논문 2012-50-3-8

연판정 Reed-Solomon 리스트 디코딩을 위한 저복잡도 Interpolation 구조

(Area-efficient Interpolation Architecture for Soft-Decision List Decoding of Reed-Solomon Codes)

이 성 만*, 박 태 근**

(Sungman Lee and Taegeun Park)

요 약

Reed-Solomon(RS) 코드는 강력한 에러 정정 능력으로 널리 사용된다. 최근 제안된 RS 코드의 리스트 디코딩 알고리즘은 일반적인 디코더보다 더 큰 디코딩 반경을 가지며 하나 이상의 코드를 찾아낸다. 리스트 디코더는 복잡도가 매우 큰 Interpolation 단계를 포함하며 효율적인 하드웨어 설계가 필요하다. 본 논문에서는 연판정 RS 리스트 디코딩 알고리즘을 위한 효율적인 저복잡도 Interpolation 구조를 제안한다. 제안된 구조는 후보다항식의 Y 차수에 대해서는 병렬로 처리하며 X 차수에 대해서는 직렬로 처리한다. 후보다항식의 처리순서는 계수의 메모리사용의 효율성을 높이기 위하여 적응적으로 결정한다. 따라서 내부 저장공간이 최소화되며 메모리 구조와 접근이 단순해진다. 또한 제안된 구조는 각 모듈의 레이턴시가 유사하고 모듈간 스케쥴링을 최대한 중첩함으로써 높은 하드웨어 효율을 보여준다. 예제로써 (255, 239) RS 리스트 디코더를 설계하였으며 동부하이텍 0.18μm 표준 셀 라이브러리를 사용하여 합성하여 검증되었고 결과 최대 동작 주파수는 200MHz이고 게이트 수는 25.1K이다.

Abstract

Reed-Solomon (RS) codes are powerful error-correcting codes used in diverse applications. Recently, algebraic soft-decision decoding algorithm for RS codes that can correct the errors beyond the error correcting bound has been proposed. The algorithm requires very intensive computations for interpolation, therefore an efficient VLSI architecture, which is realizable in hardware with a moderate hardware complexity, is mandatory for various applications. In this paper, we propose an efficient architecture with low hardware complexity for interpolation in soft-decision list decoding of Reed-Solomon codes. The proposed architecture processes the candidate polynomial in such a way that the terms of X degrees are processed in serial and the terms of Y degrees are processed in parallel. The processing order of candidate polynomials adaptively changes to increase the efficiency of memory access for coefficients; this minimizes the internal registers and the number of memory accesses and simplifies the memory structure by combining and storing data in memory. Also, the proposed architecture shows high hardware efficiency, since each module is balanced in terms of latency and the modules are maximally overlapped in schedule. The proposed interpolation architecture for the (255, 239) RS list decoder is designed and synthesized using the DongbuHitek 0.18μm standard cell library, the number of gate counts is 25.1K and the maximum operating frequency is 200 MHz.

Keywords : Reed-Solomon codes, soft-decision list decoder, interpolation, VLSI architecture

* 정회원, LG전자, ** 정회원, 가톨릭대학교 정보통신전자공학부
(Department of Information, Communication, and Electronic Engineering, The Catholic University of Korea)
※ 이 논문은 2008년도 정부재원(교육인적자원부 학술연구조성사업비)으로 한국학술진흥재단의 지원을 받아 연구되었음(KRF-2008-313-D00743).
※ 본 연구는 2011년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음(M2011-B0002-00057).
접수일자: 2012년8월1일, 수정완료일: 2013년2월27일

Ⅰ. 서 론

Reed-Solomon(RS) 코드는 에러정정코드로써, 데이터 저장, 유, 무선 통신, 위성 통신 등 다양한 어플리케이션에 사용된다. 이러한 광범위한 사용은 RS 코드가 적은 수의 패리티만으로도 뛰어난 에러정정 능력을 보이며, Berlekamp-Massey 알고리즘, 유클리드 알고리즘[1]과 같은 효율적인 디코딩 알고리즘이 연구되어 왔기 때문이다. RS 코드는 k 개의 메시지 심볼에 $n-k$ 개의 패리티 심볼을 덧붙여 n 개의 코드워드 심볼을 만든다. RS 코드워드는 일반적으로 $GF(2^q)$의 원소인 q 비트 심볼로 이루어지고 하나의 심볼의 정정은 q비트의 정정의 효과를 보이므로 연집에러정정에 적합하다.

경판정을 통해 심볼로 변환하여 복호하는 전통적인 BMD(Bounded Minimum Distance) 디코더의 디코딩 반경은 $t = \lfloor d_{\min}/2 \rfloor$ 을 갖는데, 여기서 $d_{\min} = (n-k+1)$이고, 코드워드 간의 최소거리를 뜻한다. 따라서 BMD 디코더는 이러한 디코딩 반경 내의 유일한 코드워드 한 개를 찾는다. 최근, Sudan[2]과 Guruswami and Sudan (GS)[3]은 다항시간 복잡도를 갖는 리스트 디코딩 알고리즘을 제안하였으며 하나 이상의 코드워드 리스트를 찾고 이중 가장 확률이 큰 코드워드를 고른다. Koetter and Vardy(KV)는 채널의 확률을 정수로 변환하는 알고리즘을 제안함으로써 GS 리스트 디코더를 확장한 연판정 리스트 디코딩이 가능해졌으며[4] 연판정 디코더는 경판정 디코더에 비해 256-QAM 변조방식을 사용하여 Gaussian 채널에서 $0.25 \sim 1.5$dB, Rayleigh 채널에서 $2 \sim 9$dB의 코딩이득을 얻을 수 있음이 알려져 있다[5~6].

RS 리스트 디코더를 위한 Interpolation 알고리즘으로 Nielson[7]과 Lee-O'Sullivan[8]의 알고리즘이 있다. Lee-O'Sullivan의 알고리즘은 multiplicity가 작을 때 매우 효율적이지만 Interpolation 과정이 시작되면 포인트와 multiplicity를 변화시킬 수 없다. 반면, Nielson의 알고리즘은 포인트와 multiplicity의 변화를 적용하기 쉬우므로 연판정 리스트 디코더에 적합하다. Interpolation은 매우 많은 반복 계산 과정을 포함하고 있어 레이턴시가 매우 큰데, 리인코딩(re-encoding) 기법을 사용하여 반복 계산과정을 알고리즘 차원에서 줄일 수 있다[9]. 리인코딩은 확률이 높은 Interpolation 포인트에 대한 제한조건을 만족시키는 전처리 과정을 통해 Interpolation의 복잡도를 효율적으로 줄여준다.

