

論文99-36C-6-5

Reed-Solomon decoder를 위한 Two-way addressing 방식의 Euclid 계산용 회로설계

(Implementation of Euclidean Calculation Circuit with Two-Way Addressing Method for Reed-Solomon Decoder)

柳志濞*, 李丞峻**

(Jee-Ho Ryu and Seung-Jun Lee)

요 약

고성능 VLSI 설계를 위한 pipeline 형태의 Reed-Solomon을 구현하였다. Shortened RS code의 경우에 있어서 기존의 "parallel recursive cell" 방식이나^[1] "다중 클럭 설계"와 같은 접근과는 달리 작은 면적에서 단일 클럭으로 동작할 수 있는 "이중 주소(two-way addressing) 방식 Euclid 계산"을 제안하였다. 이러한 방식은 recursive cell을 병렬 처리하는 Euclid 계산 방식에 비해 면적이나 소비 전력에 있어 장점을 갖고 있음을 synthesis와 전력 모의실험을 통해 검증하였다. 본 설계는 면적상으로 parallel recursive cell을 이용한 단일 클럭 euclid 회로가 약 5,000 gate에 비하여 40% 정도 감소한 3,000 gate 정도에 구현할 수 있었다. 또한 전력 소비면으로는 기존의 recursive cell을 이용한 다중 클럭 euclid 회로가 6mW 이상의 전력을 소비하는 반면에 본 설계는 3mW대의 전력 소비를 보여 현격한 차이를 보였다.

Abstract

Two-way addressing method has been proposed for efficient VLSI implementation of Euclidean calculation circuit for pipelined Reed-Solomon decoder. This new circuit is operating with single clock while exploiting maximum parallelism, and uses register addressing instead of register shifting to minimize the switching power. Logic synthesis shows the circuit with the new scheme takes 3,000 logic gates, which is about 40% reduction from the previous 5,000 gate implementation. Computer simulation also shows the power consumption is about 3mW. The previous implementation with multiple clock consumed about 5mW.

I. 서 론

Reed-Solomon decoder 중에서 error location

* 正會員, 現代電子産業株式會社 시스템 IC 研究所
(Hyundai Electronics Industries, System IC R&D
Center)

** 正會員, 梨花女子大學校 電子工學科
(Dept. of Electronics Eng., Ewha Womans Univ.)
接受日字: 1998年10月19日, 수정완료일: 1999年5月28日

polynomial $\sigma(x)$ 와 error evaluation polynomial $\omega(x)$ 를 구하기 위한 알고리즘으로 Euclid 알고리즘이 매우 보편화 되었다.^[1, 2, 4] 이러한 Reed-Solomon decoder의 pipeline 구조와 VLSI 설계의 기초는 1985년 H. M. SHAO와 I. S. REED에 의해 그 기초가 이루어 졌다.^[2]

I. S. REED는 Kung에 의해 제안된 다항식의 GCD 계산을 위한 systolic array 아키텍처를^[3] 대폭 향상 시킴으로써 recursive cell의 병렬 처리 방식을 제안하였다.^[1] 이러한 recursive cell 구조는 일

반적인 RS code에는 매우 적합하며 현재까지도 VLSI design으로 채택되었다. 하지만 실제의 무선 통신에서 채용되는 RS code는 대부분 Shortened RS code를 채용하기 때문에 Euclid 계산 부분에 있어서 I. S. REED에 의해 제안된 병렬 처리 구조에서는 필연적으로 면적이 2배 혹은 3배가 되어야 하며 면적을 줄이기 위해 하나의 recursive cell을 사용하는 경우에는 다중 클락을 사용할 수 밖에 없다. 따라서 기존의 방식은 단일 클락을 사용하는 경우에는 면적이 증가하고 면적을 줄이기 위해서는 다중 클락을 사용할 수 밖에 없는 단점이 있었다. 예를 들어 QPSK나 VSB의 Euclid 계산부의 경우를 보면 모두 2개의 병렬 처리를 해야 하거나 2배속의 clock이 추가적으로 필요하다.

