論文98-35T-6-9

RS 코드를 이용한 복호기 설계

(A Decoder Design for High-Speed RS code)

박 화 세 *, 金 銀 源 **

(Hwa-Sea Park and Eun-Won Kim)

요 약

본 논문은 에러 정정 코드로서 가장 많이 사용하는 RS(Reed-Solomom)코드를 이용한 고속 복호기 설계에 관한 논문이며 VHDL을 사용하여 실행을 하였으며, 이 RS 복호기는 시간 영역 대신 변환 영역에서 설계 하였다. 변환 복호기는 구조의 단순성 때문에 VLSI칩 설계가 용이하며, 모든 설계에 대하여 systolic 배열을 적용하기 쉬운 파이프라인 아키텍춰를 사용하였다. 변환 RS 복호기는 고속 데이타 전송율을 갖는 복호기에 적합하여 FPGA 기술로 합성 한 후 복호율은 43MByte/s 보다 더 크고 범위는 1853 LCs(Logic Cell)을 갖는다. 파이프라인을 갖는 다른 아키텍춰와 비교하여 볼 때 이러한 결과는 다른 기술과 비교하여 우수한 기술이며, 에러 정정 능력과 파이프라인 성능은 컴퓨터 시뮬레이션을 통하여 검증하였다.

Abstract

In this paper, the high-speed decoder for RS(Reed-Solomon) code, one of the most popular error correcting code, is implemented using VHDL. This RS decoder is designed in transform domain instead of most time domain. Because of the simplicity in structure, transform decoder can be easily realized VLSI chip. Additionally the pipeline architecture, which is similar to a systolic array is applied for all design. Therefore, This transform RS decoder is suitable for high-rate data transfer. After synthesis with FPGA technology, the decoding rate is more 43 Mbytes/s and the area is 1853 LCs(Logic Cells). To compare with other product with pipeline architecture, this result is admirable. Error correcting ability and pipeline performance is certified by computer simulation.

I. 서 론

RS부호는 적은 redundancy로도 강력한 에러 정정 능력을 갖는 에러 정정 부호로서 비2원 형태이어서 산발에러뿐만 아니라 연집에러에도 강한 특성을 갖기 때문에 현재 군사통신, 무선통신, 위성통신, 데이터통신 시스템, 컴퓨터의 저장 시스템 등에 널리 이용되고 있다.

RS부호는 에러 정정 능력이 커질수록 하드웨어가 복잡해지고 그만큼 많은 지연이 발생하여 동작속도의 감소를 가져온다. 현재 쓰이는 고속의 메모리나 앞으로 널리 이용될 ATM, HDTV 등과 같은 고속 또는 광 대역을 요하는 통신망에 적용하기 위해서는 고속의 복호기 설계가 필수적이다.

본 논문에서는 RS 복호기를 시간영역이 아닌 변환 영역의 관점에서 설계하여 구조를 단순화시키고 전 과정을 파이프라인 처리하여 고속의 전송률을 실현하였다. 또한 normal basis를 이용 구조가 단순한 유한체 역원 계산기를 설계하여 나눗셈을 수행하기 위한 LUT(Look Up Table)을 제거하였다. 이렇게 하여 속도를 개선시킨 RS부호의 복호기를 VHDL을 이용하여 설계하고 이를 검증 및 합성하여 성능 및 에러 정정 능력을 검증하였으며 상대적으로 큰 면적을 기존의 복호기 이하로 줄일 수 있는 방법을 제시하였다.

* 正會員, 大林大學 電子通信科
(Dept. of Elec. Communication, Daelim College)
接受日字: 1998年4月16日, 수정완료일: 1998年6月16日

Ⅱ. 부호화 과정

유한체 GF(2^m)상에서 m개의 선형독립인 원소들의 집합을 GF(2^m)의 basis라 한다.

α를 차수가 m인 원시다항식 p(x)의 근이라고 할 때 $\{\alpha^0, \alpha^1, \alpha^2, \cdots, \alpha^{m-1}\}$은 GF($2^m$)의 basis이다. 이러한 basis를 GF(2^m)의 standard basis라 하는데 GF(2^m)상의 모든 원소는 이들 basis로 표현될 수 있다.

GF(2^m)상에서 $\{\alpha, \alpha^2, \alpha^4, \cdots, \alpha^{2^{m-1}}\}$이 선형독립이면 이들을 normal basis라 한다.
이러한 normal basis를 이용하면 구조가 단순한 파이프라인 역원계산기를 설계할 수 있다.

1. 제곱 및 제곱근 회로

GF(2^m)상에서 normal basis로 표현된 임의의 한 원소를 E라 하면

$$E = e_0\alpha + e_1\alpha^2 + \cdots + e_{m-1}\alpha^{2^{m-1}}$$

가 된다.