Interpolation은 리스트 디코더에서 반복 계산량이 가장 많은 부분으로 효율적인 하드웨어의 구현이 필요하다. 몇몇 선행연구에서 Interpolation의 구조에 대한 연구가 이루어졌다[10~12]. Point serial 알고리즘[10]은 하나의 multiplicity에 해당하는 제한조건을 미리 예측하여 계산하기 때문에 일반적인 constraint serial 알고리즘에 비해 높은 처리량을 보였지만, 그에 따른 추가 하드웨어가 들고, 각 모듈간의 중첩이 고르게 이뤄지지 않아 하드웨어 효율이 낮다. 논문 [11]은 심볼의 곱셈이 지수에서의 덧셈이라는 점을 이용하여 심볼의 표기를 변환하는 방법을 사용하여 다단 파이프라인의 적용을 가능케 하였고, 다항식과 Y의 차수에 따라 병렬처리를 하여 높은 처리량을 얻을 수 있었다. 하지만 심볼의 표기를 변환하기 위한 LUT와 빠른 덧셈을 위한 CLA가 필요하며, 유한체의 크기가 클수록 이러한 추가 하드웨어 비용은 커진다. 논문 [12]는 다항식의 X 차수가 높은 점을 이용하여 X에 따라 병렬처리를 함으로써 처리량을 높였지만, 그에 따른 하드웨어 비용이 매우 크다. 또한, X의 차수가 천천히 증가하기 때문에 초기 유휴 하드웨어가 생겨 하드웨어 효율이 낮다. 제안한 구조는 하드웨어 비용을 줄이기 위해 X에 대해 순차로 Y에 대해 병렬로 처리하고 효율적인 처리를 위해 다항식의 처리 순서를 적응적으로 바꾼다. 이러한 처리순서는 내부 레지스터의 사용을 최소화하고, 함께 사용되는 데이터를 묶어서 저장함으로써 메모리 구조를 단순하게 하고, 메모리 참조의 빈도를 줄였다. 또한, 각 모듈간의 레이턴시가 고르고, 데이터의 처리가 중첩되어 하드웨어 효율이 높고, 적절한 처리량이 유지된다.

Ⅱ. RS 코드의 연판정 리스트 디코딩

$RS(n,k)$는 k 개의 데이터 심볼을 n 개의 코드워드 심볼로 부호화한다. 이때 각각의 심볼은 $GF(2^q)$의 원소이며, q 비트로 이루어진다. k 개의 메시지 심볼 $m_0, m_1, m_2, \ldots, m_{k-1}$을 다항식의 계수로 생각하면 다음과 같은 메시지 다항식이 생성된다.

$$f(x) = m_0 + m_1 x + m_2 x + \ldots + m_{k-1} x^{k-1} \qquad (1)$$

이 다항식에 n 개의 서로 다른 $GF(2^q)$의 원소를 대입하여 1 개의 코드워드를 얻는다. 일반적으로 $n = 2^q - 1$이며, 대입하는 원소는 0을 제외한 $\{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$을 사용한다. 여기서 α는 원시

원소이고, 대입을 통해 얻은 코드워드는 다음과 같다.

$$c = \{f(1), f(\alpha), f(\alpha^2), \ldots, f(\alpha^{n-1})\} \quad (2)$$

부호화된 코드워드의 복호는 필드의 2 차원 평면상에서 입력, 출력 값이 하나의 점으로 주어지고, 이 n 개의 점들을 지나는 $k-1$차 다항식을 구하는 Interpolation 문제로 귀결되며, 따라서 GS 알고리즘[3]에 의한 디코딩이 가능하다. 다음은 Interpolation 기반 디코딩 알고리즘에서 사용되는 몇 가지 정의이다.

Definition 1 : $Q(X, Y) = \sum_{i}^{\infty}\sum_{j}^{\infty} q_{i,j} X^i Y^j$을 $GF(2^q)$ 위에서 정의된 이변수 다항식이라 하고, w_x, w_y를 음이 아닌 정수라 하자. 그렇다면, $X^i Y^j$ 항의 (w_x, w_y)-weighted degree는 $i w_x + j w_y$이고, 다항식 $Q(X, Y) = \sum_{i}^{\infty}\sum_{j}^{\infty} q_{i,j} X^i Y^j$의 (w_x, w_y)-weighted degree는 0이 아닌 계수를 갖는 항중 가장 큰 값을 갖는 항의 (w_x, w_y)-weighted degree를 뜻한다.

Definition 2 : $P(X, Y)$ 가 $Q(X, Y)$를 각각 X, Y축으로 x, y만큼 평행 이동한 것이라 할 때, m보다 작은 차수의 모든 항의 계수가 모두 0이면, $Q(X, Y)$는 (x, y)를 multiplicity, m만큼 지나간다고 말한다. 이를 식으로 나타내면 다음과 같다.

$$\begin{aligned} P(X, Y) &= \sum_{\beta}^{r}\sum_{\alpha}^{w_t} p_{\alpha,\beta} X^\alpha Y^\beta \\ &= Q(X-x, Y-y) \\ &= \sum_{\beta}^{r}\sum_{\alpha}^{w_t} q_{\alpha,\beta} (X-x)^\alpha (Y-y)^\beta \end{aligned}$$

$$coef(Q(X-x, Y-y), X^\alpha Y^\beta) = p_{\alpha,\beta}$$

$$p_{\alpha,\beta} = 0, \ \forall \alpha+\beta < m \quad (3)$$

여기서 r은 Y의 최대차수이고, w_v 는 Y^v를 포함한 항 중에서 X의 최대차수를 나타낸다.

