Journal of The Institute of Electronics and Information Engineers Vol.53, NO.4, April 2016 http://dx.doi.org/10.5573/ieie.2016.53.4.048
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2016-53-4-6

1.4 Gbps 비이진 LDPC 코드 복호기를 위한 Fully-Parallel 아키텍처

(Fully-Parallel Architecture for 1.4 Gbps Non-Binary LDPC Codes Decoder)

최 인 준*, 김 지 훈**

(Injun Choi and Ji-Hoon Kim©)

요 약

본 논문은 GF(64) (160,80) 정규 (2,4) 비이진 LDPC 코드 복호기를 위한 높은 처리량의 병렬 아키텍처를 제안한다. 복호기의 복잡도를 낮추기 위해 체크 노드와 변수 노드의 차수가 작은 코드를 사용하며 뛰어난 에러 정정 성능을 위해 높은 위수의 유한체에서 정의된 코드를 사용한다. 본 논문은 Fully-parallel 아키텍처를 설계하고 체크 노드와 변수 노드를 interleaving하여 복호기의 데이터 처리량을 향상시켰다. 또한 체크 노드의 초기화 지연을 단축시킬 수 있는 조기 분류 기법을 제안하여 데이터 처리량을 추가로 향상시켰다. 제안된 복호기는 1 iteration에 37사이클이 소요되며 625MHz 동작주파수에서 1402Mbps의 데이터 처리량을 갖는다.

Abstract

This paper presents the high-throughput fully-parallel architecture for GF(64) (160,80) regular (2,4) non-binary LDPC (NB-LDPC) codes decoder based on the extended min sum algorithm. We exploit the NB-LDPC code that features a very low check node and variable node degree to reduce the complexity of decoder. This paper designs the fully-parallel architecture and allows the interleaving check node and variable node to increase the throughput of the decoder. We further improve the throughput by the proposed early sorting to reduce the latency of the check node operation. The proposed decoder has the latency of 37 cycles in the one decoding iteration and achieves a high throughput of 1402Mbps at 625MHz.

Keywords : Non-binary LDPC code, Extended min-sum algorithm, Fully-parallel architecture, High-throughput

Ⅰ. 서 론

오늘날의 무선 통신과 스토리지 시스템은 열악한 채널 환경에서도 신뢰성 있는 데이터를 전송하고 저장하기 위해 주로 채널 코드와 같은 에러 정정 기술을 사용한다. 다양한 채널 코드 기술 중 LDPC(low density parity check) 코드는 뛰어난 에러 정정 성능으로 다양한 시스템에서 널리 사용되고 있다[1]. 반복 알고리즘을 통해 복호되는 이진 LDPC 코드는 매우 긴 코드 길이에서 채널 용량에 근접하는 우수한 성능을 발휘하지만 짧거나 중간 길이의 코드에서는 성능이 떨어지며 오류 마루 현상과 같은 문제점들이 나타나기 시작한다. 이러한 문제점들을 해결하기 위해서 위수가 q가 2보다 큰 Galois Field(GF)에서 정의된 비이진 LDPC 코드에 대

* 학생회원, 충남대학교 전자전파정보통신공학과
(Dept. of Electronics Engineering, Chungnam National University)
** 정회원, 서울과학기술대학교 전기정보공학과
(Dept. of Electrical and Information Engineering, Seoul National University of Science and Technology)
© Corresponding Author(E-mail: jihoonkim@seoultech.ac.kr)
※ 이 연구는 충남대학교 학술연구비에 의해 지원되었음
Received ; January 19, 2016 Revised ; April 1, 2016 Accepted ; April 6, 2016

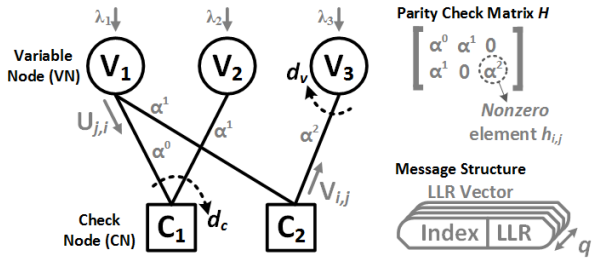


그림 1. 비이진 LDPC 코드의 Tanner 그래프와 메시지 구조
Fig. 1. Tanner graph and message structure of NB-LDPC code.

한 연구가 활발히 진행되고 있다. 비이진(non-binary) LDPC 코드는 이진 LDPC 코드에 비해 우수한 성능을 발휘하고 오류 마루 현상도 완화시키기 때문에 강력한 채널 코딩 기술을 요구하는 분야를 위한 차세대 코딩기술로 주목 받고 있다[2].

비이진 LDPC 코드는 이진 LDPC 코드와 같이 신뢰도 전파에 기반한 반복 알고리즘을 통해 복호되며 이진 LDPC에서 사용되는 Sum-Product Algorithm(SPA)에서 비이진 형태로 확장된 Qary SPA(QSPA)을 사용한다[2]. QSPA를 사용한 비이진 LDPC 코드 복호는 성능이 매우 뛰어나지만 비이진 메시지에 기반 한 연산으로 인해 복호 복잡도가 매우 복잡하다. 비이진 LDPC 코드의 높은 복잡도는 하드웨어 면적을 증가시키고 높은 처리량의 복호기 구현을 어렵게 한다. QSPA의 체크 노드 연산을 간소화하기 위해 FFT-domain[3]과 Log-domain[4]에서 정의된 QSPA가 제안되었지만 여전히 복잡도가 높아 하드웨어 구현이 어렵다. [5]에서 제안된 extended min-sum algorithm(EMSA)은 체크 노드 연산을 더욱 간소화했을 뿐만 아니라 q보다 작은 n_m으로 메시지의 길이를 줄여 복잡도를 낮추었다. EMSA를 더욱 간소화한 min-max algorithm[6]와 simplified-EMSA[7]는 복잡도를 더욱 줄였지만 에러 정정 성능저하가 비교적 크다. 본 논문은 복호 복잡도가 비교적 작고 QSPA 근접한 에러 정정 성능을 제공하는 EMSA를 기반으로 한 비이진 LDPC 코드 복호기를 설계한다.

비이진 LDPC 코드의 체크 노드 연산은 GF의 위수 q와 체크 노드의 차수 d_c가 커질수록 매우 복잡하다. 이러한 체크 노드의 복잡도를 줄이기 위해 전/후방 알고리즘이나 trellis 알고리즘이 사용된다. 전/후방 알고리즘은 위수 q가 큰 GF에서 정의된 비이진 LDPC 코드의 복호 복잡도를 낮출 수 있지만 d_c가 커질수록 전/후방 연산에 의한 연산 지연이 커져 복호기의 데이터 처리량을 떨어뜨린다. [8]에서는 전/후방 알고리즘에 의해 발생하는 연산 지연을 피하기 위해 체크 노드를 구성하는 연산 블록을 병렬로 처리할 수 있는 trellis-EMSA (T-EMSA)을 제안하였다. T-EMSA를 사용하는 비이진 LDPC 코드 복호기는 병렬연산을 통해 체크 노드의 연산 지연을 줄일 수 있지만 병렬 구조로 인한 하드웨어 면적이 크며, GF의 위수 q가 커질수록 하드웨어 면적이 급격히 커지는 단점이 있다.

본 논문은 GF(64) (160,80) 정규 (2,4) 비이진 LDPC 코드를 사용하여 높은 데이터 처리량을 갖는 EMSA기반의 복호기를 제안한다. 복호기의 복잡도를 낮추기 위해 체크 노드와 변수 노드의 차수가 작은 코드를 사용하고 뛰어난 에러 정정 성능을 위해 높은 유한체에서 정의된 코드를 사용한다. 본 논문은 복호기의 데이터 처리량을 높이고 복호 스케줄을 간소화하기 위해 병렬 아키텍처를 사용하고 체크 노드 연산의 초기화 지연을 줄일 수 있는 조기 분류기법을 제안하여 1 iteration의 복호시간을 37사이클로 단축하였다.