여기서 E의 제곱을 S라고 하면

$$S = E^2 = e_{m-1}\alpha + e_0\alpha^2 + e_1\alpha^4 + \cdots + e_{m-2}\alpha^{2^{m-1}}$$

가 되어 E를 오른쪽으로 한 번 순환 치환한 것이 된다. 따라서 GF(2^m)상에서 임의의 한 원소 E의 제곱 E^2은 다음 그림 1 과 같이 하드웨어로 구현할 수 있다.

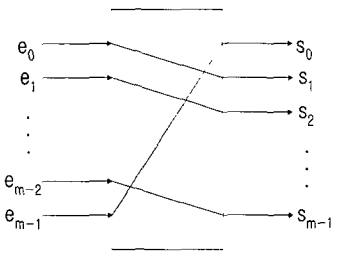

그림 1. Normal basis를 이용한 제곱근을 구하는 회로.
Fig. 1. Square root using normal basis.

2. 곱셈회로

GF(2^m)상에서 임의의 두 원소 X, Y를 normal basis로 표현하면 다음과 같다.

$$X = x_0\alpha + x_1\alpha^2 + \cdots + x_{m-1}\alpha^{2^{m-1}}$$

$$Y = y_0\alpha + y_1\alpha^2 + \cdots + y_{m-1}\alpha^{2^{m-1}}$$

X와 Y의 곱을 Z라 하면

$$Z = X \cdot Y = z_0\alpha + z_1\alpha^2 + \cdots + z_{m-1}\alpha^{2^{m-1}}$$

여기서 Z의 마지막 번째 계수 z_{m-1}은 X와 Y의 계수들로 구성된 2진 함수이므로 다음 식과 같이 나타낼 수 있다.

$$z_{m-1} = f(x_0, x_1, \cdots, x_{m-1}; y_0, y_1, \cdots, y_{m-1})$$

제곱기를 이용 계속 제곱해 나가면 Z의 모든 계수들을 같은 2진 함수로써 차례로 얻을 수 있다.

이것을 병렬구조의 하드웨어로 구현하면 그림 2 와 같다.

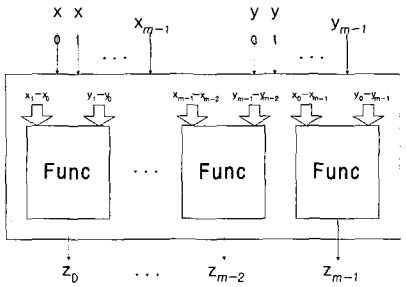

그림 2. normal basis를 이용한 병렬 곱셈기.
Fig. 2. Parallel multiplication circuit using normal basis.

3. 역원계산기

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

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

이를 하드웨어로 구현하면 그림 3과 같다.

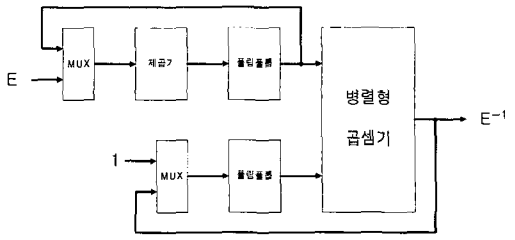

그림 3. normal basis를 이용한 역원계산기.
Fig. 3. Calculator using Normal basis.

Ⅲ. 복호화 과정

본 논문에서 설계한 RS 복호기는 변환영역의 관점에서 설계하였다. 수행 과정은 우선 신드롬(synd-

rome)을 계산하여 이로부터 에러위치다항식을 계산하고 이렇게 구한 신드롬과 에러위치다항식을 이용하여 에러 성분을 구하고 최종적으로 수신 부호 열에서 구해진 에러 성분을 빼줌으로써 에러를 정정하게 된다. 이러한 과정을 그림 4 에서 블록도로 나타내었다.

변환영역 상에서 복호화를 수행하기 위해서는 key equation을 계산하는 동안 신드롬을 지연시켜줄 필요가 있다.

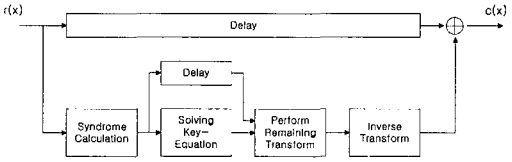

그림 4. 변환영역 상에서의 복호화 과정
Fig. 4. Process of decoding in transform domain.

이러한 변환영역에서의 복호 과정을 시간영역 상의 복호과정과 비교하기 위해 시간영역 상의 복호 과정의 블록도 그림 5 에 나타내었다.