1. 리스트 디코더

그림 1은 RS 코드의 연판정 리스트 디코더의 블록도이다. $GF(2^q)$상에서 정의된 (n, k)코드워드를 채널을 통해 수신하면 $2^q \times n$ 행렬(Π-행렬)로 표현할 수 있는데, Π-행렬의 각 원소 (x_i, y_j), $1 \le i \le n$, $1 \le j \le 2^q$는 y_j를 수신했을 때, x_i를 사용하여 코딩한 심볼을 전송한 확률을 의미한다. 따라서 Π-행렬은 채널정보를 담고 있다고 할 수 있는데, KV 알고리즘에 따라 Π-행렬을 정수 값을 갖는 M-행렬로 변환한다.

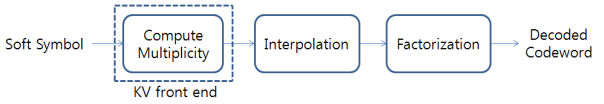

그림 1. RS 연판정 리스트 디코더 블록도
Fig. 1. RS soft decision list decoder block diagram.

이 때 M-행렬의 각 원소는 Π-행렬에 비례하며, 이 중 0이 아닌 값을 해당 포인트의 multiplicity라고 한다.

Interpolation 단계는 주어진 포인트가 해당하는 multiplicity 만큼 지나가는 이변수 다항식을 구하는 단계이며, KV 알고리즘에 의하면 일정 조건에서 이 다항식은 원래의 메시지 다항식을 인수로 포함한다[4]. Factorization은 Interpolation에서 구한 다항식중 하나를 골라 인수 분해하여 코드워드의 리스트를 구하는 과정이며, 마지막으로 이 코드워드를 Π-행렬과 곱하여 가장 확률이 높은 코드워드를 선택하여 디코딩 과정을 마친다.

2. Interpolation

Interpolation 단계는 이변수 다항식이 주어진 포인트를 해당 multiplicity만큼 지나가게 하는 과정으로 Gaussian 소거법과 GS 알고리즘의 방법이 있다. 하지만, Gaussian 소거법은 복잡도가 매우 높으므로 ($O(n^3)$), 다항시간 복잡도를 갖는 GS 알고리즘이 리스트 디코더에서 주로 사용된다. 또한 Nielson의 Interpolation 알고리즘은 포인트마다 다른 multiplicity를 갖는 연판정 디코더에도 적용이 쉽고 하드웨어 구현에 용이하므로 우리는 이를 사용한다. 그림 2는 Fundamental Iterative Algorithm(FIA)[13]에 기반한

- Initialization
  $Q_v(X,Y) = Y^v$, *for* $0 \le v \le r$.
- Iteration for each point set (x_i, y_i, m_{x_i,y_i})
  $O_v = (w_x, w_y)$ − weighted degree of $Q_v(X,Y)$, for $0 \le v \le r$.
  for $\beta = 0$ to $m_{x_i,y_i} - 1$
    for $\alpha = 0$ to $m_{x_i,y_i} - 1 - \beta$
      - DCC(Discrepancy Coefficient Compute)
        $d_v^{(\alpha,\beta)} = coef(Q_v(X+x, Y+y), X^\alpha Y^\beta)$, for $0 \le v \le r$
      - If there exist $\eta = \arg\min_{0 \le v \le r, d_v^{(\alpha,\beta)} \ne 0} \{O_v\}$
      - PU(Polynomial Update)
        $Q_v(X,Y) = d_\eta^{(\alpha,\beta)} Q_v(X,Y) + d_v^{(\alpha,\beta)} Q_\eta(X,Y)$, for $v \ne \eta$
        $Q_v(X,Y) = Q_v(X,Y)(X+x)$, for $v = \eta$
        $O_\eta = O_\eta + 1$
    end for
  end for

그림 2. Interpolation 알고리즘
Fig. 2. Interpolation algorithm.

Interpolation 알고리즘이다.

앞에서 설명했듯이 Interpolation은 이변수 다항식이 주어진 포인트를 해당 multiplicity $m_{i,j}$만큼 지나가도록 하는 과정이다. 포인트 (x, y)가 multiplicity $m_{i,j}$만큼 지나간다는 것은 다항식의 $m_{i,j}$보다 작은 차수 항의 계수가 모두 0이라는 의미이다. 따라서 Interpolation은 $m_{i,j}$보다 작은 차수 항의 계수를 구하는 Discrepancy Coefficient Computation(DCC) 단계와 여기서 구한 계수를 0으로 소거하는 Polynomial Update(PU) 단계로 이루어진다. 그림에서 볼 수 있듯이 알고리즘은 2중 반복문을 갖는데, 반복문은 각각 X, Y의 차수에 해당하며 포인트에 해당하는 m보다 작은 차수에 대해서 Interpolation 과정을 수행한다.

DCC 단계는 각 반복문에서 후보 다항식의 $\alpha+\beta<m$인 $X^{\alpha}Y^{\beta}$항 계수를 구하며, 이를 Berlekamp-Massey 알고리즘에서 사용된 것과 같이 Discrepancy Coefficient(DC)라고 부른다. 후보 다항식이 0이 아닌 DC를 갖는다면 해당 후보 다항식은 포인트를 m만큼 지나가지 않는다는 의미이므로, 이러한 조건을 만족시키기 위해 PU에서 해당 후보 다항식의 계수를 소거한다. 이와 같이 후보 다항식은 Interpolation의 각 단계를 거쳐 각각의 제약조건들을 충족시킨다.

Ⅲ. 제안한 구조

1. 전체 구조 및 스케쥴링

그림 3은 제안한 Interpolation의 전체 구조이다. 구조는 크게 DC를 계산하는 Discrepancy Coefficient Computation Unit(DCCU) 와 후보 다항식을 갱신하는 Polynomial Update Unit(PUU)로 이루어져 있으며, 후보 다항식의 weighted degree를 저장, 유지하고 정렬함으로써 데이터의 처리 순서를 결정하는 Polynomial Order Sorting Unit(POSU)과 다항식의 계수을 저장하는 메모리와 제어부로 구성된다. DCCU는 Hasse Derivative(HD)를 사용하여 DC를 구하며, HD 연산과 HD계산에 필요한 Y의 거듭제곱을 구하는 Y generator로 구성된다. 외부 입력은 Interpolation 포인트와 multiplicity인 (x,y,m)의 스트림이고, 제어부가 입력에 따라 알고리즘의 반복문을 제어하여 필요한 제어신호를 발생한다.