VLSI 설계에서 다중 클락을 사용하는 경우에는 초 미세 집적회로 설계로 갈수록 timing 분석이나 설계 변경에 많은 문제점을 일으킨다. 따라서 약간의 면적을 소비하더라도 단일 클락 설계로 개발하는 것은 소자의 안정성(stability)과 신뢰성(reliability) 같은 많은 잇점을 가져온다. 또한 기존의 Euclid 계산 방식은 shift register를 이용한 방식으로 전력의 소비가 많은 단점을 갖고 있었다. 그 이유는 CMOS 공정에서 전력 소비는 F/F의 toggling 의 회수에 비례적으로 증가하며 shift register는 클락마다 register 전체의 값이 toggling 되는 특성을 갖고 있으므로 저전력 회로를 구성하기 위해서는 좋지 않은 구조이다. 따라서 이러한 구조를 주소 방식으로 전환하면 전력 소비도 최소화 할 수 있다.

II. Reed-Solomon decoding 이론 및 recursive cell 방식의 Euclid 계산법 [6, 7]

n = 2^m - 1일 때, block length = N, message length = I인 (N, I) RS code를 가정하자. (2t = n - k) 입력을 받은 n개의 신호(r₀, r₁, r₂ ... r_{n-1})을 입력 다항식으로 표현하면 다음과 같다.

$$R(x) = \sum_{i=0}^{n-1} r_i x^i = r_0 + r_1 x + r_2 x^2 + \dots + r_{n-1} x^{n-1}$$

입력된 신호에 의해 발생하는 syndrome은 다음과 같다.

$$S_k = \sum_{i=0}^{n-1} r_i x^i \Big|_{x=\alpha^k} \\ = \sum_{i=0}^{n-1} r_i \alpha^{ki} \quad (1 \leq k \leq 2t)$$

syndrome 다항식은 다음과 같다.

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

예측 가능한 에러(erasure 다항식 = 1) 가 없다고 하면 Euclid 계산부에 입력되는 초기값은 다음과 같다.

$$\mu_0(x) = 1$$

$$\lambda_0(x) = 0$$

$$Q_0(x) = S(x)$$

$$R_0(x) = x^{2t}$$

$\lambda_i(x)$ 의 차수가 $R_i(x)$ 의 차수보다 크게 될 때까지 다음의 iteration을 반복한다. (Euclid polynomial 계산)

$$R_i(x) = [\sigma_{i-1} b_{i-1} R_{i-1}(x) + \bar{\sigma}_{i-1} a_{i-1} Q_{i-1}(x)] \\ - x^{l_{i-1}} [\sigma_{i-1} a_{i-1} Q_{i-1}(x) + \bar{\sigma}_{i-1} b_{i-1} R_{i-1}(x)]$$

$$\lambda_i(x) = [\sigma_{i-1} b_{i-1} \lambda_{i-1}(x) + \bar{\sigma}_{i-1} a_{i-1} \mu_{i-1}(x)] \\ - x^{l_{i-1}} [\sigma_{i-1} a_{i-1} \mu_{i-1}(x) + \bar{\sigma}_{i-1} b_{i-1} \lambda_{i-1}(x)]$$

$$Q_i(x) = \sigma_{i-1} Q_{i-1}(x) + \bar{\sigma}_{i-1} R_{i-1}(x)$$

$$\mu_i(x) = \sigma_{i-1} \mu_{i-1}(x) + \bar{\sigma}_{i-1} \lambda_{i-1}(x)$$

여기서 a_{i-1} 과 b_{i-1} 은 $R_{i-1}(x)$ 와 $Q_{i-1}(x)$ 의 최고차항 계수이며 l_{i-1} 과 σ_{i-1} 은 다음과 같이 정의된다.