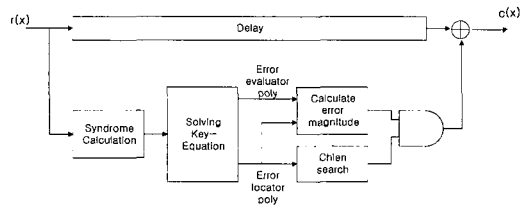

그림 5. 시간영역 상에서의 복호화 과정
Fig. 5. Process of decoding in time domain.

시간영역 상의 복호는 역 변환이 불필요하지만 에러위치다항식 외에 에러평가다항식을 따로 구해야 하고 에러위치다항식 $\sigma(x)$에 GF(2^m)상의 모든 근을 대입하는 Chien search라는 반복적인 과정을 거쳐 에러의 위치를 찾고 에러평가 다항식으로부터 에러 크기를 구해야만 최종적으로 에러다항식을 구할 수 있다. 또한 에러의 크기를 구하는 과정은 n 번의 나눗셈을 수행해야 한다. 따라서 변환영역 상의 부호화 과정에 비해 다소 더 복잡한 과정을 수행해야 함을 알 수 있다.

1. 유한체에서의 Fourier변환

Fourier변환을 복호화에 적용하려는 이유는 Fourier변환이 이미 여러 분야에 사용되고 있는 만큼 이론적 접근이 용이하며 변환영역에서의 복호기의 구조가 시간영역의 경우보다 간단해지기 때문이다.

$\bar{c}=(c_0,c_1,\cdots,c_{n-1})$을 GF(q)상의 원소를 성분으로 하는 n차원 벡터라 할 때, GF(q^m)상에서 $\bar{c}$의 변환벡터 $\bar{C}=(C_0,C_1,\cdots,C_{n-1})$로의 이산 Fourier 변환(DFT) 및 역 변환은 다음 식과 같다.

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

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

벡터 $\bar{c}$와 $\bar{C}$는 다항식

$$c(x)=c_0+c_1x+\cdots+c_{n-1}x^{n-1}$$

$$C(x)=C_0+C_1x+\cdots+C_{n-1}x^{n-1}$$

로 표현할 수 있다. 이와 같이 벡터의 다항식을 이용하면 C_j와 c_i에 대한 표현은 다음과 같이 바꿔 쓸 수 있다.

$$C_j=\sum_{i=0}^{n-1}c_i(\alpha^j)^i=c(\alpha^j), \qquad 0\le j\le n-1$$

$$c_i=\sum_{j=0}^{n-1}C_j(\alpha^{-i})^j=C(\alpha^{-i}), \qquad 0\le i\le n-1$$

즉 $C_j=0$이면 $c(x)$는 α^j을 근으로 하고 $c_i=0$이면 $C(x)$는 α^{-i}을 근으로 함을 알 수 있다. 벡터 $\bar{c}$의 성분 c_i가 다른 두 벡터 $\bar{g}$와 $\bar{f}$의 성분의 곱 g_id_i일 때 $\bar{c}$의 DFT $\bar{C}$는 다음과 같이 된다.

$$C_j=\sum_{k=0}^{n-1}G_kD_{j-k}=G*D, \qquad 0\le j\le n-1$$

따라서 시간 영역의 곱은 주파수 영역에서의 콘볼루션 합이 되며 이 관계의 역도 성립한다.

2. 신드롬 계산

t개의 오류를 정정할 수 있는 (n,k) RS부호에서 신드롬 $\bar{S}=(S_1,S_2,\cdots,S_{2t})$는 생성다항식 $g(x)$의 연속되는 2t개의 근을 수신다항식 $r(x)$에 대입한 다음 식으로 정의된다.

$$S_j=r(\alpha^j), \qquad 1\le j\le 2t$$

수신벡터 $\bar{r}$의 Fourier변환 $\bar{R}$은 앞의 식에 의해

$$R_j=\sum_{i=1}^{n-1}r_i(\alpha^j)^i=r(\alpha^j), \qquad 1\le j\le 2t$$

와 같으므로

$$S_j = R_j, \qquad 1 \le j \le 2t$$

의 관계를 가진다.

신드롬은 또한 에러벡터 $\bar{e}$의 변환 $\bar{E}$와도 밀접한 관계를 가진다.

유한체의 이산 Fourier변환 정의 식으로부터

$$\begin{aligned} R_j &= \sum_{i=0}^{n-1} \alpha^{ij} r_i = \sum_{i=0}^{n-1} \alpha^{ij}(c_i + e_i) \\ &= \sum_{i=0}^{n-1} \alpha^{ij} c_i + \sum_{i=0}^{n-1} \alpha^{ij} e_i = C_j + E_j \end{aligned}$$

RS부호의 정의에 의해 $j = 1,2,\cdots,2t$일 때 $C_j = c(\alpha^j) = 0$이 되므로

$$S_j = R_j = E_j, \qquad 1 \le j \le 2t$$

의 관계가 구해진다.