Ⅱ. 비이진 LDPC 코드 및 복호 알고리즘

유한체 GF(q), $q>2$, 에서 정의된 비이진 LDPC 코드는 이진 LDPC 코드 보다 뛰어난 에러 정정 성능을 제공할 뿐만 아니라 낮은 오류 마루를 가지고 있다. 이 장에서는 비이진 LDPC 코드의 배경지식과 복호 알고리즘인 Extended Min-Sum Algorithm (EMSA)에 대해 설명한다.

1. 비이진 LDPC 코드

LDPC 코드는 대부분 0으로 구성되어 1의 개수가 매우 적은 $M\times N$ 패리티 검사 행렬 H에 의해 정의되며, 이때 메시지 길이는 $N-M$이 되고 패리티 비트는 M이 되어 전체 코드 워드 길이는 N이 된다. 행렬 H가 GF(2)에서 정의된 경우, 이진 LDPC 코드라 불리며 위수 q가 2보다 큰 GF(q)에서 정의된 경우 비이진 LDPC 코드라 불린다. 비이진 LDPC 코드는 그림 1과 같이 H를 시각적으로 나타낸 Tanner 그래프에 의해 표현될 수 있다. Tanner 그래프에서 H의 각 행은 M개의 체크 노드(C_i), 각 열은 N개의 변수 노드(V_j)로 표현된다. H의 각 열에 포함된 1의 수를 체크 노드 차수 d_c, 각 행에 포함된 1의 수를 변수 노드 차수 d_v라고 정의한다. Edge는 각각의 노드들을 연결하며 0과 1이 아닌 GF(q)의 원소 $h(i,j)$을 곱하는 교환(permutation)과 역교환을 수행한다. Tanner 그래프는 H를 표현하는 대표

적인 방식으로 LDPC 코드의 복호 과정을 이해하는데 도움을 준다.

비이진 LDPC 코드의 메시지는 $\log_2 q$ 비트의 심볼로 구성되며 q개의 심볼 각각에 대한 사전확률을 이용하여 복호된다. 그림 1의 message structure는 각 노드들 사이에서 교환되는 메시지의 구조를 보여준다. 전달되는 메시지는 q개의 원소 각각에 대한 공산(likelihood)으로 q개의 log likelihood ratio(LLR)을 포함하는 LLR Vector(LLRV)와 각각의 LLR과 매칭되는 원소를 나타내는 GF 인덱스의 벡터로 구성된다. 본 논문은 채널 메시지, 변수 노드에서 체크 노드로 전달하는 메시지(V2C) 그리고 체크 노드에서 변수 노드로 전달하는 메시지(C2V)를 각각 $\lambda_j, U_{j,i}$ 그리고 $V_{i,j}$ 로 정의한다.

비이진 LDPC 코드를 정의하는 H는 에러 정정 성능을 결정할 뿐만 아니라 복호기의 복잡도를 결정짓는 중요한 요소이다. 본 논문은 뛰어난 에러 정정 성능을 제공하고 복호 복잡도와 배선 복잡도가 비교적 작은 GF(64) (160,80) 정규 (2,4) 비이진 LDPC 코드를 사용한다. d_c가 작은 경우에도 비교적 뛰어난 에러정정 성능을 보장하기 위해 binary images[9]를 기반으로 H의 원소를 선택하고 큰 girth를 얻기 위해서 progressive edge growth(PEG) 알고리즘[10]을 통해 매우 sparse한 H을 구성하였다.

2. Extended Min-Sum 알고리즘

비이진 LDPC 코드는 체크 노드 연산과 변수 노드 연산을 통해 계산된 메시지를 Tanner 그래프 상에서 연결된 노드들 사이에서 메시지 전달 알고리즘에 따라 반복적으로 전달함으로써 복호된다. 본 논문은 EMS 복호 알고리즘을 통해 비이진 LDPC 코드를 복호 한다. EMSA는 크게 초기화 단계, 체크 노드 과정 그리고 변수 노드 과정으로 구성된다.

초기화 단계에서 각 변수 노드는 채널 정보로부터 계산된 메시지 λ_j에 의해 초기화된다. GF(q)의 원소 각각에 대한 공산을 나타내기 위해, λ_j는 GF(q)의 모든 원소 α_k, $k \in \{1,\cdots,q\}$, 대한 LLRV를 포함한다. 채널로부터 y_j을 받았을 때 변수 노드의 심볼 v_j가 α_k일 확률을 $P(v_j=\alpha_k|y_j)$ 라고하면, j번째 변수 노드의 k번째 심볼의 LLR $\lambda_j[k]$은 수식 (1)과 같이 정의된다.

$$\lambda_j[k]=\log\frac{P(v_j=\widehat{\alpha_k}|y_j)}{P(v_j=\alpha_k|y_j)} \quad (1)$$

표 1. 낮은 복잡도의 NB-LDPC 복호 알고리즘
Table 1. Low complexity NB-LDPC decoding algorithm.

알고리즘 1. 확장 최소합 알고리즘(EMSA)

Step 1. 초기화

반복횟수 k, 변수 노드 $U_{j,i}$ 및 코드워드 Z 초기화;

$k=0$, $U_{(j,i)}=\lambda_j$,

$z_j=\arg\{\min_{d\in GF(q)}\lambda_j(d)\}$일 때, $Z=[z_1,\cdots,z_N]$

Step 2. 신드롬(Syndrome) 체크

만약, 신드롬 $S=Z\otimes H^T$가 0이라면 복호를 종료하고 코드워드 Z를 출력, 그렇지 않으면 Step 3으로 이동

Step 3. 반복횟수 확인

만약, $k=k_{\max}$라면 복호를 종료하고 복호 실패 선언, 그렇지 않으면 Step 4로 이동

Step 4. 체크 노드 과정

$V_{i,j}(d)=\min_{Conf(\beta_{i,j}=d)}\sum_{j'=1,j'\neq j}^{d_c}U_{j',i}(h_{j',i}\otimes\beta_{j',i})$

Step 5. 변수 노드 과정

$U_{j,i}(h_{j,i}\otimes d)=\lambda_j(d)+\sum_{i'=1,i'\neq i}^{d_v}V_{i',j}(h_{i',j}\otimes d)$

Step 6. 사후확률 계산 및 잠정적 판정

사후확률 연산; $\lambda_j^{Pbst}=\lambda_j(d)+\sum_{i'=1}^{d_v}V_{i',j}(h_{i',j}\otimes d)$

잠정적 판정; $z_j=\arg\{\min_{d\in GF(q)}\lambda_j^{Pbst}(d)\}$

반복횟수 증가; $k=k+1$, Step 2로 이동

이때 $\widehat{\alpha_k}$은 확률 $P(v_j=\alpha_k|y_j)$가 가장 큰 심볼을 의미한다. 수식 (1)에 따르면 LLR값이 작은 심볼이 높은 공산을 갖는다. 각각의 LLR은 그에 상응하는 GF 원소 α_k의 공산을 나타내기 때문에 메시지는 이에 대한 GF 인덱스를 포함한다. EMSA의 가장 큰 특징은 복호 복잡도를 낮추고 메모리 요구량을 줄이기 위해 공산이 높은 $n_m(n_m \ll q)$개의 LLR과 GF 인덱스만으로 메시지를 구성하는 것이다. 본 논문은 [11]의 실험 결과에 따라 GF(64)의 메시지 절단 길이 n_m을 16으로 선택하였다.