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

$$\sigma_{i-1} = 1 \text{ if } l_{i-1} \geq 0$$

$$\sigma_{i-1} = 0 \text{ if } l_{i-1} < 0$$

위의 같은 iteration은 $\deg(\lambda_i(x)) > \deg(R_i(x))$ 이면 멈추어 다음과 같이 error-location 다항식과 error-evaluation 다항식을 구한다.

$$\sigma(x) = \lambda_i(x)$$

$$\omega(x) = R_i(x)$$

$\sigma(X)$ 는 error-location polynomial로서 error의 위치를 찾기 위해 사용된다. 즉, x 에 α^{-i} , $i=0..n-1$ 의 값을 넣어 $\sigma(\alpha^{-i})=0$ 인 i 를 찾아 error의 위치를 찾아낸다. 이렇게 계산된 위치의 error의 값은 다음과 같이 계산된다.

$$e_i = -\frac{\omega(\alpha^{-i})}{\sigma'(\alpha^{-i})} \quad 0 \leq i \leq n-1$$

III. Pipeline 구조의 Reed-Solomon decoder

I. S. REED에 의해 제안된 pipeline RS decoder는 간단히 보아 다음과 같이 구성되어 있다.

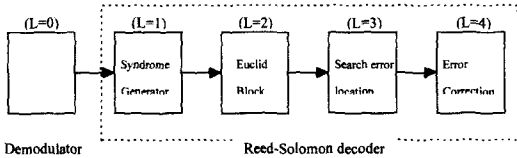


그림 1. Pipeline 구조의 Reed-Solomon decoder
Fig. 1. Pipeline Architecture of Reed-Solomon Decoder.

그림과 같이 Reed-Solomon decoder는 크게 4개의 block으로 구성되어 있다. 이 중에서 throughput의 차이에 의해 병렬처리가 필요하거나 다중 클락이 필요한 부분은 Euclid 부이다.

RS 부의 level에 따라 하나의 codeword를 처리하기 위한 symbol clock은 다음과 같다.

- (L=0) : N symbol times
- (L=1) : N symbol times
- (L=2) : $(N-I)^2 = (2t)^2 = 4t^2$ symbol times

Syndrome 발생기(L=1)는 길이 N의 codeword를 처리하여 길이 N-I의 syndrome을 발생시킨다. Euclid 계산부(L=2)은 이러한 N-I 개의 Syndrome을 받아 한 codeword 주기안에 계산을 종료하여야 한다. 만약에 그렇지 못한다면 Syndrome 발생기에서 발생되는 다음의 syndrome을 읽을 수 없기 때문이다. Euclid 계산부의 경우 N-I차 다항식의 유클리드 호제법 계산 방식을 따르기 때문에 차수 하나를 줄이기 위해 (N-I)번의 계산이 필요하므로 가장 worst한 경우 (N-I)번씩 (N-I)번 즉, $(N-I)^2$ 의 symbol time이 필

요하게 된다.

따라서 필요한 recursive cell은 syndrome 발생부와 Euclid 계산부간의 throughput ratio $(= \frac{(N-I)^2}{N})$ 에 의해 결정된다. 예를 들어 QPSK의 경우, (204, 188) RS code를 사용하므로 $(1 < \text{throughput ratio} < 2)$ 를 만족하여 throughput ratio를 1 이하로 만들기 위해서는 이중 병렬 처리 구조를 갖게 되며, VSB의 경우, (207, 187)의 RS code를 사용하므로 마찬가지로 2개의 recursive cell의 병렬구조가 필요하다. 그러나 실제의 응용에서는 이러한 면적의 낭비를 줄이기 위하여 Euclid block에 2배속의 클락을 넣은 다중 클락 설계를 한다.

IV. 향상된 Euclid 계산부

본 논문에서 제시하는 Euclid 계산부는 다음과 같은 특징을 갖고 있으며 “이중 주소 방식 Euclid 부”이라 명명한다.