제안한 구조는 Y에 대해서만 병렬처리를 적용하고 X와 다항식에 대해서 순차로 처리함으로써 하드웨어비

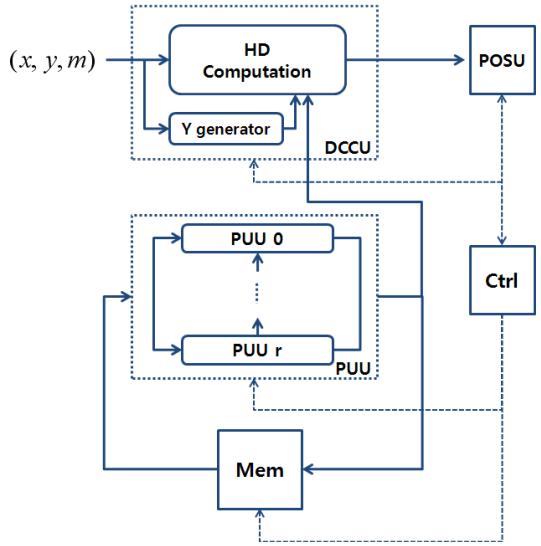

그림 3. 제안된 Interpolation 구조
Fig. 3. Proposed architecture for interpolation.

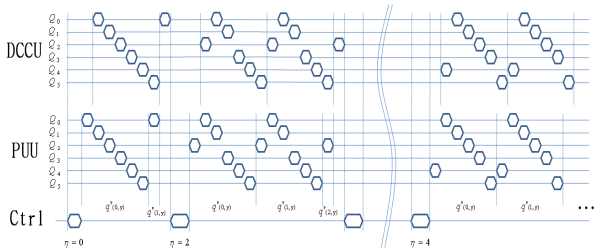

그림 4. Interpolation 스케쥴링
Fig. 4. Interpolation scheduling.

용을 줄였다. 하지만 PU단계에서 최소 다항식의 계수는 다른 후보 다항식의 갱신에 사용되고, 가장 마지막에 갱신이 이루어지기 때문에, 다항식에 대해서 순차로 처리하게 될 경우 메모리의 참조가 불규칙적이고 빈번하게 일어나게 된다. 제안한 구조는 다항식 단위의 순차처리가 아닌, 단항식 단위의 순차처리를 적용하여 이러한 문제를 해결하였다. 후보 다항식의 처리순서는 반복문에서 선택된 최소 다항식에 따라 달라지는데, 최소 다항식이란 0이 아닌 DC를 가지며 weighted degree가 가장 작은 다항식을 말하며 $Q_{\eta}(X,Y)$라고 표기한다.

그림 4는 $r=5$인 경우에 대해 DCCU, PUU의 데이터 처리 순서를 보여주는 타이밍 다이어그램이다. 그림에서 Q_v는 하나의 후보 다항식을 나타내며, $q^{v}_{(i,j)}$는 Q_v의 X^iY^j항 계수를 뜻한다. 제안된 구조는 Y에 대해 병렬로 처리되기 때문에 하나의 Q_v는 동일한 X차수를 갖는 모든 항의 계수$(q^{v}_{x,0}, q^{v}_{x,1}, \cdots, q^{v}_{x,r})$가 처리되는 것을 의미한다. 또한, DCCU의 입력은 이전 반복문에서 PUU의 출력이므로 PUU에서 계산된 계수를 바

로 DCCU에 입력하면, 두 모듈이 동시에 동작할 수 있기 때문에 높은 하드웨어 효율을 얻을 수 있고 레이턴시도 효과적으로 줄일 수 있다. 따라서, 알고리즘의 첫 DC를 계산하는 과정 이후 DCCU와 PUU는 중첩되어 동시에 동작하며, 갱신된 다항식의 계수들은 메모리에 저장됨과 동시에 바로 다음 DC를 계산하기 위해 DCCU로 보내진다. DCCU단계를 마치면 계산된 DC를 바탕으로 $Q_\eta(X,Y)$을 선택하고 각 모듈의 적절한 스케쥴과 제어신호의 발생을 위한 소수 클록을 거쳐 다음 반복문을 처리하게 된다. 그림 4는 $\eta=0,2,4$인 경우의 예를 보여주고 있다.

2. DCCU 구조

앞서 설명했듯이 Interpolation는 주어진 포인트를 m만큼 지나는 다항식을 구하는 과정이며, DCC단계는 각 반복문에 해당하는 DC를 구하는 과정이다. DC를 계산하는 방법은 식 (3)과 같이 다항식 $Q(X,\ Y)$을 X축으로 x만큼, Y축으로 y만큼 평행이동한 다항식 $Q(X+x,\ Y+y)$을 구하여 $X^\alpha Y^\beta$항의 계수를 얻는 것이다. 여기서 α, β는 $\alpha+\beta<m$을 만족하는 음수가 아닌 정수이다. 논문 [10]은 이러한 방법을 사용한 구조를 제안하였는데 먼저 다항식을 적절히 변환한 후, X에 대해 평행이동하고 다시 Y에 대해 평행이동하여 DC를 구하였다. 하지만, 이 구조는 변환을 위해 매우 많은 유한체 곱셈기와 레지스터가 사용되어 하드웨어 비용이 크다. DC를 계산하는 또 다른 방법은 HD를 사용하는 것이다. 식 (4)는 HD를 이용하여 DC를 구하는 식이다.

$$d_v^{(\alpha,\beta)} = coef(Q_v(X+x,Y+y),X^\alpha Y^\beta)$$
$$= \sum_{t=\beta}^{r}\sum_{s=\alpha}^{w_{v,t}}\binom{s}{\alpha}\binom{t}{\beta}q_{s,t}^v x^{s-\alpha}y^{t-\beta},\ \text{for } v=0,1,\ldots,r \qquad (4)$$

하지만 위 식은 이중 누적덧셈의 형태를 가지고 있으므로 그대로 하드웨어에 매핑할 경우 레이턴시가 매우 크다. 제안한 구조는 식 (4)를 사용하여 DC를 구하지만 하드웨어 비용을 줄이기 위해 유한체 곱셈의 사용을 최소화하고, Y에 따라 병렬로 X와 다항식에 대해서 순차로 DC를 구한다. Y에 따라 병렬처리를 하기위해 식 (4)를 풀어쓰면 다음과 같다.