체크 노드 과정에서는 변수 노드로부터 메시지를 전달 받아 C2V를 계산한다. 교환은 H의 원소와 메시지의 GF 인덱스를 곱하는 과정으로 인덱스를 $\beta(j,i)$라고 정의할 때 $h(j,i)\otimes\beta(j,i)$로 표현된다. 연산자 $\otimes$는 GF 곱셈을 나타내며 look up table을 통해 구현된다. 체크 노드는 i번째 패리티 체크 식 $\sum_{j=1}^{d_c}h_{(i,j)}\otimes\beta_j = d$을 만족하는 β_j의 세트 $Conf(\beta_j=d)$ 중에서 각각의 β_j와 상응하는 LLR들의 합 중 최솟값들을 구하는 연산을 수행하며 알고리즘 1의 Step 4을 따라 수행된다.

변수 노드 과정에서는 체크 노드에서 메시지를 전달 받아 V2C 메시지와 사후확률을 계산하고 잠정적 판정을

수행한다. 변수 노드 과정에서는 역 교환이 수행되며 교환과 마찬가지로 LUT로 구현된다. V2C 연산과 사후확률 연산은 q개의 GF 인덱스 중에서 각각의 원소 d에 상응하는 LLR들의 합을 구하는 연산으로 알고리즘 1의 Step 5와 Step 6을 따라 수행된다. 변수 노드 과정에서는 잠정적 판정을 통해 코드워드의 경판정 값을 구한 다음 신드롬 체크를 수행한다. EMSA의 전체 복호 과정을 요약하면 알고리즘 1과 같다.

Ⅲ. 체크 노드 및 변수 노드 설계

비이진 LDPC 코드 복호는 체크 노드 연산과 변수 노드 연산을 반복적으로 수행하기 때문에 각각의 연산 블록을 최적화 하는 것이 매우 중요하다. 본 논문은 체크 노드의 복잡도를 낮추기 위해 전/후방 알고리즘 사용하였으며 bubble check 알고리즘과 two-pass scan 기법을 사용하여 각각의 노드들을 구현했다.

1. 낮은 복잡도의 체크노드 설계

EMSA 기반의 비이진 LDPC 복호기에서 체크 노드 연산은 전/후방 알고리즘을 통해 단한 elementary check node(ECN) 블록으로 나누어 처리되며 각각의 ECN은 bubble check 알고리즘을 통해 구현된다. 본 절은 전/후방 알고리즘과 bubble check 알고리즘에 대해 설명하고 이를 기반으로 한 낮은 복잡도의 체크노드 아키텍처를 소개한다.

가. 전/후방 알고리즘

전/후방 알고리즘은 특정 연산을 전방 연산과 후방 연산으로 분해하여 이를 반복적으로 수행하고 연산의 결과를 다시 조합(merge)하는 방식이다. 높은 차수를 갖는 체크 노드의 연산을 직접 수행하면 연산의 복잡도가 매우 커질 뿐만 아니라 같은 연산을 여러 번 반복하기 때문에 연산 지연도 매우 크다. 따라서 d_c가 3보다 큰 체크 노드의 연산의 경우 전/후방 알고리즘을 통해 ECN 단위로 분해하여 전/후방 연산을 수행하고 그 결과를 다시 조합하여 체크 노드 연산을 효율적으로 수행할 수 있다. ECN은 체크 노드의 기본 연산 모듈로 수식 (2)를 수행한다.

$$V(d) = \min_{conf(\beta = d)} \sum_{j=d}^{2} U_j(\beta_j) \quad (2)$$

두 개의 메시지 U_1과 U_2을 입력 받아 하나의 메시지

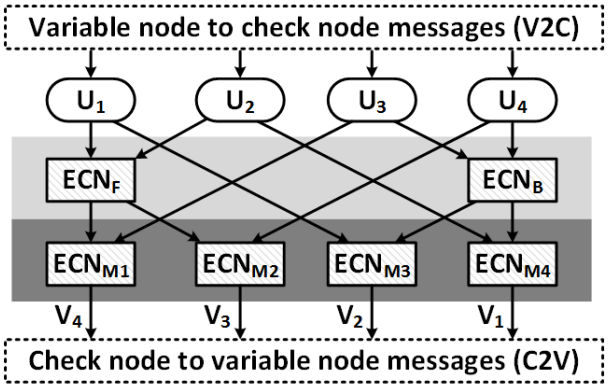


그림 2. 전/후방 알고리즘을 사용한 체크 노드 연산
Fig. 2. Check node operation with forward-backward algorithm.

V을 출력하며 전/후방 연산의 결과 V는 다시 전/ 후방 연산 또는 조합 연산을 위한 ECN으로 입력된다. 전/후방 알고리즘의 마지막 단계에서 수행되는 조합 연산은 V_j을 계산하며 이는 Tanner 그래프에 따라 연결된 변수 노드로 전달된다.

그림 2는 $d_c = 4$일 때의 전/후방 알고리즘을 통한 체크 노드 연산 과정을 나타낸다. 체크 노드는 ECN_F와 ECN_B으로 전/후방 연산을 수행하고 $ECN_{M1 \sim M4}$을 통해 조합 연산을 수행하여 총 2 단계의 연산을 수행한다. 체크 노드 연산은 전/후방 연산에서 $d_c - 3$ 단계 그리고 조합 연산에서 1 단계가 걸려 총 $d_c - 2$ 단계에 걸쳐 수행된다. 정규 (2,4) 코드의 경우 체크 노드 연산을 수행하는데 총 2 단계가 필요하다.

나. Bubble check 알고리즘

EMSA에서의 체크 노드 연산은 메시지들의 합과 그 합들 중 최솟값을 찾는 연산을 수행한다. 메시지가 n_m개의 LLR로 구성된 EMSA의 경우 $n_m{}^2$개의 교차 합이 존재하며 교차 합들 중 n_m개의 최솟값을 찾아야 한다. 기존의 EMAS는 $n_m{}^2$개의 교차 합 전체를 고려하여 연산하기 때문에 체크 노드는 $O(n_m{}^2)$의 복잡도를 갖는다. Bubble check 알고리즘은 분류해야 될 교차 합의 범위를 줄이고, 최솟값을 찾기 위한 분류기의 길이를 줄여 체크 노드 연산의 복잡도를 $O(n_m \sqrt{n_m})$까지 줄인다[12]. Bubble check 알고리즘은 수식 (3)을 따라 두 입력 U_1과 U_2의 교차 합으로 구성된 가상 행렬 T_Σ위에서 n_m개의 최솟값을 찾는다.

$$T_\Sigma(i_b, j_b) = U_1(i_b) + U_2(j_b) \quad (3)$$
$$where, i_b \in [1, n_m], j_b \in [1, n_m]$$

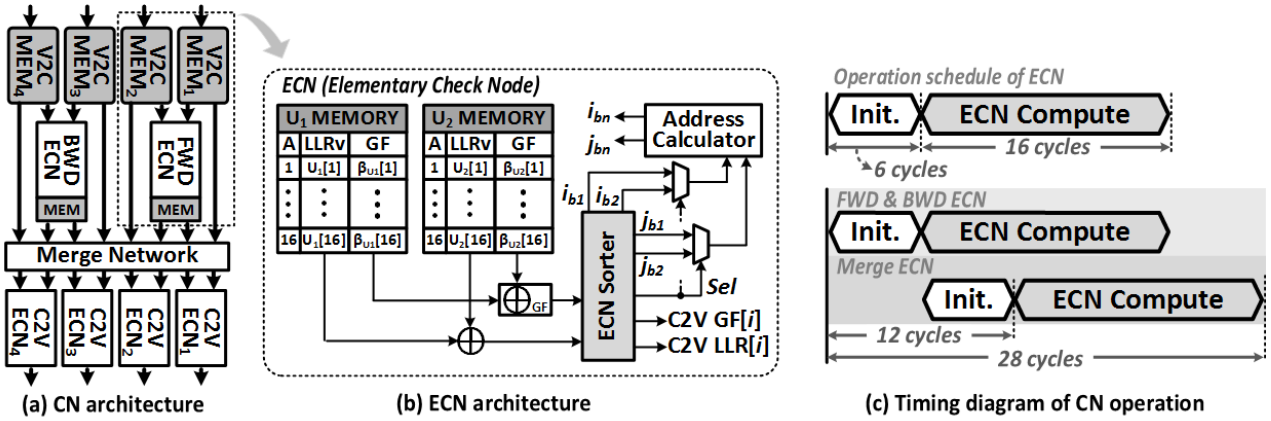


그림 3. (a) 체크 노드 아키텍처 (b) ECN 아키텍처 (c) 체크 노드 연산의 타이밍 다이어그램
Fig. 3. (a) check node architecture (b) ECN architecture (c) timing diagram of check node operation.