따라서 신드롬 S_j는 부호어와는 무관하고 에러다항식에 의해 결정되어지는 값이다. 그러므로 만약 에러가 발생하지 않았다면 신드롬의 값은 영이 된다.

위의 식들에 의해 (31,25) RS부호의 신드롬은 다음과 같이 구할 수 있고 그림 6 과 같이 병렬형태의 하드웨어로 구현할 수 있다.

$$\begin{aligned} S_j &= \sum_{i=0}^{30} \alpha^{ij} r_i, \qquad 1 \le j \le 6 \\ &= ((\cdots((r_{30}\alpha^j + r_{29})\alpha^j + r_{28})\alpha^j + \cdots)\alpha^j + r_0) \end{aligned}$$

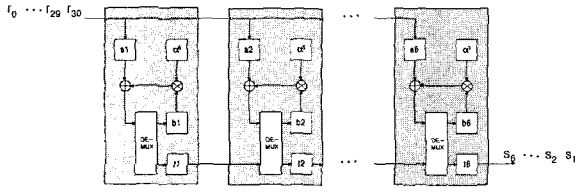

그림 6. 신드롬을 계산하기 위한 회로
Fig. 6. Circuit for syndrome calculation.

3. 에러위치다항식 계산

x^{2t}와 $S(x) = \sum_{j=1}^{2t} S_j x^{2t-j}$에 Euclid 알고리듬을 수행하면 다음의 다항식을 얻게 된다.

$$\lambda(x) = \lambda_0 x^t + \lambda_1 x^{t-1} + \cdots \lambda_t$$

이렇게 구한 다항식 $\lambda(x)$를 가장 높은 차수의 계수로 나눔으로써 에러위치 다항식을 얻게 된다.

$$\begin{aligned} \sigma(x) &= \lambda(x)/\lambda_0 \\ &= x^t + \sigma_1 x^{t-1} + \cdots + \sigma_t, \qquad \text{assuming } \lambda_0 \neq 0 \end{aligned}$$

그런데 일반적인 Euclid 알고리듬은 수행하는 과정에서 몫이 필요하기 때문에 각 단계에서 나눗셈이 필요하게 되고 이러한 나눗셈은 VLSI회로를 구현하는데 어려움이 따르게 된다.

아래에 설명하는 Euclid 알고리듬은 나눗셈이 없는 형태로 개선된 과정이다.

두개의 다항식을 가정해 보자

$$A(x) = x^{2t}$$
$$S(x) = \sum_{k=1}^{2t} S_k x^{2t-k}$$

개선된 Euclid 알고리듬은 i번째 나머지 $R_i(x)$와 $\gamma_i(x)$, $\gamma_i(x)A(x) + \lambda_i(x)S(x) = R_i(x)$ $\lambda_i(x)$값을 찾는 순환 과정이다.

$R_i(x)$의 차수가 t보다 작을 때 과정을 완료하게 된다. 알고리듬이 종료되었을 때 결과 $\lambda_i(x)$가 구하고자 하는 다항식 $\lambda(x)$가 된다.

이 알고리듬의 초기조건은

$$\begin{aligned} R_0(x) &= A(x) & Q_0(x) &= S(x) \\ \lambda_0(x) &= 0 & \mu_0(x) &= 1 \\ \gamma_0(x) &= 1 & \eta_0(x) &= 0 \end{aligned}$$

과 같고 다음의 과정을 반복하게 된다.

$$\begin{aligned} R_i(x) &= \sigma_{i-1}[b_{i-1}R_{i-1}(x) + a_{I-1}x^{|I_{i-1}|}Q_{i-1}(x)] \\ &\quad + \overline{\sigma_{i-1}}[R_{i-1}(x)] \\ Q_i(x) &= \sigma_{i-1}[Q_{i-1}(x)] + \\ &\quad \overline{\sigma_{i-1}}[b_{i-1}x^{|I_{i-1}|}R_{i-1}(x) + a_{I-1}Q_{i-1}(x)] \end{aligned}$$

$$\begin{aligned} \lambda_i(x) &= \sigma_{i-1}[b_{i-1}\lambda_{i-1}(x) + a_{I-1}x^{|I_{i-1}|}\mu_{i-1}(x)] \\ &\quad + \overline{\sigma_{i-1}}[\lambda_{i-1}(x)] \end{aligned}$$

$$\begin{aligned} \mu_i(x) &= \sigma_{i-1}[\mu_{i-1}(x)] + \\ &\quad \overline{\sigma_{i-1}}[b_{i-1}x^{|I_{i-1}|}\lambda_{i-1}(x) + a_{I-1}\mu_{i-1}(x)] \end{aligned}$$

a_{i-1}과 b_{i-1}은 각각 $R_{i-1}(x)$와 $Q_{i-1}(x)$의 최고 차수의 계수 값이며 $\sigma_{i-1}(x)$의 값은 다음 식에 의해 구

해진다.

