논문 2010-47SD-5-6

비휘발성 메모리를 위한 병렬 BCH 인코딩/디코딩 방법 및 VLSI 설계

(Parallel BCH Encoding/decoding Method and VLSI Design for Nonvolatile Memory)

이 상 혁*, 백 광 현**

(Sang-Hyuk Lee and Kwang-Hyun Baek)

요 약

본 논문에서는 SSD (solid state disk)에 쓰이는 NAND flash 메모리 에러 정정에 관한 오류정정 방법 중에서 Parallel BCH (Bose-Chaudhuri-Hocquenghem) 방법 및 VLSI 설계를 제안하였다.

제안된 설계는 에러 정정 능력(t=18, 8) 을 가변적으로 하여 사용빈도수의 증가로 높은 에러 율을 가진 데이터 공간에 신뢰성을 높였고, 디코더의 병렬처리 비트 수를 인코더의 처리 비트 수에 2배로 하여 디코더의 수행시간을 줄였고 이에 따르는 latency도 기존 회로에 비해 1/2로 감소함을 확인 하였다.

Abstract

This paper has proposed parallel BCH, one of error correction coding methods which has been used to NAND flash memory for SSD(solid state disk).

To alter error correction capability, the proposed design improved reliability on data block has higher error rate as used frequency increasingly. Decoding parallel process bit width is as two times as encoding parallel process bit width, that could reduce decoding processing time, accordingly resulting in one half reduction over conventional ECC.

Keywords: VLSI, BCH, 부호이론, SSD (Solid State Disk)

I. 서 론

최근 하드 디스크를 대체하는 SSD (solid state disk) 시장이 빠르게 성장하고 있다. SSD의 특성은 하드디스크와 비교 할 때 저 전력이고 고속이며 또한 데이터의 보안성이 우수한 장점을 가지고 있다. 이러한 장점 때문에 고성능 서버나 SAN (Storage Area Network)같은 장비 분야에서 SSD의 사용이 늘고 있다.

개인 사용자 환경에서도 앞에서 언급한 이유로 SSD를 채택하는 경우가 늘어나게 되었지만, 저장장치에 사용되는 메모리-SLC(single layer cell), MLC(multi layer cell)는 데이터의 읽기/쓰기 횟수가 늘어날수록 오류발생 가능성이 점차 높아지는 단점을 가지고 있다. 따라서 SSD에서는 이러한 오류율을 줄일 수 있는 오류정정 알고리즘의 중요성이 높아지고 있다. 본 논문에서는 이러한 SSD에서 발생되는 오류를 정정하는 방법을 제안하였다.

통상적으로 비휘발성 메모리나 자기디스크에서 오류를 검출 및 보정은 에러 정정 코딩 (error correction coding) 방식을 사용한다. 메모리의 에러정정방식은 주로 블록 코딩이 쓰이는데 순회 부호인 블록코딩은 좋은

* 학생회원, ** 평생회원, 중앙대학교 전기전자공학부
(School of Electrical and Electronics Engineeing, Chung-Ang University)

※ 이 논문은 2007년도 중앙대학교 학술연구비 (일반연구비) 지원에 의한 것임

접수일자: 2009년11월18일, 수정완료일: 2010년3월22일

연집 (burst) 오류검출 특성이 있다.

대표적인 블록 코딩 기법은 RS (Reed-Solomon) 코드와 BCH (Bose-Chaudhuri-Hocquenghem) 코드[1~2] 등이 있다. RS code는 연집된 오류에 좋은 성능이지만 산발적인 오류에는 불리하기 때문에 BCH 코드가 메모리의 에러정정기법으로 많이 쓰이게 되었다.

BCH는 멀티 비트 에러 정정을 위해 쓰이고 설계자의 필요에 따라 두 개의 파라미터 (n :부호의 길이, t : 에러정정능력)를 통해 쉽게 정의 할 수 있다.

본 논문에서 제안한 설계는 flash 메모리 콘트롤러의 ECC에 중점을 두며 신뢰성과 내구성을 높이기 위해서 SLC/MLC 경우에는 8 bit 에러정정, 사용빈도가 높아 에러율이 높은 MLC의 경우는 18 bit 에러 정정을 선택할 수 있게 하였다.

본 논문은 인코딩 및 디코딩의 속도를 높이기 위한 병렬 처리 방법을 제안하였고 디코딩 속도를 높이기위해서 unfold 방법을 채용하였다.