Bubble check 알고리즘에서 버블 (bubble)이란 행렬 T_{Σ}의 원소 중 n_b개의 최소 후보 군을 의미하며 n_b의 길이는 n_m에 따라 정해진다. n_m이 16일 때 버블의 수 n_b는 6으로 결정된다[12]. Bubble check 알고리즘 기반의 ECN은 n_b개의 버블 중 하나의 최솟값을 찾기 위한 분류기(sorter)와 다음 버블의 주소인 i_b와 j_b를 계산하기 위한 주소계산기로 구성된다. 두 입력 U_1과 U_2 각각의 원소를 하나씩 읽어와 둘을 더한 값이 분류기에 채워지며 이는 주소계산기의 값이 갱신될 때마다 수행된다. 현재 LLR값이 가장 작은 버블의 주소는 i_{b1}와 j_{b1}으로 다음 버블의 주소는 i_{bn}와 j_{bn}로 정의되며 flag F와 현재 버블의 주소에 따라 다음 버블의 주소가 결정된다.

다. 체크 노드 구현

본 논문은 전/후방 알고리즘과 bubble check 알고리즘을 통해 그림 3과 같이 체크 노드와 ECN을 구현하였다. ECN 연산은 크게 분류, 주소 계산 그리고 fetch 단계를 거쳐 수행되며 그림 3의 (b)와 같이 메모리, 분류기 그리고 주소 계산기로 구현된다. 그림 3의 (c)에 나타난 체크 노드 연산의 타이밍 다이어그램과 같이 각각의 ECN은 $n_b = 6$ 사이클 동안 버블을 초기화한 다음 $n_m = 16$ 사이클 동안 ECN 연산을 수행하여 총 22 사이클이 소요된다. 그림 3의 (a)와 같이 $FWD(\text{forward})$ ECN과 BWD $(backward)$ ECN은 병렬로 처리되어 22 사이클 동안 전/후방 연산을 수행하며 C2V $ECN_{1\sim4}$는 FWD와 BWD ECN의 초기화가 끝난 직후 시작하여 22사이클 동안 조합 연산을 수행한다. 체크 노드 연산의 총 연산 시간은 전/후방 연산 초기화 6사이클에 조합 연산 초기화 및 ECN 연산 22 사이클을 더해 총 28 사이클이 소요된다. U_1메모리와 U_2메모리는 16개의 LLR들과 16개의 GF 인덱스들을 저장한다. LLR은 5-bit로 양자화 되어 있으며 GF 인덱스는 6-bit의 GF 원소이다. 각각의 ECN은 그림 3의 (b)와 같이 LLR 합을 구하기 위한 가산기와 GF 인덱스 합을 구하기 위한 GF 가산기를 포함한다. 본 논문은 임계 경로를 줄이기 위해 다음 버블을 pre-fetch하여 분류와 fetch를 병렬로 처리한다. i_{b2}과 j_{b2}는 두 번째 최소 LLR의 주소이고 Sel은 pre- fetch된 버블과 분류된 버블을 비교한 결과이다. 주소계산기는 i_{b1}와 j_{b1}대신 멀티플렉서의 출력을 이용하여 다음 주소를 계산한다.

2. Interleaving 가능한 변수노드 설계

본 절에서는 변수 노드 연산의 기본 연산 단위인 elementary variable node(EVN)을 두 단계로 나누어 처리하는 two-pass scan EVN과 경판정에 의한 변수 노드 연산 지연을 줄이기 위한 변수 노드 아키텍처에 대해 설명한다. Two-pass scan EVN을 사용한 EVN은 연산을 두 단계로 나누어 처리함으로써 체크 노드 연산과 interleaving 가능한 구조이다.

가. Two-pass scan EVN 연산

EVN은 변수 노드를 구성하는 기본 연산 블록으로 체크 노드로부터 L과 V을 입력 받아 U를 계산한다. 출력 U는 수식 (4)를 만족하는 집합 중 n_m개의 최소 LLR들과 그에 상응하는 GF 인덱스들로 구성된다.

$$L(i_v)+V(j_v), L^{GF}(i_v)=V^{GF}(j_v) \quad where, i_v \in [1,n_m], j_v \in [1,n_m] \tag{4}$$

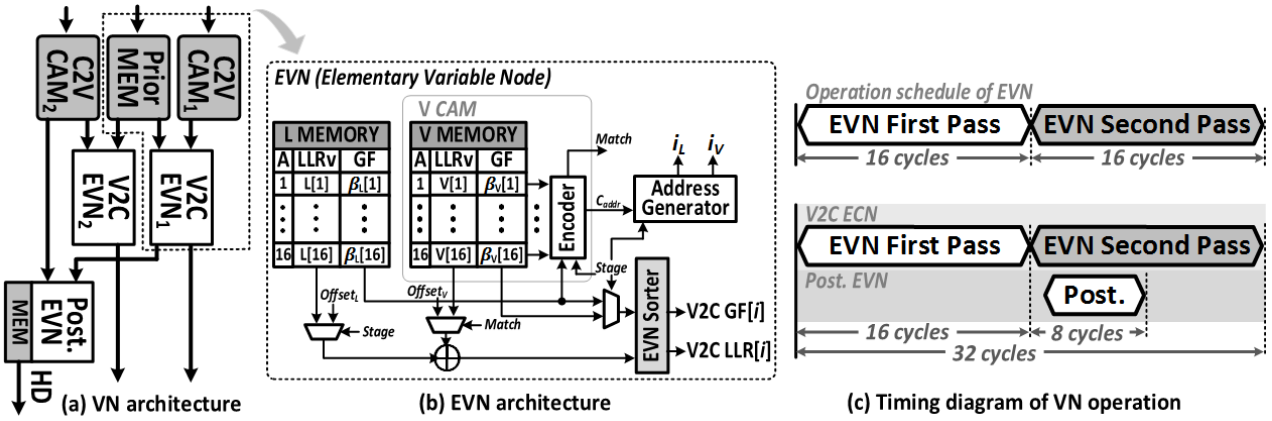


그림 4. (a) 변수 노드 아키텍처 (b) EVN 아키텍처 (c) 변수 노드 연산의 타이밍 다이어그램
Fig. 4. (a) variable node architecture (b) EVN architecture (c) timing diagram of variable node operation.

수식 (4)의 GF 인덱스 매칭 $L^{GF}(i_v) = V^{GF}(j_v)$ 을 만족하는 GF 인덱스를 찾기 위해 입력 중 하나인 V는 contents addressable memory (CAM)에 저장된다. 또한 CAM을 통해 매칭된 합들 중 n_m 개의 최소 합을 찾기 위해서 분류기가 필요하다. 분류기의 크기는 에러 정정 성능에 직접적으로 영향을 주며 분류기의 길이는 적어도 n_m+1이 되어야 성능의 저하가 없다[15]. 본 논문은 성능 저하를 피하기 위해서 분류기의 길이를 17로 정하였다.