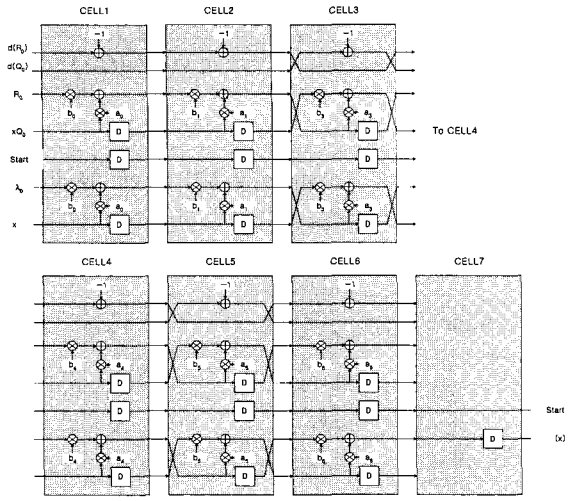

그림 7. Euclid 알고리듬 계산 회로
Fig. 7. Euclid algorithm calculation circuit.

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

$$\sigma_{i-1} = \begin{cases} 1, & l_{i-1} \geq 0 \text{ 일 때} \\ 0, & l_{i-1} < 0 \text{ 일 때} \end{cases}$$

순환은 $\deg(R_i(x)) < t$일 때 멈추게 된다.

이러한 개선된 알고리듬은 기본적인 Euclid 알고리듬과 같은 계산 결과를 가지며 매 단계마다 필요한 나눗셈 과정이 여기서는 필요가 없음을 수식으로 알 수가 있다.

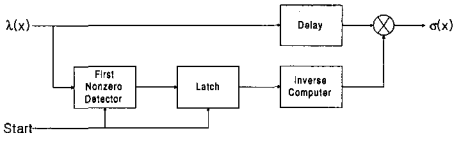

그림 8. $\sigma(x)$를 구하기 위한 회로
Fig. 8. Circuit for find the value of $\sigma(x)$.

(31,25) RS부호의 경우 Euclid 알고리듬은 그림 7과 같이 하드웨어로 구성할 수 있고 그림 8 과 같이 역원 계산기를 이용하여 에러 위치 다항식을 구 할 수 있다.

3. 변환영역 에러의 나머지 계산

$\nu \leq t$인 ν개의 오류가 $k = 1, 2, \cdots, \nu$인 i_k의 위치에 발생하였을 때 α^{-i_k}를 근으로 하는 변환영역의 오류 위치 다항식은 다음과 같다.

$$\sigma(x) = \prod_{k=1}^{\nu}(1 - x\alpha^{i_k})$$
$$= \sigma_0 + \sigma_1 x + \cdots + \sigma_\nu x^\nu$$

$\bar{\sigma}$의 역 변환을 $\bar{\lambda}$라 하면 $\sigma(\alpha^{-i_k}) = 0$이므로 $\lambda_{i_k} = 0$이다. 이것은 $\bar{e}$가 영이 아니면 $\bar{\lambda}$는 영임을 뜻하므로 $\bar{\lambda}$와 $\bar{e}$의 곱은 항상 영임을 의미한다. 따라서 $\bar{\sigma}$와 $\bar{E}$는 다음 식과 같이 표시할 수 있다.

$$\sum_{k=0}^{n-1} \sigma_k E_{j-k} = 0, \qquad 0 \leq j \leq n-1$$

또한, $\sigma(x)$의 정의식으로 부터 $\sigma_k = 0 (k > \nu)$, $\sigma_0 = 1$이므로 위의 식은 다음과 같이 나타내어진다.

$$E_j = \sum_{k=1}^{\nu} \sigma_k E_{j-k}, \qquad 0 \leq j \leq n-1$$

이 식으로부터 변환영역 에러의 일부인 신드롬 $E_1, E_2, \cdots, E_{2t}$와 Euclid 알고리듬을 이용하여 구한 에러정정다항식의 계수 $\sigma_1, \sigma_2, \cdots, \sigma_\nu$를 대입함으로써 에러의 나머지 $E_7, E_8, \cdots, E_{2^m-1}$를 구할 수 있게 된다. 예를 들어 $\sigma(x) = x^3 + \alpha^{22}x^2 + \alpha^{16}x + \alpha^{25}$일 때 변환영역 에러의 나머지를 구하는 회로는 그림 9 과 같다.