$$d_v^{(\alpha,\beta)} = \sum_{s=\alpha}^{w_{v,t}}\binom{s}{\alpha}x^{s-\alpha}\left\{\binom{0}{\beta}q_{s,0}^v y^{0-\beta}+\binom{1}{\beta}q_{s,1}^v y^{1-\beta}+,\ldots,+\binom{r}{\beta}q_{s,r}^v y^{r-\beta}\right\} \qquad (5)$$

여기서 $s-\alpha$, $t-\beta$ $(0\le t\le r)$은 각각 $s>\alpha$, $t>\beta$일 때 유효한 값을 가진다. 그림 5는 식(5)를 하드웨어 맵핑한 DCU 구조이다. 입력부분의 곱셈은 X의 거듭제곱($x^{s-\alpha}$)을 계산하며, 제안한 데이터 처리순서는 X에 따라 오름차순으로 진행되고 후보 다항식이 단항식 단위로 번갈아가며 처리되므로 한번 계산된 X의 거듭제곱 값은 다른 후보 다항식의 DC를 계산할 때마다 다시 계산할 필요 없이 모두 사용된다. X의 거듭제곱은 식 (5)와 같이 모든 Y차수에 대해 공통이므로 그림과 같이 각 행에 분배된다. $q_{s,t}^v$는 v번째 후보 다항식 ($Q_v(X,Y)$)의 X^sY^t항의 계수를 의미하며, PUU에서 갱신된 값이 바로 입력된다. $\binom{s}{\alpha}$, $\binom{t}{\beta}$는 Lucas의 정리에 따라 간단히 구현되며, X의 거듭제곱과 마찬가지로 $\binom{s}{\alpha}$는 모든 Y차수에 대해 공통으로 입력되고, $\binom{t}{\beta}$는 t값에 따라 최적 값을 구할 수 있다. 뒷부분의 레지스터는 각 후보 다항식의 DC의 중간 계산 값을 저장하며, 저장되는 순서는 다항식의 처리순서와 같다.

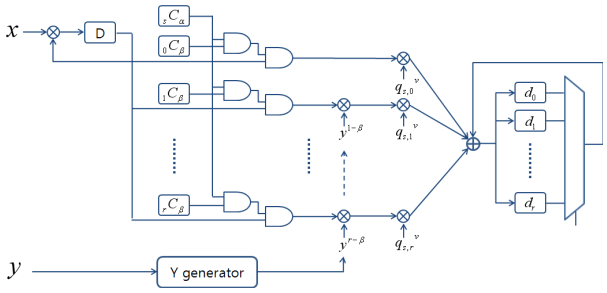

그림 5. DCCU 구조
Fig. 5. The structure of DCCU.

Y generator는 Y의 거듭제곱을 구하는 부분인데 외부에서 새로운 포인트가 입력되면 y의 거듭제곱을 계산하여 레지스터에 차례로 저장한다. 새로운 m에 따른 첫 반복문은 $\alpha=0,\beta=0$이므로 $y^{0-0}=y^0=1$, $y^{1-0}=y^1,\ldots,y^{r-0}=r^r$이다. 따라서 초기 $\beta=0$의 경우 계산된 값이 바로 사용되며, α의 반복문을 마치고 β가 증가하면 레지스터에 저장된 값을 아래로 이동하고 가장 위의 레지스터에 0을 입력하면, 부가적인 mux나 DCCU구조의 변경 없이 새로운 $y^{t-\beta}$값을 DCCU에서 바로 사용할 수 있다. 이러한 y의 거듭제곱을 계산하기 위한 유한체 곱셈기는 $r-1$개가 필요하며, 새로운 m이 입력될 때마다 초기 1번만 $\lfloor \log_2 r \rfloor$의 레이턴시를 가진다. 일반적으로 r은 그리 크지 않으므로 Y generator의 하드웨어 비용은 작고, 레이턴시는 r에 따라 선형적으로 증가하지 않기 때문에 제안한 DCCU의 구조에 적합하다.

3. PUU(Polynomial Update Unit) 구조

PU단계는 선택된 최소 다항식을 가지고 각각의 후보 다항식을 갱신하는 과정이다. 최소 다항식이 아닌 후보 다항식을 먼저 갱신하고 최소 다항식을 마지막으로 갱신하는데 PU 계산식을 다시 쓰면 다음과 같다.

$$Q_v(X,Y)=\begin{cases} d_\eta^{(\alpha,\beta)}Q_v(X,Y)+d_v^{(\alpha,\beta)}Q_\eta(X,Y), & \text{for } v\neq\eta \\ Q_v(X,Y)(X+x), & \text{for } v=\eta \end{cases} \tag{6}$$

여기서 $d_\eta^{(\alpha,\beta)}$, $d_v^{(\alpha,\beta)}$는 각각 최소 다항식과 최소 다항식이 아닌 다항식의 DC를 의미한다. 식 (6)과 같이 최소 다항식이 아닌 다항식은 최소 다항식의 DC를 곱하고 최소 다항식에는 자신의 DC를 곱하여 더함으로써 식 (3)의 평행 이동한 다항식($P(X,Y)$)의 $X^\alpha Y^\beta$항 계수(DC)를 소거한다. 모든 후보 다항식을 갱신한 후에 마지막으로 최소 다항식에 $(X+x)$를 곱하여 m보다 작은 차수항의 계수를 0으로 만드는 제약조건을 만족시켜준다.

PU단계는 DCCU와 마찬가지로 모든 다항식의 계수에 연산이 이뤄져야 하므로 레이턴시가 크다. 이를 위해 논문 [10, 11]은 다항식과 Y에 대해 병렬처리를 적용하여 높은 처리량을 얻었지만 단순한 병렬처리는 $(r+1)^2$만큼의 동일한 하드웨어가 필요하므로 비용이 크다. 제안한 구조는 하드웨어 비용을 줄이기 위해 다항식에 대해 순차로 데이터를 처리한다. 하지만 최소 다항식이 아닌 다항식의 갱신은 갱신되기 이전의 최소 다항식을 필요로 하므로 메모리의 참조가 불규칙적이고 빈번해 지거나 최소 다항식을 저장하기 위한 레지스터가 필요하게 된다. 이러한 문제를 해결하기 위해 단항식 단위의 순차처리 스케쥴을 제안한다. 제안한 구조는 Y에 대해 병렬처리를 하므로 하나의 다항식에서 같은 X차수를 갖는 항들의 계수는 모두 동시에 처리되고, X에 대해 오름차순으로 순차처리를 하여 X에 대해 상수항부터 시작하여 최고차항까지 진행한다. 제안한 스케쥴은 먼저 최소 다항식의 상수항을 읽어와 갱신하고 동시에 레지스터에 저장하면 이를 이용하여 다른 다항식의 상수항을 갱신하는 방법을 사용한다.