Two-pass scan 기법은 EVN을 매칭 단계와 교차 합 분류 단계로 나누어 처리한다. 첫 번째 단계는 메모리를 스캔 하여 $L(i_v)$와 $L^{GF}(i_v)$을 읽어 $L^{GF}(i_v)$와 매칭되는 GF 인덱스 $V^{GF}(j_v)$를 CAM에서 찾는다. 만약 매칭되는 $V^{GF}(j_v)$을 찾으면 $L(i_v)+V(j_v)$을 분류기에 삽입하고 그렇지 않으면 V의 offset과 $L(i_v)$을 더해 분류기에 삽입한다. 두 번째 단계에서는 CAM을 스캔 하여 $V(j_v)$와 L의 offset을 더해 분류기에 삽입하고 매 사이클 분류기에서 최솟값을 분류해 V2C LLR[i]와 V2C GF[i]을 출력한다. 두 번째 단계에서 매 사이클 하나의 V2C entry가 출력되기 때문에 두 번째 단계가 수행되는 동안 체크 노드 연산을 동시에 수행할 수 있다. 이처럼 two-pass scan ECN은 변수 노드 연산에 체크 노드 연산을 interleaving하여 복호 시간을 단축시킨다.

나. 변수 노드 구현

EVN은 메모리와 CAM, 분류기 그리고 주소 생성기로 구성되어 있으며, EVN의 입력 중 하나는 채널 메시지 λ_j이고 다른 하나는 체크 노드로부터 전달은 메시지 $V_{i,j}$이다. 변수 노드에서 두 개의 V2C EVN은 병렬로 처리되어 그림 4의 (c)의 EVN 연산 스케줄과 같이 첫 번째 단계에 16 사이클 두 번째 단계에 16 사이클이 소요되어 총 32 사이클이 소요된다. 그림 4의 (a)와 같이 사후확률을 구하기 위한 $Post.(Posteriori)\,EVN$은 V2C EVN의 출력과 분리하여 사후확률 계산에 의한 복호지연을 제거하였다. $Post.\ EVN$은 V2C EVN의 첫 번째 단계가 끝나고 1사이클의 쓰기 지연 이후에 시작하여 경판정 (hard decision, HD)값을 결정하기 까지 8사이클이 소요된다. EVN은 그림 4(b)와 같이 CAM, 분류기 그리고 주소계산기로 구성되어있다. 인덱스 매칭의 상태를 나타내는 $Match$신호와 현재 단계를 나타내는 $Stage$신호를 통해 LLR을 선택하고 $Stage$신호와 매칭된 내용의 주소를 나타내는 C_{addr}을 통해 다음 메모리의 주소 i_L과 i_V를 계산한다. 메시지 L과 V의 offset $Offset_L$과 $Offset_V$는 각각 메시지의 최대 LLR 값인 $L(16)$과 $V(16)$이다.

Ⅳ. 높은 처리량을 갖는 복호기 설계

본 논문은 복호기를 fully-parallel 아키텍처로 설계하고 two-pass EVN 기반의 변수 노드를 통해 체크 노드 연산과 변수 노드 연산을 interleaving하여 높은 처리량을 갖는 비이진 LDPC 코드 복호기를 설계하였다. 또한 체크 노드에서 큰 지연을 유발하는 초기화 지연을 단축할 수 있는 조기 분류 기법을 제안하여 데이터 처리량을 추가로 향상시켰다.

가. Interleaving을 적용한 Fully-parallel 아키텍처

비이진 LDPC 코드는 코드길이가 짧고 체크 노드 차수와 변수 노드 차수가 작아 배선 복잡도가 비교적 작

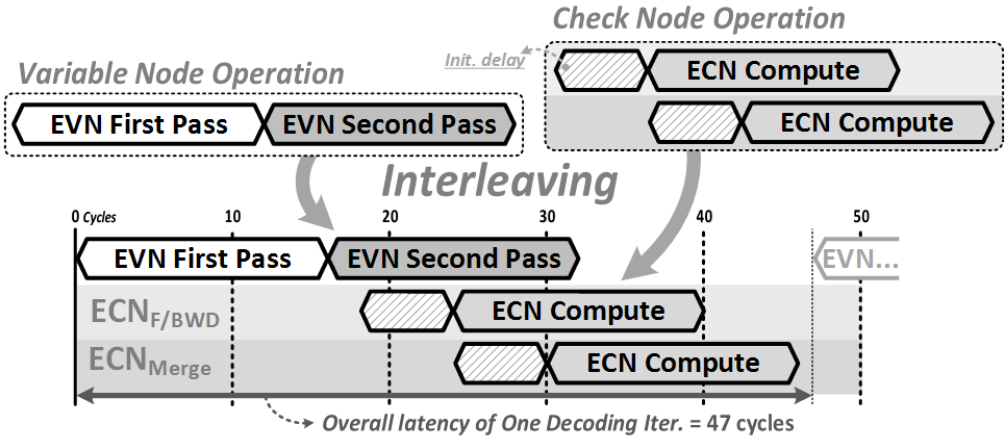


그림 5. 체크 노드와 변수 노드를 interleaving한 디코더의 타이밍 다이어그램
Fig. 5. Timing diagram of the decoder using the interleaving check node and variable node.

아 병렬 구조로 구현하기에 적합하다. 병렬 구조는 데이터 처리량을 향상 시킬 뿐만 아니라 복호의 컨트롤이나 메시지 스케줄을 간소화하여 효율적으로 복호기를 설계할 수 있다[13~14]. 본 논문은 복호기를 fully-parallel 아키텍처로 구현하여 높은 데이터 처리량을 갖는 복호기를 설계하였다.

길이가 짧고 체크 노드 차수와 변수 노드 차수가 작아 병렬 아키텍처를 구현하는데 용이하다. 그림 14과 같이 병렬 비이진 LDPC 코드 복호기는 160개의 변수 노드와 80개의 체크 노드로 구성되어있으며 (2,4)구조에 따라 각 변수 노드는 2개의 체크 노드와 각 체크 노드는 4개의 변수 노드와 연결되어있다. 각각의 노드들은 매 사이클 하나의 LLR과 GF 인덱스를 전달하며 Tanner 그래프의 edge를 따라 구성된 Routing Network를 통해 이동한다. 변수 노드와 체크 노드 사이에는 변환과 역 변환을 수행하기 위한 변환 노드(Perm. Node)가 있으며 이는 간단한 LUT를 통해 구현된다. 메시지를 구성하는 LLR의 양자화 bit는 좋은 성능을 보장하고 메모리 요구량을 줄이기 위해 [15]의 모의실험 결과에 따라 5-bit로 정하였다. 복호 알고리즘은 EMSA를 사용하며 bubble check 알고리즘과 two-pass EVN을 사용하여 체크 노드와 변수 노드를 구현하였다.

Two-pass scan EVN은 연산을 두 단계로 나누어 처리하기 때문에 ECN 연산과 interleaving 가능하다. 그림 5는 EVN 연산과 ECN 연산이 interleaving된 비이진 LDPC 코드의 복호 연산 스케줄을 나타낸다. EVN 연산의 두 번째 단계가 시작하면 매 사이클 하나의 V2C entry를 생성하기 때문에 ECN 연산을 바로 수행할 수 있다. 그림 5와 같이 EVN second pass에서 첫 번째 LLR이 계산되면 1사이클의 쓰기 지연 이후 F/BWD ECN이 초기화를 시작한다. F/BWD ECN의 초기화가 끝나면 C2V ECN이 초기화 단계를 거친 뒤 ECN 연산 단계에서 매 사이클 하나의 C2V entry를 출력한다. C2V ECN의 연산이 끝나면 1사이클의 쓰기 지연 이후에 다음 iteration의 EVN 연산이 시작된다. 그림 5와 같이 interleaving을 적용한 복호기의 1 iteration 복호 연산 시간은 총 47 사이클이다.