본 논문에서는 BCH 오류 검출 및 정정기법을 병렬로 수행함으로써 최대 0.94 Gbit/s 성능 (throughput) 을 가지는 (n=4330, k=4096, t=18 및 n=4200, k=4096, t=8) 저 전력 BCH 인코딩 및 디코딩 알고리즘 구현 및 설계하고 단지 동작 속도만 빠른 것이 아니라 지연시간 (latency) 도 적게 하여 높은 성능 (high throughput) 을 갖는 구조를 제안하였다 (k: 정보의 길이).

그림 1은 병렬 BCH 코덱의 블록도 이다. 병렬 BCH 코덱은 인코더, 디코더, 코딩된 데이터를 저장하는 FIFO (first in first out) Memory와 인코더, 디코더 및 FIFO를 제어하는 Control 블록으로 구성되었다. 인코딩

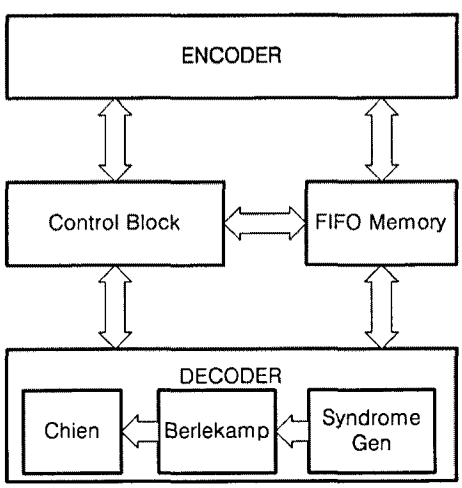

그림 1. BCH 코덱 블록도
Fig. 1. BCH codec block plot.

된 데이터를 프레임 (Frame) 이라고 하고 t=18인 경우 프레임의 크기는 4330 bit 이고 t=8은 4200 bit이고 인코딩, 디코딩 시에는 프레임 단위로 처리한다.

본 논문의 이후의 구성은 다음과 같이 구성이 된다. Section Ⅱ는 인코더 설계, Section Ⅲ는 디코더 설계를 제안하였고 Section Ⅳ에서는 설계된 결과물을 모의실험 및 성능 평가하고 최근 논문과 비교하였다. Section Ⅴ은 결론이다.

Ⅱ. BCH 인코더 설계 (BCH Encoder Design)

인코더는 원시 다항식 (M_1,M_3,...) 의 곱의 형태로 이루어진 생성다항식이다. 생성 다항식의 차수는 t*m (m=원시 다항식의 차수) 이고 식 (1)의 f(x)는 생성다항식을 표현한다.[3]

$$\begin{aligned} M_1(x) &= (x+a)(x+a^2)\cdots(x+a^{2\times(t-1)}) \\ M_3(x) &= (x+a^3)(x+a^6)\cdots(x+a^{3\times 2^{(t-1)}}) \\ &\vdots \qquad\qquad \vdots \\ M_{2m+1} &= (x+a^{2m+1})\cdots(x+a^{(2m+1)\times 2^{(t-1)}}) \end{aligned} \tag{1}$$

$$f(x) = LCM\{M_1(x)M_3(x)M_5(x)\cdots M_{2t-1}(x)\}$$

차수가 k 인 정보 다항식 d(x)에 x^{n-k}를 곱하면 차수가 n인 다항식이 나오는데, d(x)xn-k를 차수가 t*m인 생성 다항식 g(x)로 나누면 어떤 몫과 나머지 q(x)가 나온다. 이 정보다항식에 나머지인 검사다항식인 p(x)를 붙이는 형태를 조직 (systematic) 코드라 하고 식 (2)같이 표현한다.

$$c(x) = d(x) + q(x) \tag{2}$$

식 (2)의 형태로 수식부 (메모리)에 전송이 된다. 이렇게 검사 비트 다항식 p(x)를 구하고 정보다항식 d(x)와 결합하는 형태를 BCH 인코딩이라고 한다.

한 클록 당 한 비트 씩 직렬로 입력 받는 LFSR (linear feedback shift register) 를 이용하여 이진 나눗셈 연산을 하여 인코딩을 수행하지만, 특히 저장매체 분야에서 전송속도와 기록/독출 속도가 Giga bps에 이르므로 현재에는 그 성능 요건을 만족시키기 어려워 졌다. 따라서 한 클록 당 다수의 비트를 동시에 입력받은 결과가 한 비트 씩 직렬로 입력되는 경우와 연산 결과가 동일하도록 병렬 구조로 되도록 하여 요건을 만족시키고 있다. CRC 처리는 기본적인 LFSR 형태로 병렬 회로[4]를 구성 된다.