그림 6은 제안하는 PUU의 구조이다. 먼저, 최소 다항식의 상수항인 $q_{0,y}^\eta$가 입력되면 출력단의 mux를 제외한 모든 mux의 select 신호가 0이 되면서 $q'=x\,q_{0,y}^\eta$가 계산되고, 레지스터에는 갱신되기 전의 $q=q_{0,y}^\eta$가 저장된다. 이 후, 최소 다항식이 아닌 다항식의 계수, $q_{0,y}^v$가 입력되고 mux의 select 신호가 1로 바뀌어 레지스터에 $q_{0,y}^\eta$를 이용하여 $q_{0,y}^v{}'=d_v\,q_{0,y}^\eta+d_\eta\,q_{0,y}^v$를 계산한다. 최소 다항식을 제외한 다항식 계수의 입력 순서는 편의상 오름차순이며, d_v는 해당하는 다항식과 함께 입력된다. 이전 DCCU에서 구한 다항식의 DC가 모두 0이면 최소 다항식이 존재하지 않는데 이를 나타내는 특정값(Ψ)이 출력단의 mux를 통해 PU과정을 생략한다.

그림 7은 Y에 따라 병렬로 구현된 PUU의 전체 모습이다. 각 PUU는 하나의 Y차수에 해당하는 X에 대한 다항식($\theta_v(X)$)을 계산하므로, 총 $(r+1)$개의 PUU로 구성된다. PUU의 앞, 뒤 레지스터는 입, 출력 계수를 저장하며 후보 다항식을 하나씩 담당하여 저장한다. 하나의 레지스터는 하나의 후보 다항식에서 X의 차수가 같은 모든 계수를 저장하고 그 크기는 $(r+1)\cdot p$이다. 여기서 p는 $GF(2^p)$를 이루는 비트 수이다. 출력단의 레지스터는 PU단계를 거친 계수를 하나씩 저장하고 동시에 DCCU에 보내어 두 모듈의 동작이 중첩되도록 한다. 모든 레지스터가 다 저장되면 메모리에 쓰는데, 이러한 레지스터의 사용은 빈번한 메모리 참조를 $(r+1)$clock 동안 한 번 읽고, 쓰는 두 번의 참조로 줄여주며, 메모리의 구조도 단순하게 한다.

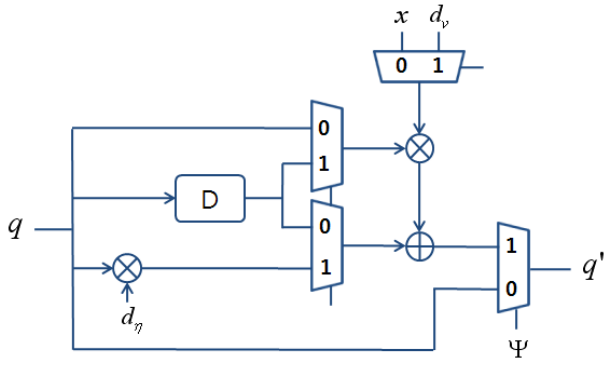

그림 6. 하나의 PUU 구조
Fig. 6. Structure of a PUU element.

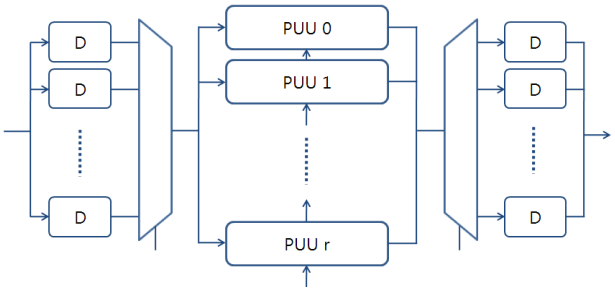

그림 7. 전체 PUU 구조
Fig. 7. Overall structure of PUU.

4. POSU(Polynomial Order Sorting Unit) 구조

Interpolation에서 또 하나의 중요한 부분은 DCCU에서 구한 DC와 weighted degree를 통해 최소 다항식을 구하는 것이다. 매번 최소 다항식을 구하기 위해 다항식의 weighted degree를 계산하고 비교하는 것보다 한 번 계산하여 저장해 두고, 다항식이 갱신될 때마다 차수를 갱신하고 정렬하는 것이 더 빠르고 효율적이므로 이런 기능을 갖는 POSU를 구현하여 사용한다. 식 (6)에서 보듯이 최소 다항식은 $(X+x)$를 곱하여 차수가 증가하지만 그 외의 다항식은 차수가 그대로 유지되므로 최소 다항식의 차수만 1을 더해주면 된다. 따라서 하나의 다항식만 weighted degree에 따라 정렬하면 되므로 한 번의 거품정렬을 통해 쉽게 수행할 수 있다.

그림 8은 weighted degree에 따라 다항식을 재 정렬하는 모듈이다. 먼저 각 레지스터는 weighted degree가 저장된 부분(O_v)과 다항식의 번호를 저장하는 부분(v)로 구성되며, $(r+1)$개가 모여 쉬프트 레지스터를 이룬다. O_v는 $1,(k-1),2(k-1),...,r(k-1)$로, v는 $0,1,2,...,r$로 초기화되고, Interpolation이 진행되면서 DCCU에서 계산된 DC가 입력되어 η가 선택되면 레지스터는 위로 이동하면서 $v=\eta$의 여부에 따라 weighted degree에 1을 더하고, 갱신된 값을 weighted degree 비교기에 입력하여 더 낮은 값을 갖는 다항식을 레지스터에 입력하여 한 번의 거품정렬을 수행한다.

