개선된 변환영역 RS 복호기의 VLSI 설계

박혁찬, 박종진, 국일호, 조원경
경희대학교 전자공학과
nandh@csvlsi.kyunghee.ac.kr

A VLSI Design of Modified Transform RS Decoder

Hyukchan Park, Jongjin Park, Ilho Kook, Wonkyung Cho
Dept. of Electronics Eng. Kyunghee Univ.

Abstracts

In this paper, a RS(Reed-Solomon) decoder is designed in the transform domain instead of most time domain. The transform RS decoder have simpler structure for error-correction procedure but because of his larger chip area, the time domain RS decoder is popular currently. To solve this problem, the normal basis representation and the conjugate property is utilized. Therefore the chip area can be reduced for the structure of syndrome delay, normalization and inverse transform circuit. These modified structures have been implemented using VHDL and synthesized on 0.8 μm CMOS technology. The results have been compared with other structure for chip area and performance.

1. 서론

RS부호는 적은 redundancy로도 강력한 에러 정정 능력을 갖는 에러 정정 부호로서 non-binary 형태이어서 산발에러뿐만 아니라 연집 에러에도 강한 특성을 갖으며 현재 군사통신, 무선통신, 위성통신, 데이터통신, 컴퓨터의 저장 시스템 등에 널리 이용되고 있다.[6][7]

변환영역 RS 복호기는 에러를 구하는 과정이 단순한 반면 VLSI화하였을 때 칩 면적이 크다는 이유로 현재 시간영역 복호기가 주로 설계되고 있다.[2]

본 논문에서는 공액(conjugate) 특성과 normal basis 표현을 이용 역변환 회로의 크기를 줄일 수 있는 구조를 제안하고 또한 신드롬 지연기 및 정규화(normalization) 회로의 크기도 줄일 수 있는 구조를 제시하였다. 이렇게 하여 개선된 구조의 변환영역 RS 복호기를 VHDL을 이용 설계 및 실제 공정에 합성하여 VLSI 구현 시 칩 면적 및 성능을 확인하였고 기존의 시간영역 및 변환영역의 복호기와도 상대적으로 비교하였다. 그리고 마지막으로 실제 응용될 수 있는 분야 및 앞으로의 연구 방향을 제시하였다.

2. RS 부호의 복호 과정[1][2][7]

신드롬을 구하는 과정은 수신 다항식을 생성 다항식으로 나누는 과정인데 이는 생성 다항식의 근을 대입하는 것이며 또한 Fourier 변환 식의 형태를 가진다. 따라서 신드롬은 변환 영역에서의 에러의 일부가 되고 변환 복호기는 변환 영역 에러의 나머지를 구하는 과정을 수행하게 된다.

변환복호 과정은 우선 신드롬을 구하고 이로부터 Euclid 또는 Berlekamp 순환 알고리듬을 이용하여 에러위치다항식을 계산한 후 구해진 신드롬과 에러위치다항식을 이용 변환 영역에서의 에러 성분을 구하고 역 변환하여 에러를 구하고 정정을 하게 된다.

시간영역 상의 복호는 에러위치 다항식뿐만 아니라 에러평가 다항식이 필요하며 두 다항식으로부터 에러의 위치와 크기를 계산하고 최종적으로 에러를 정정하게 된다.

변환 영역 상의 복호기는 시간 영역에 비해 에러를 구하는 과정이 단순하나 에러위치 다항식을 정규화 하는 회로와 역변환 회로 그리고 신드롬 지연기가 필요하여 VLSI로 구현했을 때 칩 면적이 커지는 단점이 있는데 본 논문에서는 이러한 단점을 개선하였다.