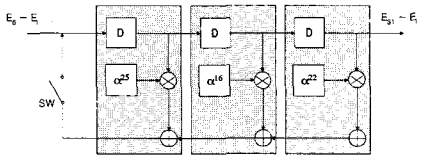

그림 9. 변환 에러 계산 회로
Fig. 9. Transform error calculate circuit.

4. 역 변환 및 에러 정정

에러 성분은 변환영역 에러를 역변환 함으로써 구할 수 있다.

$$e_n = \sum_{k=0}^{30} E_k \alpha^{-nk}, \quad 0 \leq n \leq 30$$
$$= (\cdots(((E_{30}\alpha^{-n} + E_{29})\alpha^{-n} + E_{28})\alpha^{-n} + \cdots + E_0)$$

이 식은 그림 10과 같이 회로로 구현될 수 있다.

이렇게 구한 에러다항식과 입력된 수신어와의 차를 구하면 원하는 부호어를 얻을 수 있다.

$$c(x) = r(x) - e(x)$$

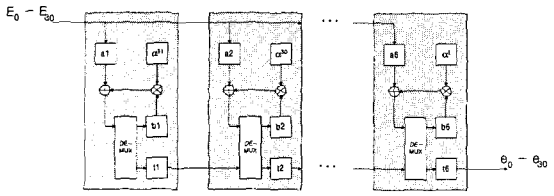

그림 10. 역 변환 계산 회로
Fig. 10. Inverse transform calculate circuit.

Ⅳ. VHDL 설계, 검증 및 고찰

전체 설계 방법은 Top-down 방식의 계층구조로 VHDL을 이용하여 설계하였다. 최상위 계층에서의 합성 결과의 Schematic도는 그림 11과 같고 전체적인 파이프라인 수행 과정을 그림 12에 나타내었다.

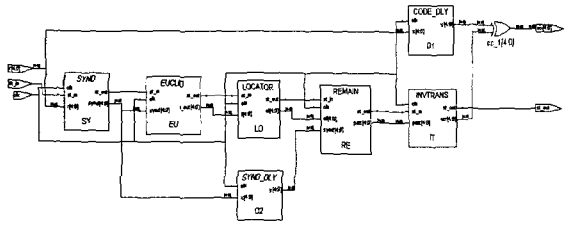

그림 11. TOP.VHD의 합성 schematic도
Fig. 11. Synthetic schematic diagram of TOP. VHD.

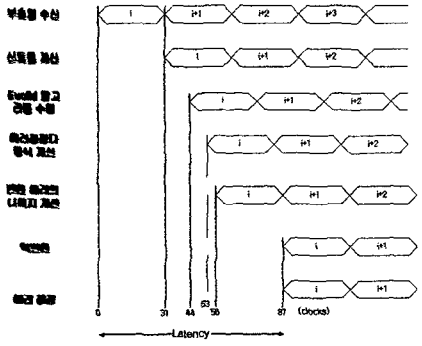

그림 12. 파이프라인 수행 과정
Fig. 12. Process of pipeline performance.

실제 반도체 칩으로 구현되었을 때의 성능을 확인하기 위하여 Altera FPGA(Field Programmable Gate Array) Tool인 MAX+Plus II를 이용 FLEX10K 공정에 직접 합성하여 보았으며 전체 복호기의 각 계층별 구성 및 합성 결과는 표 1과 같다.

표 1. VHDL 표현의 각 구성 요소 별 합성 결과
Table 1. Synthesis result with each structural element of VHDL expression.

계층	VHDL Description	f_{max} (Mhz)	Delay (ns)	Area (LCs)
LEVEL1	TOP.표 1. VHDL 표현의 각 구성 요소 별 합성 결과 Table 1. Synthesis result with each structural element of VHDL expression.VHD(RS 복호기)	43.10	17.0	1,858
LEVEL2	SYND.VHD(신드롬 계산)	75.18	12.9	99
	EUCLID.VHD (반복알고리듬 수행)	42.73	12.9	581
	LOCATOR.VHD (에러위치다항식 계산)	69.44	12.8	98
	REMAIN.VHD (변환영역 에러의 나머지 계산)	53.19	12.2	103
	INVTRANS.VHD(역변환)	52.91	12.9	454
	CODE_DLY.VHD(부호열 지연)	125.00	12.9	440
	SYND_DLY.VHD(신드롬 지연)	125.00	12.9	105
LEVEL3	INITIAL.VHD (EUCLID 알고리듬을 초기화)	96.15	13.0	28
	CELL1.VHD (EUCLID 알고리듬의 셀)	125.00	13.0	116
	CELL2.VHD (")	125.00	13.0	116
	CELL3.VHD (")	125.00	12.8	11
LEVEL4	INVERSE.VHD(역원 계산)	67.56	12.8	40
LEVEL5	MULT.VHD(곱셈기)		18.5	21
LEVEL6	FUNC.VHD (곱셈연산을 위한 논리함수)		17.4	4

표 2. 다른 제품과의 비교
Table 2. Comparison of another product.