그림 2는 인코더 블록도이고 인코더의 연산블록은 LFSR Ⅱ를 병렬회로를 구성하여 LFSR 병렬회로 보다 프레임의 처리시간을 단축하였다.

LFSR의 프레임의 처리 시간은 식 (3)이다.

$$T_{LFSR}=\frac{n}{p}\,(cycle) \tag{3}$$

여기서 p는 1 사이클 (클록) 동안 병렬로 처리된 비트수이다. LFSR Ⅱ를 병렬처리 쓰이는 생성다항식은 식 (4)처럼 나타낼 수 있다.

$$g(x)=\sum_{i=0}^{m}w(i)x^i\,(w(i):weight) \tag{4}$$

식 (4)을 이용하여 LFSR Ⅱ를 p bit의 병렬처리 방식으로 수식을 유도하면 식 (5)과 같이 나타 낼 수 있다.

$$\begin{aligned}
U(p-1)&=\sum_{i=0}^{m-1}(d(p-1)*w(i))x^i\\
U(p-1)+U(p-2)&=\sum_{i=0}^{m-1}(d(p-1)*w(i))x^{i+1}\\
&\quad+\sum_{i=0}^{m-1}(d(p-2)*w(i))x^i\\
U(p-1)+\cdots+U(0)&=\sum_{i=0}^{m-1}(d(p-1)*w(i))x^{i+p-1}\\
&\quad+\cdots+\sum_{i=0}^{m-1}(d(0)*w(i))x^i
\end{aligned} \tag{5}$$

$$\begin{aligned}
f_{enc}(x)&=\sum_{\nu=p-1}^{0}U(d(\nu))\\
&=\sum_{i=0}^{m-1}\sum_{j=0}^{p-1}(d(j)*w(i)x^{i+j})
\end{aligned}$$

여기서 $f_{enc}(x)$는 다시 g(x)로 나누어 주면

$$\begin{aligned}
p_{enc}(x)&=f_{enc}(x)/g(x)\\
p_{enc}(x)&=\sum_{i=0}^{t*m-1}p_{enc}(i)x^i
\end{aligned} \tag{6}$$

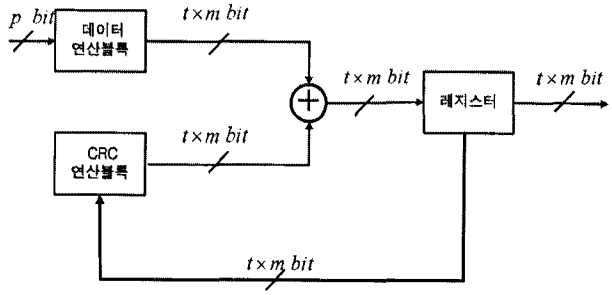

그림 2. 인코더 블록도
Fig. 2. Encoder block plot.

식 (6)에서 구한 p(x)를 이용한 병렬 LFSR Ⅱ의 프레임 처리 시간은 식 (7)과 같이 표현된다.

$$T_{LFSR2}=\frac{k}{p}(cycle) \tag{7}$$

상기에 설명한 부분은 데이터의 연산 블록에 해당한다. CRC 연산 부분은 앞에서 언급한 논문[4]에서 자세히 설명하고 있으므로 생략하고자 한다. k bit 정보 비트가 입력되면 검사비트는 독출해서 정보비트와 함께 송신한다.

Ⅲ. BCH 디코더 설계 (BCH Decoder Design)

디코더는 인코더보다 frame 당 수행시간이 2배 이상 사이클 (클록) 이 필요 하므로 병렬로 처리하는 데이터의 비트수를 $P_{decode}=2*P_{encode}$ 로 하여 설계하고 이에 따라 신드롬 생성기와 병렬 치엔 탐색 블록의 비트 수를 인코더의 2 배로 해서 디코더의 수행시간을 줄였다.

1. 신드롬 생성기 (Syndrome Generator)

신드롬 생성기는 수신된 코드를 원시다항식 (primitive polynomial) 로 나누어서 신드롬 (또는 오증)을 구하는 회로이다. 인코더는 원시다항식의 곱으로 이루어지며 이를 통해 생성된 코드는 다시 원시 다항식으로 나누어떨어진다. 여기서 나머지의 형태를 신드롬이라고 하며 신드롬이 0 이 아닌 경우에 전송된 코드는 오류를 포함함을 알 수 있다.