- (1) Shift register 방식 대신에 주소방식을 채택하여 F/F toggling 시에 발생하는 전력 소비를 최소화 하였다.
- (2) 병렬 주소 방식을 택하여 하나의 code word를 처리하는데 필요한 symbol time을 $(N-I)^2$ 에서 $[(N-I)/2]^2$ 으로 줄임으로써 병렬 처리를 위한 mux/demux 및 추가적인 Euclid부를 줄일 수 있다.
- (3) 유효한 $\sigma(X)$ 와 $\omega(X)$ 가 구해진 다음에는 더 이상의 recursive 계산을 수행하지 않아 toggling을 최소화 하였다.

A. 전체 시스템 구성

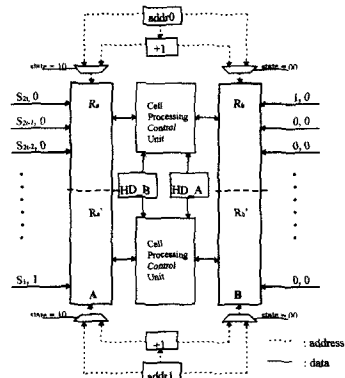


그림 2. 이중 주소 방식의 Euclid 계산부
Fig. 2. Euclidean ALU with Two-Way Addressing

그림 -2는 이중 주소 방식 Euclid를 나타내는 것이다. 다항식 계산을 위한 register로 2개(A와 B)가 존재하며 각각 Ra, Ra'와 Rb, Rb'으로 나뉘어 병렬 계산 된다. 따라서 상부의 주소와 하부의 주소가 독립적으로 존재하며 데이터 처리를 위해서도 병렬 처리를 위해 2개의 제어부 존재한다. 이러한 제어부는 state에 따라 data의 흐름을 제어하기 위한 것으로 내부 구조는 그림-3과 같다.

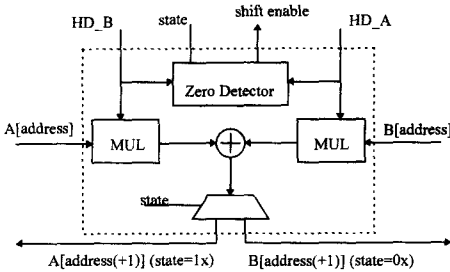


그림 3. 데이터 처리 제어부
Fig. 3. Control Block.

B. 초기화

초기화 시에는 주 register인 A와 B 및 관련 파라미터 정보를 초기화 하며 다음과 같이 정리할 수 있다.

$$\begin{aligned}
 A &= (A[0], A[1], \dots, A[2t]) \\
 &= Q_0(x) (S_1, S_2, \dots, S_{2t}) \text{ for } \omega(x) \\
 &= (1, 0, 0, \dots, 0) \text{ for } \sigma(x) \\
 B &= (B[0], B[1], \dots, B[2t]) \\
 &= R_0(x) (0, \dots, 0, 0, 1) \text{ for } \omega(x) \\
 &= (0, 0, 0, \dots, 0) \text{ for } \sigma(x) \\
 \text{deg}_A &= 2t-1 \text{ (} Q_0(x) \text{의 최고 차수)} \\
 \text{deg}_B &= 2t \text{ (} R_0(x) \text{의 최고 차수)} \\
 \text{HD}_A &= S_{2t} \text{ (} Q_0(x) \text{의 최고항 계수)} \\
 \text{HD}_B &= 1 \text{ (} R_0(x) \text{의 최고항 계수)} \\
 \text{addr0} &= 2t-1 \text{ (상위 register의 address)} \\
 \text{addr1} &= t-1 \text{ (하위 register의 address)} \\
 \text{state} &= 00
 \end{aligned}$$