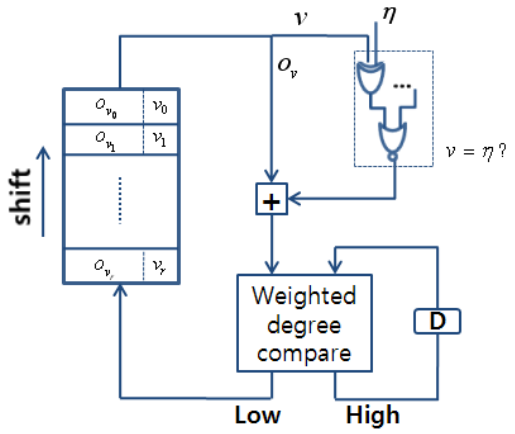

그림 8. POSU 구조
Fig. 8. Structure of POSU.

Ⅳ. 설계 및 성능 분석

이 장에서는 제안한 구조를 $RS\ code\ (n,\ k)=(255,\ 239)$에 적용하여 하드웨어 비용과 레이턴시, 처리량을 분석하고, 기존의 구조와 비교한다. 원시 다항식은 $GF(2^8)$상에서 $p(x)=1+x^2+x^3+x^4+x^8$를 사용하였다. KV 전단부는 $\lambda=5$인 low complexity method를 사용하여 $m_{\max}=5$가 되며, C = 3,800일 때, 10^{-5}에서 연판정 디코더가 경판정 디코더에 비해 0.5dB의 코딩이득을 얻을 수 있으므로, $r=5$을 얻는다. 높은 처리량을 얻기 위해선 알고리즘 수준에서 반복문의 수를 줄이는 것이 가장 효과적이므로 리인코딩 기법을 사용했다. 리인코딩을 사용하여 반복문의 수를 계산하면 $C'=C\times(n-k)/n=3,800\times16/255\cong239$이고, C'을 Interpolation에 적용했을 때 다항식의 X 차수를 계산하면 $C'/(r+1)=239/6\approx40$을 얻고, 이로부터 최대 X차수로 50을 사용하면, 총 레이턴시는 $(\lceil \frac{C'}{2} \rceil+\epsilon)C'$이고, 여기서 ϵ는 그림 4에서 나타난 최소 다항식을 선택하고 제어신호를 발생하는데 드는 클록 수이다. 위 식에 $C'=239$, $\epsilon=4$를 대입하면, $(\lceil \frac{239}{2} \rceil+4)\times239=29636$clock이다.

표 1은 알고리즘 레벨에서 DCCU, PUU의 게이트를 분석한 것이다. 논문 [10]은 point serial 알고리즘을 사용하고 다항식과 Y에 따라 병렬처리를 적용하여 높은 처리량을 얻었다. 하지만 point serial 알고리즘은 모든 포인트가 높은 multiplicity를 가질 때 높은 성능을 얻을 수 있으며, 이를 위해 사용되는 Discrepancy Polynomial(DP)는 추가 하드웨어를 필요로 한다. 논문 [10]의 구조는 DPC모듈을 통해 DC를 효율적으로 구할 수 있지만 PU모듈은 여전히 레이턴시가 크다. 따라서 DPC모듈이 PU모듈을 기다려야 하므로 하드웨어 효율이 떨어진다. 파라미터 τ은 $r\approx m$이므로 r보다 크거나 같기 때문에 하드웨어 비용이 큰 것을 알 수 있다. 논문 [11]은 다항식과 Y에 따라 병렬처리를 적용하고, 심볼

표 1. 하드웨어 복잡도 및 성능분석
Table 1. Hardware complexity and performance.

		HW 복잡도	Latency
DCCU	$GF(2^q)$ multiplier	$3r+1$	$\lceil \frac{C}{2} \rceil$
	$GF(2^q)$ adder	$r+1$	
	registers	$(2r+3)+\lceil \log_2 r \rceil$	
PUU	$GF(2^q)$ multiplier	$2(r+1)$	$\lceil \frac{C}{2} \rceil$
	$GF(2^q)$ adder	$r+1$	
	registers	$(r+1)+2(r+2)^2$	
total			$(\lceil \frac{C}{2} \rceil+\epsilon)C$

의 표기를 지수 표기와 병행하여 사용함으로써 심볼의 곱셈을 덧셈으로 변환하여 파이프라인을 적용하여 높은 처리량을 얻을 수 있었다. 하지만 병렬 처리한 만큼 하드웨어가 필요하며 입력이 일반적인 포맷이 아니므로 전처리 과정이 필요하며, 계산 후 다시 일반적인 포맷으로 변환해 주어야 하므로 LUT가 추가로 필요하다.

표2는 제안한 구조와 기존 구조를 하드웨어와 성능에 대하여 비교한 표이다. 유한체 곱셈기는 64 XOR 게이트와 48 AND 게이트로 구현할 수 있다. Interpolation 구조의 하드웨어 복잡도는 $(m_{\max}+1)^2$에 비례한다[11]. 논문 [14],[15]의 구조가 $m_{\max}=2$를 적용하였으므로 공정한 분석을 위하여 논문 [10~11]도 동일한 파라메터를 적용하여 분석하였다. 각 논문에서 제안한 최적화 방법을 적용하였으며, AND/OR는 XOR의 3/4, MUX와 메모리 셀은 XOR와 동일하게, 레지스터는 XOR의 3배 면적으로 평가하였다[14]. 표 2에서 보는 바와 같이 제안한 구조의 효율이 가장 높게 나타났다. 표에서 제안한 구조의 레이턴시는 증가하였으나 하드웨어 감소에 의한 최장경로지연시간(critical path delay)이 감소되어 실제 전체 성능에 영향은 줄어들었다.

제안한 구조는 제어부와 함께 Verilog HDL로 모델링 되었으며, Synopsys Design Compiler를 이용하여 게이트 수준의 합성 결과를 얻었다. 동부하이텍 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과, 최대 동작 주파수는 200㎒이며 약 25.1K 게이트가 사용되었다.

연판정 Reed-Solomon 디코더는 그림1에서 보는 바와 같이 Interpolation 구조 다음에 Factorization 구조가 더 필요하며 하드웨어 복잡도가 매우 커서 실제 구현 단계에서는 디코딩 성능을 만족하는 범위 내에서 하드웨어의 구조를 단순화 하는 것은 매우 중요하다. 제안한 구조의 성능은 중요한 어플리케이션에 적용이 가능하다. 또한 최근의 60-70nm급 DSM(Deep Sub-Micron) 반도체공정 기술을 적용할 경우 동작주파수는 쉽게 500㎒ 이상이 될 것으로 생각되며, 본 구조에 파이프라인 등과 같은 고성능 구조를 적용한다면 다항식 처리의 직렬화에 의한 디코딩 성능저하를 많이 개선할 것이라 생각된다.