회사 및 제품명	파이프라인구조	최대 전송률 (Symbol/sec)	Area(Gates)
Sharp	4-Stage	16 Mhz	
LG	Sequential	5 Mhz	
LSI Logic		40 Mhz	20,000(Memory 제외)
AHA4011		10 Mhz	
ARES		6.2 Mhz	
Mentor		44 Mhz	9530(Memory 제외)
본 논문의 변환 복호기	Sequential	43 Mhz 이상	15,000 ~ 20,000

최상의 계층에서 delay가 커지는데 그 것은 하위 계층간에 junction delay가 원인이며 이를 재 배치하여

최적화 하였을 경우 동작 주파수는 최대 77 Mhz까지 가능하다.

이를 실제 제품화된 파이프라인 구조를 가지는 복호기와 몇 가지 관점에서 상대적으로 성능을 표 2와 같이 비교해 보았다.

하드웨어의 크기는 부호 길이와 에러 정정 능력이 각각 다르므로 직접적인 비교를 할 수 없고 현재 주류를 이루는 시간영역에서 같은 부호 길이와 같은 에러 정정 능력을 갖는 복호기의 경우 systolic cell을 기준으로 크기를 표 3과 같이 비교하여 보았다.

표 3. RS 부호의 영역별 크기 비교
Table 3. Compare domain area size of RS code.

구 성 부	시간영역(단위 : Cell)	변환영역(단위 : Cell)
에러위치 계산	2t	불필요
에러의 크기 계산	2t	불필요
Look UP Table	n × m ROM	불필요
변환에러 계산	불필요	t
역변환	불필요	n
Euclid Algorithm	8t	시간영역의 1/2
신드롬 지연기	불필요	Latch로 대체

표에서와 같이 부호 길이가 짧은 RS 부호에서는 성능뿐만 아니라 크기에서도 이점이 있음 을 알 수 있다. 부호 길이가 긴 RS 부호는 시강영역에 비해 크기가 크다는 것을 알수 있다. 이러한 이유는 역변환 회로에 의한 것이며, 이러한 경우 시간영역복호에서의 Chien search를 이용하여 에러가 발생한 부분에서만 역 변환을 구함으로써 역변환 회로를 줄일 수 있어 전체적인 크기가 시간영역의 복호기 보다 더 작아진다.

그림 13은 제안한 시스템의 구조를 보여주고 있으며, 상대적인 크기를 표 4에서 보여주고 있다.

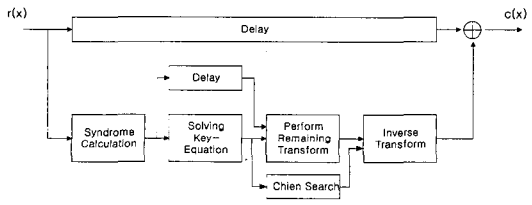

그림 13. 개선된 변환 복호기
Fig. 13. Improved transform decoder.

표 4. 개선된 변환영역 복호기의 상대적인 크기(공통되는 구조와 Memory 제외)
Table 4. Relative size of improved transform domain decoder.(Except command structure and memory)

영역(cell) / RS code	시간영역 복 호 기	변환영역 복호기	개선된 변환영역복호기
(31,25)	36	46	24
(31,15)	96	71	64
(255,247)	48	275	32

부호 길이에 관계없이 가장 작은 크기를 가짐을 알 수 있다.

에러 정정 능력과 파이프라인 성능을 확인하기 위하여 입력 파형으로 3개의 에러가 발생한 부호열을 연속적으로 입력을 넣어 시뮬레이션 하였으며 이를 이용 각 구성부 별로 검증을 하고 최종적으로 최상위계층에서 동작을 확인하였다. 검증 결과 그림 14 와 같다.

에러가 정정된 6비트의 부호열들이 한 클럭에 한 번씩 연속해서 출력으로 나옴을 확인하였다. 각각의 부호열의 시작에는 항상 start신호가 발생하고 이 신호를 기준으로 모든 파이프라인이 제어된다.