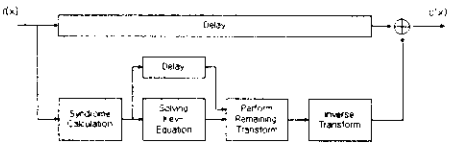

그림 1. 변환영역 상의 복호 과정

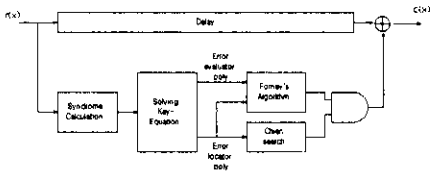

그림 2. 시간영역 상의 복호 과정

3. 개선된 복호기의 구조

3.1 역변환 회로

본 논문에서는 제곱 연산이 단순한 normal basis 표현과 유한체 Fourier 변환의 공액 특성을 이용 역변환 회로의 크기를 줄였다.

유한체 $GF(2^m)$ 상에서 $\{\alpha, \alpha^2, \alpha^4, \cdots, \alpha^{2^{m-1}}\}$이 선형독립이면 이들을 normal basis라 하며 이를 이용하면 제곱 연산을 단순히 한 비트 순회 치환함으로써 구현할 수 있게 된다.[3]

또한 Fourier 변환의 공액 특성을 이용할 수 있는데 c를 GF(q)상의 원소를 성분으로 하는 n차원 벡터라 할 때, $GF(q^m)$상의 변환벡터 C로의 이산 Fourier 변환 (DFT) 및 역변환은 다음 식과 같다.[7]

$$C_i = \sum_{j=0}^{n-1} \alpha^{ij} c_j, \qquad 0 \le i \le n-1$$

$$c_i = \sum_{j=0}^{n-1} \alpha^{-ij} C_j, \qquad 0 \le i \le n-1 \tag{1}$$

이러한 Fourier 변환은 다음과 같은 공액 관계가 성립한다.[6]

$$C_i^q = C_{(qi)}, \qquad i = 0, 1, 2, \cdots, n-1 \tag{2}$$

Modulo n은 다음과 같이 공액 집합으로 나눌 수 있다.

$$A_i = \{i, iq, iq^2, \cdots, iq^{m_i-1}\} \tag{3}$$

이러한 공액 집합은 스펙트럼 상의 주파수들의 집합으로 정의할 수 있고 시간영역의 신호가 GF(q)상에 존재할 때 한 primitive 주파수에서의 스펙트럼 값으로부터 공액 집합의 다른 모든 주파수에서의 스펙트럼 값을 유도할 수 있다.

변환 복호 과정에서 역변환은 $GF(2^m)$에서 $GF(2^m)$으로의 변환이다. 따라서 역변환 회로에 공액 특성을 적용하기 위해서 역변환 대상이 되는 심벌을 $GF(2^m)$에서 GF(2)로 즉 비트 분할하였다.

변환 에러의 각 계수 E_i는 $GF(2^m)$ 상의 원소로 GF(2) 상의 원소로 이루어진 벡터 $(B_i, B_{2i}, \cdots, B_{mi})$로 표현할 수 있으며 B_{mi}의 역변환 b_{mi}는 $GF(2^m)$ 상의 원소가 되며 에러를 구하는 역변환 식은 다음과 같이 표현할 수 있다.

$$\begin{aligned} e_i &= \sum_{j=0}^{n-1} \alpha^{-ij} E_j, \quad 0 \le i \le n-1 \\ &= \left(\sum_{j=0}^{n-1} \alpha^{-ij} B_i\right)\alpha + \left(\sum_{j=0}^{n-1} \alpha^{-ij} B_{2i}\right)\alpha^2 + \\ &\quad \cdots + \left(\sum_{j=0}^{n-1} \alpha^{-ij} B_{mi}\right)\alpha^{2^{m-1}} \\ &= b_i\alpha + b_{2i}\alpha^2 + \cdots + b_{mi}\alpha^{2^{m-1}} \end{aligned} \tag{4}$$

공액 관계에 있는 e_{2i}는 다음과 같이 제곱으로 유도할 수 있다.

$$\begin{aligned} e_{2i} &= b_{2i}\alpha + b_{4i}\alpha^2 + \cdots + b_{2mi}\alpha^{2^{m-1}} \\ &= b_i^2\alpha + b_{2i}^2\alpha^2 + \cdots + b_{mi}^2\alpha^{2^{m-1}} \end{aligned} \tag{5}$$

즉 공액 집합에서 한 primitive에 대해 역변환만 구하면 나머지 변환은 순회 치환하여 제곱을 구함으로써 유도할 수 있다. 각 비트에 대한 역변환은 최종적으로 더하는 과정이 필요하다. 이러한 역변환 회로를 그림 3.과 그림 4.와 같이 하드웨어로 구현하였다.