C. 이중 주소 방식에 의한 다항식 계산

클락마다 주소는 하나씩 감소하고, 다항식 계산이 끝날 때(addr0 = t, addr1 = 0)마다 다시 초기화 된다. state는 한 번의 다항식 계산이 끝날 때마다 (00, 01, 10, 11) 의 순서대로 싸이클을 돈다. 그림 -2에서

와 같이 state = 10일 경우에는 register A에 address+1 을 넣어주며, state = 00인 경우에는 register B에 address+1을 넣어 준다. 그림 -3에서와 같이 state=1x의 경우에는 register B의 다항식의 차수를 하나 낮추고, state = 0x인 경우에는 register A의 다항식 차수를 하나 낮춘다. 단 낮추고자 하는 다항식의 최고차 항의 계수(HD_A(B))의 값이 0(zero)인 경우에는 모든 bit을 하나씩 register의 상위로 shift한다. 이러한 제어를 위해 그림 -3에서와 같이 “Zero Detector”를 둔다. 또한 다항식 A의 차수가 줄어든 경우에는 deg_A의 값을 다항식 B의 차수가 줄어든 경우에는 deg_B의 값을 하나 줄이며 HD_A와 HD_B는 $\omega(x)$ 를 구하기 위한 다항식의 최고차 항의 계수로 바꾼다. state에 따른 다항식 계산 과정은 다음과 같다.

● shift enable = 0

[Ra / Rb]

state (00) :

$$\begin{aligned}
 B[addr0+1] &\leq \text{HD}_A \times B[addr0] \\
 &\quad + \text{HD}_B \times A[addr0] \\
 (\text{addr0} = t \text{인 경우, } B[t] &\leq \text{temp})
 \end{aligned}$$

state (01) :

$$\begin{aligned}
 B[addr0] &\leq \text{HD}_A \times B[addr0] \\
 &\quad + \text{HD}_B \times A[addr0]
 \end{aligned}$$

state (10) :

$$\begin{aligned}
 A[addr0+1] &\leq \text{HD}_A \times B[addr0] \\
 &\quad + \text{HD}_B \times A[addr0] \\
 (\text{addr0} = t \text{인 경우, } A[t] &\leq \text{temp})
 \end{aligned}$$

state (11) :

$$\begin{aligned}
 A[addr0] &\leq \text{HD}_A \times B[addr0] \\
 &\quad + \text{HD}_B \times A[addr0]
 \end{aligned}$$

[Ra' / Rb']

state (00) : addr1 = t-1 이 아닌 경우,

$$\begin{aligned}
 B[addr1+1] &\leq \text{HD}_A \times B[addr1] \\
 &\quad + \text{HD}_B \times A[addr1]
 \end{aligned}$$

addr1 = t-1인 경우,

$$\text{temp} \leq \text{HD}_A \times B[addr1] + \text{HD}_B \times A[addr1]$$

state (01) :

$$\begin{aligned}
 B[addr1] &\leq \text{HD}_A \times B[addr1] \\
 &\quad + \text{HD}_B \times A[addr1]
 \end{aligned}$$

state (10) : addr1=t-1이 아닌 경우,
 $A [addr1+1] \leq HD_A \times B [addr1]$
 $+ HD_B \times A [addr1]$
 addr1=t-1인 경우,
 $temp \leq HD_A \times B [addr1] + HD_B \times A [addr1]$
 state (11) :
 $A [addr1] \leq HD_A \times B [addr1]$
 $+ HD_B \times A [addr1]$

● shift enable = 1

[Ra / Rb]

state(0x) : B [addr0+1] <= B [addr0]
 (addr0 = t인 경우, B [t] <= temp)
 state(1x) : A [addr0+1] <= A [addr0]
 (addr0 = t인 경우, A [t] <= temp)

[Ra' / Rb']

state (0x) : addr1 = t-1이 아닌 경우,
 $temp \leq B [addr1]$
 addr1 = t-1인 경우,
 $B [addr1+1] \leq B [addr1]$
 state (1x) : addr1 = t-1 이 아닌 경우,
 $temp \leq A [addr1]$
 addr1 = t-1 인 경우,
 $A [addr1+1] \leq A [addr1]$