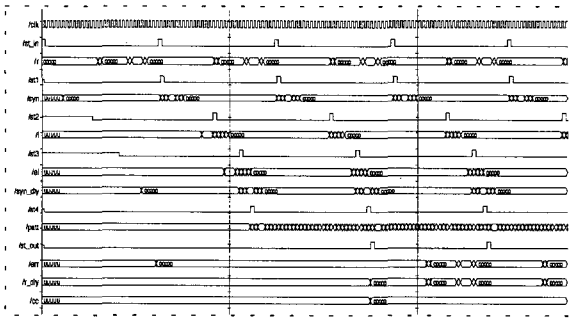

그림 14. Simulation 결과
Fig. 14. Simulation result.

VI. 결 론

본 논문에서는 주파수 영역의 관점에서 전체 구조를 파이프라인화하여 고속의 RS부호의 복호기를 설계하였고 만족할 만한 성능을 얻을 수 있었으며 에러 정정 능력 또한 검증을 하였다.

성능의 측면에서 보면 설계한 복호기가 43Mhz에서 동작하고 하위계층의 배치를 최적화 함으로써 최대 77Mhz에서 동작함을 확인하였다. 이는 최적화된 파이

프라인 구조를 가지는 기존 제품이 10~20 Mhz가 주류를 이루고 최근 제품이 40Mhz 부근에서 동작하므로 설계한 복호기가 상대적으로 우수함을 알 수 있다.

성능뿐만 아니라 저 비용, 저 전력, 신뢰성, 생산성 등을 위해서 하드웨어의 크기도 상당히 고려되어야 한다.

본 논문에서 설계한 변환 복호기는 부호 장이 짧은 RS 부호의 경우에는 기존의 복호기에 비해 더 작은 하드웨어 크기를 가지므로 성능과 크기에서 모두 만족할 수 있는 복호기가 가능하다.

그러나 부호 장이 긴 경우 역 변환회로에 의해 시간영역에 비해 크기가 상당히 커지게 된다. 따라서 부호 장에 관계없이 두 가지 측면을 모두 만족하기 위해서는 역 변환회로의 구조를 단순화시킬 필요가 있는데 이는 앞에서 제시한 개선된 구조를 이용함으로써 해결이 가능하다.

전체 구조에서 Euclid 알고리듬을 수행하는 부분은 상당히 큰 면적을 차지하는데 앞으로 이 부분을 줄이는 연구도 병행되어야 하겠다.

참 고 문 헌

[1] I. S. Hsu, T. K. Truong, L. J. Deutsch and I. S. Reed, "A Comparison of VLSI Architecture of Finite Field Multipliers Using Dual, Normal, or Standard Bases", IEEE Transactions on Computers, vol.37, no.6, June 1988, pp.735-739.

[2] 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.

[3] 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.

[4] Sterling R. Whitaker, John A. Canaris and Kelly B. Cameron, "Reed Solomon VLSI Codec for Advanced Television", IEEE Transactions on Circuits and Systems for Video Technology, vol.1, no.2, June 1991, pp.230-236.

[5] Tetsuo Iwaki, Toshihisa Tanaka, Eiji Yamada, Tohru Okuda, Taizoh Sasada, "Architecture of a High Speed Reed-Solomon Decoder", IEEE Transactions on Comsumer Electronics, vol.40, no.1, February 1994, pp.75-81.

[6] C. S. Yeh, Irving S. Reed, T. K. Truong, "Systolic Multipliers for Finite Fields GF (2m)", IEEE Transactions on Computers, vol.c-33, no.4, April 1984, pp.357-360.

[7] 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(2m)", IEEE Transactions on Computers, vol.c-34, no.8, August 1985, pp.709-717.

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

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

[10] Man Young Rhee, Error Correcting Coding Theory, McGRAW-HILL, 1989.

저 자 소 개

박 화 세(正會員)

1963년 2월 1일생. 1987년 경희대학교 전자공학과 졸업 공학사. 1989년 경희대학교 대학원 전자공학과 졸업 공학석사. 1997년~현재 한국 PC통신(주) 전략사업본부 기술위원. 1998~현재 용성전기(주) 부설연구소 책임 연구원. 1995년~현재 대림전문대 전자통신과 겸임교수 재직. 주관심분야는 네트워크, 통신시스템, VLSI설계

金 銀 源(正會員)

1962년 1월 20일생. 1985년 . 경희대학교 공과대학 전자공학과 졸업 공학사. 1990년 경희대학교 대학원 전자공학과 졸업 공학석사. 1998년 경희대학교 대학원 전자공학과 졸업공학박사. 1998년~현재 대림대학 전자통신과 전임강사. 주요관심분야는 B-ISDN 및 ATM, Digital 신호처리 및 Chip설계, 오류제어 이론