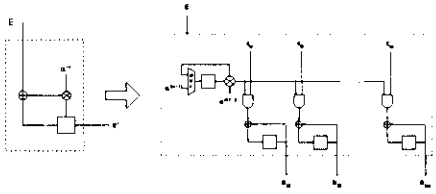

그림 3. processing cell의 개선

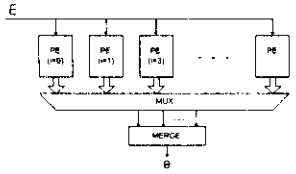

그림 4. 역변환 회로

공액집합은 부호어의 길이 n에 의해 결정되어지는데 n에 따라서 역변환회로에 필요한 곱셈기의 수와 레지스터의 수를 기존의 역변환 구조와 표 1.과 같이 상대적으로 비교하여 보았다. 표 1.와 같이 부호어의 길이 n이 클수록 더 많은 곱셈기가 감소함을 알 수 있다.

표 1. 부호어의 길이에 따른 역변환회로의 크기

부호어의 길이 (n)	기존 구조		개선된 구조	
	곱셈기 수	레지스터 수	곱셈기 수	레지스터 수
7	7	7	5	9
15	15	15	8	19
31	31	31	11	37
63	63	63	18	69
127	127	127	25	134
255	255	255	42	264

3.2 정규화 회로

변환 복호 과정에서 에러 정정 다항식을 구하고 최고차 항의 계수를 1로 만드는 정규화 과정이 필요하다. 이는 최고 차 항 계수로 전체 다항식을 나눔으로써 구현할 수 있다.[1][2]

그런데 나눗셈을 수행하기 위해서는 역원을 구해야 하는데 한 부호어를 처리하는 과정에서 한 번의 역원만을 구하면 된다. 그림 5.에서와 같이 GF(2^m)에서 m이 6이상일 경우 곱셈기에 비해 LUT(Look Up Table)의 크기가 상대적으로 커지게 되는데 이를 LUT로 구현하는 대신 곱셈기 하나로 구현할 수 있는 normal basis 역원계산기를 이용함으로써 하드웨어의 크기를 줄일 수 있다.

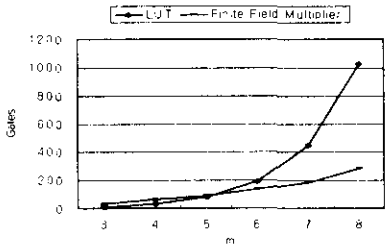

그림 5. GF(2^m)에 따른 크기 비교

유한체 GF(2^m) 상에서 임의의 한 원소 E의 역원은 다음과 같이 구할 수 있다.[3]

$$E^{-1} = E^{(2^m-1)-1} = E^{(2+2^2+\cdots+2^{m-1})} = (E^2)(E^{2^2})(E^{2^3})\cdots(E^{2^{m-1}}) \quad (6)$$

이를 하드웨어로 구현하면 그림 6.과 같다.

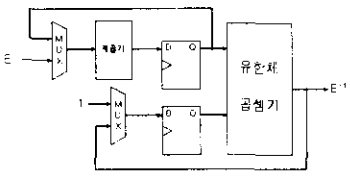

그림 6. 역원 계산기

3.3 신드롬 지연기

변환 복호 과정에서 에러 위치다항식을 구하는 동안 신드롬을 지연시킬 필요가 있는데 신드롬이 지연하는 동안 다음 신드롬이 입력되지 않는다. 따라서 에러위치 다항식을 계산하는 동안 소요되는 클럭만큼의 레지스터를 다 사용할 필요가 없다.

key equation을 구하기 위해 순환 알고리듬을 사용하거나 에러정정능력 t가 클 때 많은 클럭의 지연이 필요한데 이를 위해서 상당히 많은 레지스터가 필요하게 된다.