D. error-location 다항식 ($\sigma(x)$), error-evaluation 다항식 ($\omega(x)$) 계산

전술한 바와 같이 다항식의 차수가 하나 줄어들 때 마다 deg_A나 deg_B의 값이 줄어든다. 이러한 iteration은 deg_A나 deg_B의 값이 t(에러 정정수)와 같을 경우에 iteration을 멈추어 다음과 같이 구한다. $\omega(x)$ 와 $\sigma(x)$ 를 구하는데 서로 다른 점은 초기화를 제외하고 모두 동일하다.

$$\omega(x) = \begin{cases} A[2t-2]x^{t-1} + A[2t-3]x^{t-2} + \dots + A[t-1] & (\text{deg}_A = t) \\ \sigma(x) + A[2t-2]x^{t-1} + A[2t-3]x^{t-2} + \dots + A[t-1] & (\text{deg}_B = t) \end{cases}$$

$$\sigma(x) = \begin{cases} A[2t-1]x^t + A[2t-2]x^{t-1} + \dots + A[t-1] & (\text{deg}_A = t) \\ \sigma(x) + A[2t-1]x^t + A[2t-2]x^{t-1} + \dots + A[t-1] & (\text{deg}_B = t) \end{cases}$$

V. 결과

본 연구에서는 QPSK용 RS에서 다음과 같은 두 Euclid 계산부 간의 면적, 전력 소비 등을 상호 비교하였다. 참고적으로 I. S. REED가 제안한 병렬 처리 cell 방식은 면적이 매우 크므로 본 연구의 비교대상에서 제외하였다.

Sample 1 : 하나의 symbol clock과 Euclid block을 위한 2배속의 euclid clock으로 구성된 다중 클락 설계로서 I. S. REED가 제안한 recursive cell을 하나 채용하였다.

Sample 2 : symbol clock 하나만을 사용한 단일 클락 설계로서 이중 주소 방식 Euclid부를 채용하였다.

A. 면적

면적을 상호 비교하기 위해 이미 simulation을 통해 검증된 RTL code를 synthesis하여 추출되는 gate netlist의 gate count에 의해 대략적인 평가를 하였다. Gate count에 의하면 recursive cell 방식에 비해 이중 주소 방식을 사용한 경우, 약 5%의 면적 증가율을 얻을 수 있었다. 이것은 연속성을 이용한 것에 비해 addressing을 해야 하는 제어 신호 발생에 필요한 최소한의 로직에 의한 것으로 해석된다. 이에 따르면 단일 클락을 사용하는 I. S. REED의 mux/demux multiple recursive cell 구조와 비교하여서는 약 50%의 면적 절감 효과를 가져옴을 의미한다. 왜냐하면 두개의 recursive cell을 사용하여야 하기 때문이다. 따라서 single clock을 채용하는 Reed-Solomon block용 Euclid 계산을 위해 이중 주소 방식은 면적 절감 효과가 뛰어나다. 실제적으로 이중 주소 방식을 통한 단일 클락 설계는 약 3,000 gate 정도였으며 다중 클락 recursive cell을 사용하는 경우 2,800 gate였다. 따라서 단일 클락 병렬 recursive cell을 이용한 설계의 경우 5,000 gate 이상으로 추산되며 이는 본 설계가 약 40 % 정도의 면적 절감효과가 있음을 알 수 있다. 즉, 면적에 있어서 는 다음과 같은 상호 관계를 갖는다.