표 2. 하드웨어 복잡도 및 성능비교
Table 2. Hardware complexity and performance comparison.

Design	Area (# of XOR gates)	Critical path (# of XOR gates)	Latency	Throughput (normalized)	Efficiency (normalized)
[10]	8535	12	1437	1	1
[11]	11726	4	1775	2.43	1.77
[14]	7872	10	916	1.88	2.04
[15]	10718	12	454	3.17	2.52
proposed	1321	4	10650	0.4	2.62

표 3. 합성결과 분석
Table 3. Synthesis results and analysis.

Parameter				Performance	
C	$m_{\max}$	r	ϵ	total latency	max clock freq.
239	5	5	4	29636	200MHz

	DCCU	PUU	control	total
Gate count	7K	11K	7.1K	25.1K

V. 결 론

본 논문은 연판정 RS 리스트 디코더에서 계산량이 매우 많은 Interpolation모듈에 새로운 스케쥴을 적용하여 하드웨어 비용을 줄인 구조를 제안하였다. 제안한 스케쥴은 Y에 대해 병렬처리를 적용하였으며 X와 다항식에 따라 순차로 처리하되 단항식 단위로 다항식의 계수를 번갈아 처리한다. 이러한 스케쥴은 내부 레지스터의 수를 최소화하고 메모리의 참조를 규칙적이고 그 횟수를 줄였으며, 메모리의 구조를 단순하게 하여 면적면에서 매우 효율적이다. 제안한 구조는 병렬처리의 적용을 Y로 제한하여 하드웨어 비용을 낮추고, DCCU와 PUU의 처리를 서로 중첩하여 레이턴시를 낮추고 하드웨어 효율을 높였다. 제안한 구조는 동부하이텍 0.18㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 약 25.1K의 면적으로 최대 200㎒로 동작이 가능하다.

감사의 글

저자들은 본 연구를 위하여 설계 환경을 제공하여준 IDEC에 감사드린다.

참 고 문 헌

[1] R. E. Blahut, Theory and practice of Error Control Codes, Addison-Wesley, Reading MA, 1983.

[2] M. Sudan, “Decoding of Reed-solomon codes beyond the error correction bound”, J. complexity, vol. 12, pp. 180-193, 1997.
[3] V. Guruswami and M. Sudan, “Improved decoding of Reed-Solomon and algebraic-geometric codes”, IEEE Trans. Inf. Theory, vol. 45, no. 6, pp. 1755-1764, Sep. 1999.
[4] R. Koetter and A. Vardy, “Algebraic soft-decision decoding of Reed-Solomon codes”, IEEE Trans. Inf. Theory, vol. 49, no. 11, pp. 2809-2825, Nov. 2003.
[5] W. J. Gross, F. R. Kschischang, R. Koetter, and P. Gulak, “Simulation results for algebraic soft-decision decoding of Reed-Solomon codes”, in Proc. 21st Biennial symp. Commun., pp. 356-360, 2002.
[6] W. J. Gross, F. R. Kschischang, R. Koetter, and P. Gulak, “Applications of algebraic soft-decision decoding of Reed-Solomon codes”, IEEE Trans. Commun., vol. 54, no. 7, pp. 1224-1234, Jul. 2006.
[7] R. R. Nielson, List decoder of Linear Block Codes, Ph.D thesis, Technical University of Denmark, 2001.
[8] K. Lee and M. O'Sullivan, “An interpolation algorithm using Grobner bases for soft-decision decoding of Reed-Solomon codes”, Proc. of ISIT, Seattle, WA, Jul. 2006, pp. 2032-2036
[9] R. Koetter, J. Ma, A. Vardy and A. Ahmed, “Efficient interpolation and factorization in algebraic soft decision decoding of Reed-Solomon codes”, Proc. of IEEE Symp. On Info. Theory, 2003
[10] A. Ahmed, R. Koetter, and N. Shanbhag, “VLSI architectures for soft-decision decoding of Reed-Solomon codes”, in Proc. ICC, 2004, pp. 2584-2590.
[11] Z. Wang, and J. Ma, “High-speed interpolation architecture for soft-decision decoding of Reed-Solomon codes”, IEEE Trans. VLSI systems, vol. 14, no. 9, pp. 937-950, Sep. 2006.
[12] W. J. Gross, F. R. Kschischang, and P. Gulak, “Architecture and implementation of an interpolation processor for soft-decision Reed-solomon decoding”, IEEE Trans. VLSI systems, vol. 15, no. 3, pp. 309-318, Mar. 2007.
[13] G. L. Feng and K. K. Tzeng, “A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes”, IEEE Trans. Inf. Theory, vol. 37, no. 5, pp. 1274-1287, Sep. 1991.
[14] J. Zhu, X. Zhang, and Z. Wang, “Backward interpolation architecture for algebraic soft-decision Reed - Solomon decoding,” IEEE Trans. VLSI systems, vol. 17, no. 11, pp. 1602-1615, Nov. 2009.
[15] X. Zhang and J. Zhu, “High-throughput interpolation architecture for algebraic soft-decision Reed - Solomon decoding,” IEEE Trans. Circuits and systems, vol. 57, no. 3, pp. 581-591, Mar. 2010.

저 자 소 개

이 성 만(정회원)
2008년 가톨릭대학교 정보통신전자공학과 졸업.
2010년 가톨릭대학교 정보통신전자공학과 석사 졸업.
2010년～현재 LG전자 연구원
<주관심분야 : VLSI 설계, 통신신호처리, SOC 설계>

박 태 근(정회원)-교신저자
1985년 연세대학교 전자공학과 졸업.
1988년 Syracuse Univ. Computer 공학석사 졸업.
1993년 Syracuse Univ. Computer 공학박사 졸업.
1991년～1993년 Coherent Research Inc. USA VLSI 엔지니어.
1994년～1998년 현대전자 System IC 연구소 책임연구원
1998년～현재 가톨릭대학교 정보통신전자공학부 교수
<주관심분야 : VLSI 설계, CAD, 컴퓨터 구조>