하지만 2t개의 신드롬이 레지스터에 입력된 후 값을 유지하다가 에러 정정 다항식이 계산되는 순간 순차적으로 출력할 수 있게 함으로써 레지스터의 수를 줄일 수 있다. 이러한 구조를 하드웨어로 구현하면 그림 7.과 같다.

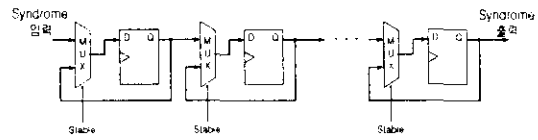

그림 7. 신드롬 지연기

4. 성능평가 및 비교

본론에서 제시한 구조를 이용하면 크기가 얼마나 줄어드는지를 erasure정정 능력을 갖춘 (255,223) RS 부호에 대해 Shao의 시간영역 및 변환영역 복호기[2]와 본 논문에서 제안한 변환영역 복호기를 구성부별로 비교를 해 보았으며 이를 표 2.과 같이 나타내었다.

표 2. RS (255,223)인 경우 영억별 칩 면적 비교

영역 / 구성부	Shao의 시간영역 복호기	Shao의 변환영역 복호기	개선된 변환영역 복호기	
			Euclid	Berlekamp
Look UP Table (8×256 ROM)	6	6	-	6
정규화 수행부		2	2	2
역원 계산기	-	-	2	2
신드롬 계산부	32	32	32	32
다항식 발생부 1	64	64	64	64
다항식 발생부 2	64	64	64	-
α^n 발생부	8	8	8	8
Berlekamp 수행부	-	-	-	60
Euclid 수행부	300	200	200	-
에러 변환부	-	64	64	64
다항식 평가부 1	32	-	-	
다항식 평가부 2	32	-	-	-
신드롬 지연부	-	256	16	16
역변환부	-	255	175	175
합 계	438 Ā	951 Ā	627 Ā	429 Ā

전체적인 복호기의 구조는 유한체 곱셈기, 덧셈기, reset입력을 가지는 레지스터로 구성된 processing cell 들을 기본으로 구성할 수 있다. 상수×변수의 연산을 수행하는 곱셈기를 포함하는 배선을 제외한 processing cell의 면적을 Ā라고 하였을 때 다음 도표에서는 각 수행 부의 면적을 이러한 Ā로 나타내었다. 변수×변수 연

산을 수행하는 곱셈기를 포함하는 processing cell은 약 2Å의 면적을 차지한다.[2]

정규화 수행 부는 역원계산기를 이용 구현했을 때 대략 4Å의 면적을 차지하며 8×256bits LUT를 이용하였을 때 대략 8Å 정도의 면적을 차지하므로 면적을 반정도 줄일 수 있다. 신드롬 지연기는 신드롬 계수의 개수만큼만 레지스터가 필요하고 멀티플렉스를 포함하는 레지스터는 대략 1/2Å의 면적을 차지한다. 역변환 회로는 n이 255인 경우 표 1.에 의하면 곱셈기가 모두 42개 필요하고 레지스터는 263개 필요하므로 제어를 위한 회로까지 포함하여 대략 255Å에서 175Å로 줄어들게 된다. 따라서 전체적인 변환 복호기의 면적이 다음과 같이 324Å만큼 줄어들게 된다.

그런데 변환 복호에서는 에러평가다항식이 필요 없으므로 Euclid 순환 알고리듬을 Berlekamp 순환 알고리듬으로 대체하면 필요한 cell의 수와 곱셈기가 줄어들어 면적을 대략 65% 정도 감소시킬 수 있고 또한 이 경우 Euclid 알고리듬처럼 신드롬의 다항식 확장을 하지 않아도 되어 이 부분을 제거할 수 있다.[4][5]

개선된 구조에 대해 실제 칩으로 구현하였을 때 얼마만큼의 면적이 줄어드는지 에러정정 능력 t가 3인 간단한 (31,25) RS부호의 복호기를 VHDL을 이용하여 설계한 후 Compass 합성 툴의 0.8 μm 공정 라이브러리를 이용하여 합성하여 보았고 결과는 다음 표 3.와 같다. 전체적인 크기가 크게 줄지 않는데 그 이유는 앞에서 제시하였지만 부호어의 길이 n과 에러정정능력 t가 작기 때문에 역변환 부와 신드롬 지연기에서 크게 면적이 감소하지 않았고 또한 GF(2^5)에서는 유한체 곱셈기가 LUT에 비해 크기 때문에 정규화 수행 부는 오히려 면적이 증가하였다.