Recursive cell을 이용한 단일 클락 방식 >> (예상 40%) 이중 주소 방식 > (5%) recursive cell을 이용한 다중 클락 방식

B. 전력 소비

전력 소비에 대한 평가는 Gate level simulation동

안 gate 의 toggling 수를 계산하여 대략의 power의 값을 측정하는 simulation tool(EPIC : powermill)을 이용하여 이루어졌다. Simulation은 에러의 수를 1, 3, 8로 변화 시켜 가며 측정하였다. 에러가 존재하지 않는 경우에는 Euclid block이 동작하지 않으므로 전력 소비 측정에서 제외하였다. 측정 결과는 다음과 같다.

error 수	전력 소비 (mW)	
	Sample 1	Sample 2
1	6.4	3.5
3	6.6	3.6
8	6.9	3.8

결과에서와 같이 이중 주소 방식에 의한 euclid 계산의 경우 전력 소모량이 매우 적었다. 그 이유는 recursive cell의 경우 shift register를 이용하기 때문에 clock 마다 전력 소비가 크고 고속의 clock으로 여러 번 계산을 하는 반면에 이중 주소 방식의 경우는 address 방식을 통해 필요한 부분만을 읽어 계산하기 때문으로 분석된다. 즉, 전력 소비면으로 보면 다음과 같은 상호 관계를 갖는다.

Recursive cell을 이용한 단일 클락 방식 >> (예상 100 %) recursive cell을 이용한 다중 클락 방식 >> (90%) 이중 주소 방식

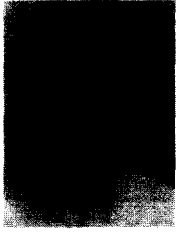
VI. 결 론

VLSI technology가 미세화 됨에 따라 설계의 방향도 무조건적인 면적 감소보다는 소자의 안정성(stability)와 신뢰성(reliability)이 무엇보다 중요하다. 이에 따라 본 연구에서는 면적과 power 소비 측면에서도 장점이 있으며 single clock을 채용한 design을 제시하게 되었다. 이것은 I. S. REED에 의해 제안된 pipeline 구조의 Reed-Solomon block에서 Euclid block을 대폭 수정함으로써 이루어졌다. 결과에서도 알 수 있듯이 거의 모든 방면에서 진보된 architecture를 제시하였다.

참 고 문 헌

- [1] H. M. SHAO, I. S. REED, "On the VLSI Design of a Pipeline Reed-Solomon Decoder Using Systolic Arrays", IEEE Trans. Comput., vol. 37-10, Oct. 1988.
- [2] 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, May 1985.
- [3] R. P. Brent and H. T. Kung, "Systolic VLSI arrays for polynomial GCD computations", Dep. Comput. Sci., Carnegie-Melon Univ., Pittsburgh, PA, Pep., 1982.
- [4] H. M. SHAO, T. K. Truong, J. S. Hsu, L. J. Deutsch, and I. S. REED, "A single chip VLSI Reed-Solomon decoder", Jet Propulsion Lab. TDA Progress Rep. 42-84, Oct.-Dec. 1985.
- [5] I. S. REED, T. K. Truong, and R. L. Miller, "Decoding of BCH and RS codes with errors and erasures using continued fractions", Electron. Lett., July 1979.
- [6] S. LIN, D. J. COSTELLO Jr., Error Control Coding. Prentice-Hall, 1983.
- [7] R. J. McEliece, The Theory of Information and Coding. Reading, MA : Addison-Wesley, 1977.

저 자 소 개



李 丞 暎(正會員)

1986년 서울대학교 전자공학과 학사. 1989년 University of California, Berkeley 전자공학과 석사. 1993년 University of California, Berkeley 전자공학과 박사. 1992년 1월 ~ 1998년 10월

현대전자 시스템IC 연구소 1999년 3월 ~ 현재 이화여자대학교 전자공학과 조교수. 주관심분야는 통신용 반도체 설계, CAD



柳 志 游(正會員)

1994년 고려대학교 전기공학과 학사. 1996년 동대학원 졸업 석사. 1996년 ~ 현재 현대전자 시스템IC 연구소 근무. 주관심분야는 통신 알고리즘 및 반도체 소자 설계