나. 조기 분류 기법

Bubble check 알고리즘은 체크 노드 연산의 복잡도를 낮추었지만 항상 n_b개의 버블을 비교하여 최소값을 분류하기 때문에 n_b사이클의 초기화 시간을 갖는다. 전/후방 알고리즘을 사용하는 체크 노드의 경우 (d_c-2) 단계의 ECN 연산을 수행하기 때문에 총 $n_b(d_c-2)$ 사이클의 초기화 시간이 필요하다. 이 절에서는 체크 노드 연산에서 큰 연산 지연을 유발하는 초기화 지연을 단축할 수 있는 조기 분류 기법을 제안한다.

ECN의 두 입력이 오름차순으로 정렬된 경우 가상 행렬 T_Σ는 다음 성질 1을 만족한다[12]. 성질 1: $i_b, j_b, i_b', j_b' \in \{x | 1 \le x \le n_m\}$ 일 때, 만약 $i_b \le i_b'$ 이고 $j_b \le j_b'$이면 $T_\Sigma(i_b, j_b) \le T_\Sigma(i_b', j_b')$ 이다. 이때, i_b, j_b, i_b', j_b'는 가상 행렬의 행과 열의 index이다. 가상 행렬 T_Σ는 식 (3)을 따라 두 입력 U_1과 U_2의 교차 합으로 구성되며 T_Σ의 i_b번째 열과 j_b번째 행의 원소를 $T_\Sigma(i_b, j_b)$로 정의한다. 성질 1을 통해 가상 행렬 T_Σ의 원소 $T_\Sigma(i_b, j_b)$는 i_b와 j_b가 작을수록 작은 값임을 알 수 있다. 이때 두 입력 U_1과 U_2가 오름차순으로 정

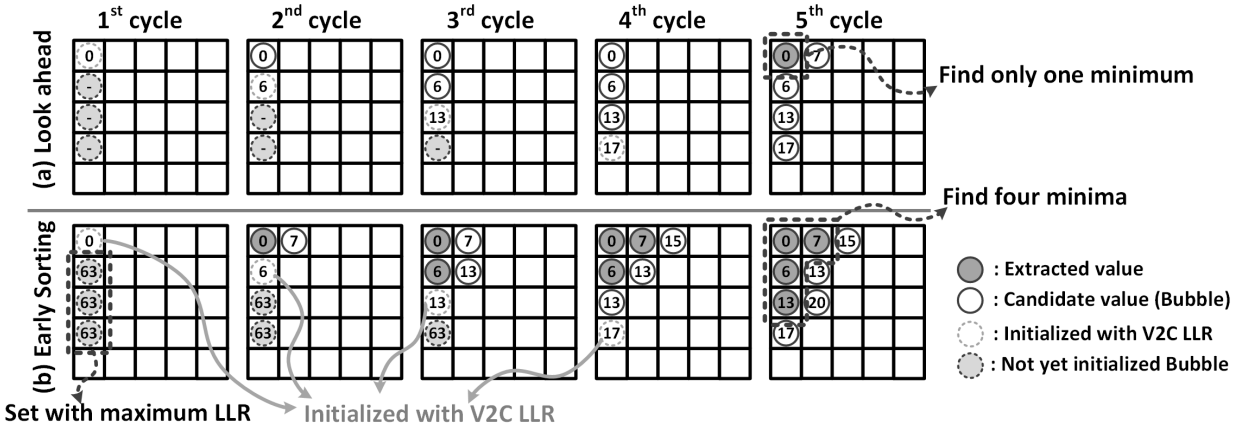


그림 6. (a) bubble check 알고리즘과 (b) 조기 분류 기법을 사용한 ECN 연산 예제
Fig. 6. Example of ECN operation using (a) bubble check algorithm and (b) early sorting.

렬되어 있을 경우 다음 성질 2를 만족한다. 성질 2: $1 \le x \le l$, $x \in (x|i_b,j_b)$ 일 때, $T_\Sigma(i_b,j_b)$는 적어도 l 개의 최소 LLR을 포함한다. 성질 2를 따르면 두 입력 U_1과 U_2에서 l개의 LLR들이 계산된 상태라면 ECN은 적어도 l개의 최소 LLR을 분류할 수 있다. 본 논문은 성질 2를 통해 ECN은 n_b개의 버블이 전부 초기화 되지 않은 상태에서도 ECN 연산을 미리 수행할 수 있는 조기 분류 기법을 제안한다. 조기 분류 기법은 알고리즘 2를 따르며 ECN 연산의 초기화 지연을 단축시킨다.

그림 6은 n_b가 4일 때의 bubble check 알고리즘과 조기 분류 기법의 ECN의 연산과정을 나타낸다. 기존의 bubble check 알고리즘은 성질 1을 이용해 최솟값의 후보를 결정하여 그림 6의 (a)와 같이 $i_b = 1, j_b \le 4$을 만족하는 $T_\Sigma(i_b,j_b)$로 bubble을 초기화하며 bubble의 값이 모두 결정된 후에 최솟값을 분류한다. Bubble의 수 n_b이 4인 경우, bubble check 알고리즘 기반의 ECN은 bubble을 초기화하는데 4사이클이 소요된다. 본 논문에서 제안된 조기 분류 기법을 사용한 ECN은 그림 6의 (b)와 같이 아직 계산되지 않은 bubble을 최대 LLR 값으로 초기화한다. 두 입력 U_1과 U_2가 오름차순으로 입력되며 성질 2를 이용하면 하나의 bubble만 결정된 상태에서도 분류를 시작할 수 있다. 조기 분류 기법은 1사이클의 초기화 지연 이후에 바로 ECN 연산을 시작하기 때문에 그림 6의 (b)와 같이 5번째 사이클에서 4개의 최솟값을 찾을 수 있다.

조기 분류 기법은 ECN의 초기화 지연시간을 n_b사이클에서 1 사이클로 단축시킨다. 기존의 Bubble check를 사용한 체크 노드는 그림 7의 (a)와 같이 총 12 사이클의 초기화 지연 이후 16 사이클 동안 ECN 연산을 수행

표 2. ECN 연산 지연을 줄이기 위해 제안된 알고리즘
Table 2. Proposed algorithm to reduce the latency of ECN.

알고리즘 2. 조기 분류 기법

Step 1. 분류기 초기화

분류횟수 $k=1$, 초기화 상태 $l=1$로 설정

분류기를 $T_\Sigma(i_b,1)$로 초기화. 이때 $i_b=1,\cdots,l$

비어있는 i_b-l개의 버블은 최대 LLR값으로 초기화

Step 2. 분류

현재 값이 가장 작은 버블을 출력으로 내보내고, 그 버블의 주소를 i_{b1}과 j_{b1}으로 한다.