표 3. 합성 결과 (Gate Equivalents)

설계방법 / 구성부	Shao의 구조	개선된 구조
신드롬 계산부	824.8	824.8
Euclid 수행부	6,216.5	6,216.5
정규화 수행부	353.8	825.5
에러 변환부	925.5	925.5
신드롬 지연부	707.8	363.8
역변환부	3,920.0	3215.0
전체 복호기의 크기	12,948.4	12,371.1

동작 및 성능에 대한 검증은 FPGA 합성 툴인 MAX+Plus II를 이용하였으며 FLEX10K에 합성한 결과 데이터 최대 전송률은 53 MHz을 보였고 또한 3개의 에러를 가지는 입력으로 그림 8.과 같이 simulation함으로써 에러정정능력을 확인하였다.

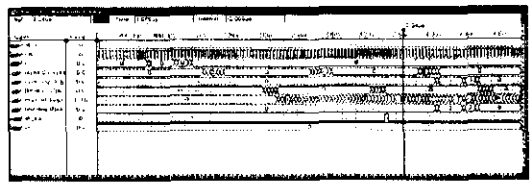

그림 8. Simulation 결과 파형

5. 결론

본 논문에서는 변환 영역의 복호기에 대해서 고찰하여 보았다. 변환 영역의 복호기는 시간 영역의 복호기에 비해 에러를 구하는 회로가 간단하나 VLSI로 구현하였을 때 칩 면적이 크다는 문제점이 있었는데 본 논문에서는 이러한 문제점을 해결할 수 있는 구조로 개선을 하였다.

제안한 구조는 부호어의 길이 n이 작은 경우보다 큰 경우에 칩 면적이 많이 감소되며 n이 작은 경우 개선의 대상이 된 수행 부의 크기 자체가 그다지 문제가 되지 않는다. 따라서 이러한 n에 따른 상관관계를 고려하여 설계하면 효율적인 복호기 설계에 가능하다.

변환 복호기는 n이 작은 경우 또는 단축부호 등에 적합하며 또한 수식적으로 단순한 구조이므로 소프트웨어로 처리하기에 적합한 구조이므로 이런 측면에서 실용화를 위한 연구가 필요하겠다.

참고문헌

[1] Howard M. Shao, T. K. Truong, Irving S. Reed, "A VLSI Design of a Pipeline Reed-Solomon Decoder", IEEE Transactions on Computers, vol.c-34, no.5, May 1985, pp.393-403

[2] Howard M. Shao, Irving S. Reed, "On the VLSI Design of a Pipeline Reed-Solomon Decoder Using Systolic Arrays", IEEE Transactions on Computers, vol.37, no.10, October 1988, pp.1273-1280

[3] Charles C. Wang, T. K. Truong, Howard M. Shao, Leslie J. Deutsch, Jim K. Omura, Irving S. Reed, "VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)", IEEE Transactions on Computers, vol.c-34, no.8, August 1985, pp.709-717

[4] Kuang Yung Liu, "Architecture for VLSI Design of Reed-Solomon Decoders", IEEE Transactions on Computers", vol.C-33, no.2, February 1984, pp.178-189

[5] 金勇煥, "Reed-Solomon 디코더의 VLSI 구현에 관한 연구", 工學碩士學位論文, 1991

[6] Richard E. Blahut, Theory and Practice of Error Control Codes, Addison Wesley, 1983

[7] 이만영, BCH 부호와 Reed-Solomon 부호, 민음사, 1990