원

1996년 3월~현재 영동대학교 정보통신사이버경찰학과 교수

<관심분야> 유한체연산, 오류정정부호, 암호시스템