Step 3. Flag 계산

만약 $i_{b1}=1$이면, $F=1, \overline{F}=0$ 그렇지 않고 만약

$j_{b1}=1$이고 $i_{b1} \ge n_b$이면, $F=0, \overline{F}=1$

Step 4. 다음 주소 계산

만약 $T_\Sigma(i_{b1}+\overline{F}, j_{b1}+F)$가 참조되었다면 $i_{bm}=i_{b1}+F$,

$j_{bm}=j_{b1}+\overline{F}$ 그렇지 않으면 $i_{bm}=i_{b1}+\overline{F}$, $j_{bm}=j_{b1}+F$

Step 5. 다음 버블 fetch 및 분류 횟수 확인

만약 $l<n_b$이면, $T_\Sigma(i_{bm},j_{bm})$을 분류기에 채운 다음

$l=l+1$, $k=k+1$로 설정하고 Step 1로 이동, 그렇지 않고

만약 $k<n_m$이면, $T_\Sigma(i_{bm},j_{bm})$을 분류기에 채우고

$k=k+1$로 설정하고 Step 2로 이동, 그렇지 않으면 종료

하여 총 28 사이클의 체크 노드 연산 시간을 갖는다. 조기 분류 기법을 적용한 본 논문의 체크 노드는 그림 7의 (b)과 같이 각 단계의 ECN 초기화 지연을 5 사이클씩 단축하여 체크 노드 연산 시간을 18 사이클로 단축하였다. 그림 8은 조기 분류 기법을 적용한 복호기의 타이밍 다이어그램을 나타낸다. 조기 분류 기법을 적용한 본 논문의 복호기는 체크 노드의 초기화 지연을 10 사이클 줄여 1 iteration의 복호 시간을 37 사이클로 단축하였다. 표 3은 조기분류기법을 사용한 복호기와

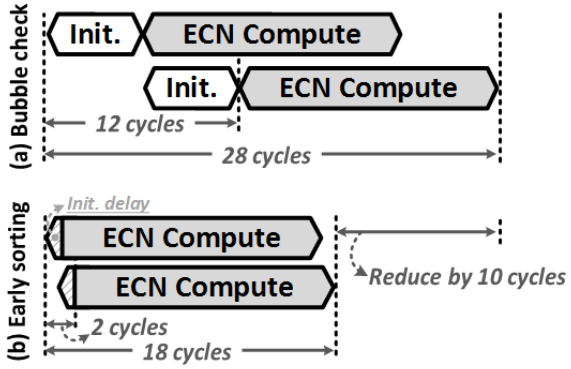


그림 7. (a) bubble check 알고리즘과 (b) 조기 분류 기법을 사용한 ECN의 연산 스케줄

Fig. 7. Operation schedule of ECN using (a) bubble check algorithm and (b) early sorting.

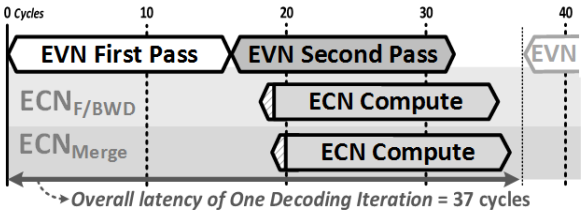


그림 8. 제안된 조기 분류 기법을 적용한 복호기의 타이밍 다이어그램

Fig. 8. Timing diagram of the decoder using proposed early sorting.

bubble check 알고리즘을 사용한 복호기의 측정 결과를 요약한 표이다. Synopsys사의 PrimeTime PX을 통해 전력을 측정한 결과 조기분류기법을 사용한 복호기의 전력증가율은 0.8%이다. 조기분류기법을 사용할 경우 소비전력 측면에서는 작은 오버헤드가 있지만 복호기의 데이터처리량이 27.1%증가하여 소비에너지는 20.6% 감소하였다.

V. 복호기 성능 및 구현결과

본 논문은 조기 분류 기법을 적용한 fully-parallel 아키텍처를 제안하여 높은 처리량을 갖는 비이진 LDPC 코드 복호기를 구현하였다. 이 장에서는 모의실험을 통해 제안하는 복호기의 에러 정정 성능을 평가하고 하드웨어 구현 결과 및 성능을 나타낸다.

그림 9는 조기 분류 기법을 적용한 복호기의 에러 정정 성능을 나타낸다. 모의실험은 GF(32)와 GF(64)에서 정의된 두 가지 (160,80) 정규 (2,4) 비이진 LDPC 코드를 사용하였으며 메시지 절단 길이 n_m은 각각 10과 16이다. 본 논문은 [9]의 binary image를 통해 정해진 GF

표 3. 측정 결과 요약

Table 3. Measurement summary.

	동작주파수 [MHz]	소비전력 [mW]	처리량 [Mbps]	소비에너지 [nJ/b]
Bubble check 알고리즘	625	1683 (100%)	1103 (100%)	1.53 (100%)
조기분류기법	625	1695 (100.7%)	1402 (127.1%)	1.21 (79.2%)

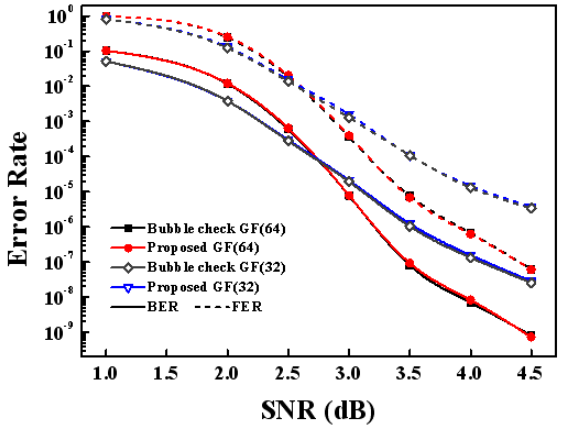


그림 9. 제안된 복호기의 error rate 성능

Fig. 9. Error rate performance of proposed decoder.

원소와 [10]의 Progressive Edge-Growth 알고리즘에 기반한 parity check matrix를 사용하였다. EMSA 복호 알고리즘을 사용하며 양자화 비트는 5-bit이고 최대 반복횟수는 30이다. 모의실험은 binary phase shift keying(BPSK)를 사용하며 additive white Gaussian noise(AWGN) 채널 환경에서 수행되었다. 실험결과 조기 분류 기법의 에러 정정 성능은 GF(32) 코드와 GF(64) 코드에서 모두 bubble check 알고리즘을 사용한 복호기와 거의 유사한 성능을 발휘한다. 조기 분류 기법은 체크 노드의 기본 연산 블록인 ECN의 초기화 시간을 단축하는 알고리즘으로 체크 노드의 차수와 관계없이 구현되기 때문에 다양한 코드에 적용가능하며 그림 9의 모의실험 결과와 같이 GF(64) 외의 차원에서도 적용가능하다. 그림 9의 결과와 같이 조기 분류 기법은 에러 정정 성능 저하 없이 체크 노드 연산의 지연시간을 단축할 수 있다.

본 논문은 조기 분류 기법을 적용한 fully-parallel 아키텍처를 사용하여 높은 처리량을 갖는 EMSA 기반 GF(64) (160,80) 정규 (2,4) 비이진 LDPC 코드 복호기를 제안한다. 제안하는 복호기는 Verilog HDL로 기술되었으며 Synopsys Design Compiler를 통해 65nm CMOS cell library를 사용하여 합성되었다. 표 4는 제

표 4. 제안된 복호기와 최근 연구된 복호기의 구현 결과
Table 4. Implementation result of proposed decoder and state-of-the-art decoder.

	This work	JSSC '15 [15]	TCAS I '15 [16]	TCAS I '12 [7]
코드	(160,80)	(160,80)	(837,726)	(837,726)
유한체	GF(64)	GF(64)	GF(32)	GF(32)
알고리즘	EMSA	EMSA	T-MSM	SMSA
공정 (nm)	65	65	90	180
게이트 수 (NAND)	2.82M	2.78M	2.07M	1.29M
주파수 (MHz)	625	700	250	200
처리량 (Mbps)	1402	1221	729	64
처리량* (Mbps)	1402	1221	1009	177
면적 효율* (Mbps/K-gate)	0.497	0.439	0.488	0.137

* 65nm 이외의 공정은 scale factor에 따라 65nm 공정으로 조정되었다.

안된 복호기와 최근 발표된 비이진 LDPC 복호기의 구현 결과를 나타낸다. 65nm 공정을 사용하지 않은 복호기는 scale factor에 따라 65nm로 조정되었다.

최근 연구 결과 중[7]은 EMS 복호 알고리즘을 간소화한 simplified min-sum algorithm(SMSA)을 사용하여 메모리의 크기를 줄여 복호기의 크기가 비교적 작지만 에러 정정 성능 저하가 두드러진다. [16]은 체크 노드 메시지를 통계적으로 분석하여 두 번째 최솟값을 추정하는 one minimum only(OMO) 알고리즘을 제안하며 OMO 알고리즘을 trellis min-sum algorithm(T-MSA)와 trellis min-max algorithm (TMMA)에 적용하여 데이터 처리량을 올리고 면적을 감소시켰다. [16]에서 사용하는 trellis 알고리즘은 병렬 처리를 통해 체크 노드의 처리속도를 향상시킬 수 있지만 GF의 위수에 따라 면적이 크게 증가한다. EMSA를 간소화한 MMA는 EMSA에 비해 비교적 작은 복호 복잡도를 갖지만 에러 정정 성능 저하가 두드러지는 단점이 있다. [15]은 EMSA 기반 복호기를 구현하였으며 look ahead 기법을 통해 임계경로를 단축하고 two pass EVN 기법을 활용한 interleaving을 통해 데이터 처리량을 향상시켰다.

본 논문의 복호기는 EMSA 기반의 fullty-parallel 아키텍처를 사용하며 625MHz 동작주파수로 합성했을 때 2.82M의 게이트 수로 구현된다. 제안된 복호기는 조기 분류 기법을 통해 1 iteration의 복호시간을 37 사이클로 단축하여 1402Mbps의 데이터 처리량을 달성하였다. 제안된 복호기는 [7,15~16]와 비교했을 때 데이터 처리량이 가장 높으며 하드웨어 오버헤드 거의 없이 전체 복호 시간을 단축하여 우수한 면적 효율을 가진다. 또한 복호 성능이 뛰어난 EMS 복호 알고리즘과 위수가 큰 유한체에서 정의된 코드를 사용하여 비해 뛰어난 에러 정정 성능을 발휘한다.

VI. 결 론

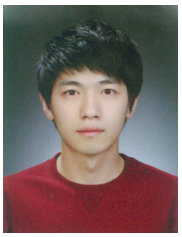
본 논문은 GF(64) (160,80) 정규 비이진 LDPC 코드 복호기를 위한 높은 처리량의 병렬 아키텍처를 제안한다. 제안된 복호기는 Fully-parallel 아키텍처를 사용하고 체크 노드와 변수 노드를 interleaving하여 복호기의 데이터 처리량을 향상시켰다. 또한 체크 노드의 초기화 지연을 단축시킬 수 있는 조기 분류 기법을 제안하여 데이터 처리량을 추가로 향상시켰다. 제안된 복호기는 1 iteration에 37사이클이 소요되며 625MHz 동작주파수에서 1402Mbps의 데이터 처리량을 갖는다. 본 논문은 조기 분류 기법을 적용한 fully-parallel 아키텍처를 사용하여 하드웨어 오버헤드 거의 없이 높은 처리량을 갖는 비이진 LDPC 코드 복호기를 제안한다.

REFERENCES

[1] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, “Design of capacity-approaching low-density parity check codes”, IEEE Trans. Inf. Theory, vol. 47, pp. 619-637, Feb. 2001
[2] M. C. Davey and D. MacKay, “Low-density parity check codes over GF(q)”, IEEE Commun. Lett., vol. 2, no. 6, pp. 165 - 167, Jun. 1998.
[3] L. Barnault and D. Declercq, “Fast decoding algorithm for LDPC over GF(q^2)”, in Proc. IEEE Inf. Theory Workshop, pp. 70 - 73, Mar. 2003
[4] H. Wymeersch, H. Steendam, and M. Moeneclaey, “Log-domain decoding of LDPC codes over GF(q)”, in Proc. IEEE Int. Conf. Commun., vol. 2, pp. 772 - 776, Jun. 2004
[5] D. Declercq and M. Fossorier, “Decoding algorithms for nonbinary LDPC codes over GF(q)”, IEEE Trans. Commun., vol. 55, no. 4, pp. 633 - 643, Apr. 2007.
[6] V. Savin, “Min-max decoding for non binary LDPC codes”, in Proc. IEEE Int. Symp. Inf. Theory, pp. 960 - 964, Jul. 2008
[7] X. Chen and C.-L. Wang, “High-throughput efficient non-binary LDPC decoder based on the simplified min-sum algorithm, IEEE Trans”. Circuits Systems I: Regular Papers, vol. 59, no. 11, pp.2784 -2794, Nov. 2012
[8] E. Li, K. Gunnam, and D. Declercq, “Trellis based extended min-sum for decoding nonbinary LDPC codes”, in Proc. 8th Int. Symp. Wireless

Commun. Syst. (ISWCS, pp. 46 - 50), Nov. 2011
[9] C. Poulliat, M. Fossorier and D. Declercq, "Design of regular ($2,d_c$)-LDPC codes over GF(q) using their binary images", vol. 56, no. 10, pp.1626-1635, Oct. 2008
[10] A. Venkiah, D. Declercq and C. Poulliat, "Design of Cages with a Randomized Progressive Edge-Growth Algorithm", in IEEE Commun. Letters, vol. 12, pp. 1-3, Apr. 2008.
[11] A. Voicila, D. Declercq, F. Verdier, M. Fossorier and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields", IEEE Trans. Commun., vol. 58, no. 5, pp. 1365-1375 May 2010
[12] E. Boutillon and L. Conde-Canencia, "Bubble check: A simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders", IEEE Electron. Lett., vol. 46, no. 9, pp.633 - 634, Apr. 2010.
[13] Y. Jung, Y. Jung, S. Lee, and J. Kim, "8.1 Gbps High-Throughput and Multi-Mode QC-LDPC Decoder based on Fully Parallel Structure", Journal of The Institute of Electronics Engineers of Korea, vol. 50, no. 10, pp. 78-89, Oct. 2013
[14] J. Park, S. Lee, K. Chung, S. Cho, J. Ha, and Y. Song, "A Memory-efficient Partially Parallel LDPC Decoder for CMMB Standard", The Institute of Electronics Engineers of Korea - Semiconductor and Devices vol. 48, no. 1, pp. 22-30, jan. 2011
[15] Y. S. Park, Y. Tao and Z. Zhang, "A fully parallel nonbinary LDPC decoder with fine-grained dynamic clock gating", IEEE J. Solid-State Circuits, vol. 50, no. 2, pp.464 -475, Feb. 2015
[16] J.O. Lacruz, F.G. Herrero, J. Valls and D. Declercq, "One Minimum Only Trellis Decoder for Non-Binary Low-Density Parity-Check Codes", IEEE Trans. Circuits Systems I: Regular Papers, vol. 62, no. 1, pp. 177 - 184, Jan. 2015.

저 자 소 개



최 인 준(학생회원)
2014년 충남대학교 전자공학과 학사 졸업
2016년 충남대학교 전자전파정보통신공학과 석사 졸업.
<주관심분야 : 신호처리 및 보안 SoC 설계>

김 지 훈(정회원)
2004년 KAIST 전자전산학과 학사 졸업.
2009년 KAIST 전기 및 전자공학과 박사 졸업.
2009년 7월 ~ 2010년 2월 삼성전자 DMC연구소 책임연구원
2010년 3월 ~ 2016년 2월 충남대학교 전자공학과 부교수
2016년 2월 ~ 현재 서울과학기술대학교 전기정보공학과 부교수
<주관심분야 : CPU 코어 설계, SoC 설계>