신드롬 생성기의 경우 LFSR 의 병렬 회로의 형태로 데이터 연산 블록을 구현 했다. 데이터 연산 블록은 $p\le m$ 인 경우에서 수식 (8)과 같이 출력이 0 또는 데이터로 출력이 된다.

$$\begin{aligned}
&\sum_{\nu=p}^{m-1}0*U(\nu)+\sum_{\nu=0}^{p-1}U(\nu)=[0\ \cdots 0|d(p-1)\cdots d(0)]\ \ (p<m)\\
&\sum_{\nu=0}^{p-1}U(\nu)=[d(p-1)\cdots d(1)d(0)]\ \ (p=m)
\end{aligned} \tag{8}$$

하지만 p>m인 경우는

$$\sum_{\nu=p-1}^{p-m}U(\nu)=[d(p-1)\cdots d(p-m)](p-m\le\nu<p) \tag{9}$$

$0\le\nu<p-m$ 영역은 식 (6)과 같은 방식으로 유도하여 $f_{synd}(x)$를 식 (8)과 $0\le\nu<p-m$ 영역에서 유도된 식 (9)의 합으로 식 (10)과 같이 나타 낼 수 있다.

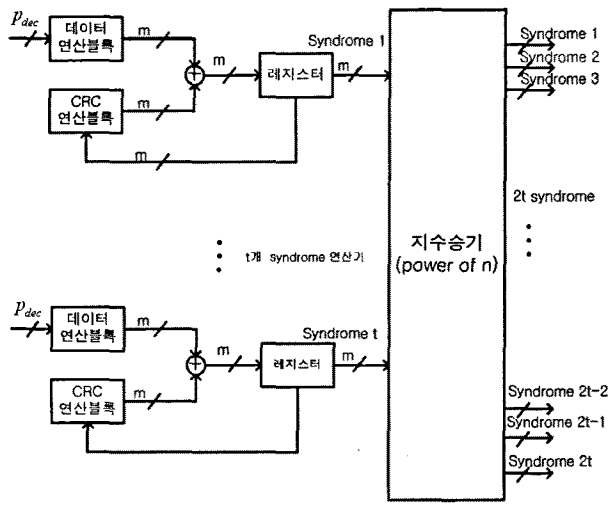

그림 3. 신드롬 생성기 블록도
Fig. 3. Syndrome generator block plot.

$$f_{synd}(x) = \sum_{\nu=p-m-1}^{0} U(v) + \sum_{\nu=p-1}^{p-m} U(v) \tag{10}$$

여기서 $f_{synd}(x)$를 원시 다항식 (primitive polynomial)로 나눈 결과가 병렬 신드롬 생성기의 데이터 연산이 되고 식 (11)으로 표현 할 수 있다.

$$p_{synd}(x) = \sum_{i=0}^{m-1} p_{synd}(i)x^i \tag{11}$$

그림 3에서 신드롬 생성기는 t 개 syndrome 연산기와 지수승기[3]로 이루어져 있고 신드롬 연산기는 데이터 연산 블록과 CRC 연산 블록과 레지스터로 이루어진다.

2. 오류 위치 다항식 생성기 (BM Algorithm)

오류 위치 다항식 생성기는 BM (Berlekamp Massey) 알고리즘이나 Euclidean 알고리즘이 쓰이는데 둘 다 반복적인 연산을 사용하여 위치 다항식을 푸는 방법이다. Euclidean 알고리즘은 BM 알고리즘에 비해 이해하기 쉽고 일반적인 구조를 가지나 반면에 효율에서는 BM 알고리즘이 더 효율적임을 알 수 있다.

이러한 BM 알고리즘은 inversion 방법과 inversion-less 방식이 있다. 병렬 BCH 경우는 inversion-less 방법을 쓰인다. inversion-less 의 특징은 나눗셈 연산이 없고 대신에 곱셈과 덧셈 연산만으로 이루어진다. 본 논문에서는 sawata가 제시한 RiBM[5] 방식으로 설계하였고 다른 연산기 보다 적은 하드웨어로 자원으로 timing multiplexing을 이용하여 많은 동작을 수행함으로서 silicon area를 줄여준다.

3.병렬 치엔 탐색 알고리즘 (Chien Search Algorithm)

오류 위치 다항식 생성기에서 구한 오류 위치의 해 값을 치엔 탐색 알고리즘 블록에 전달한다.

그림 4는 치엔 탐색 블록을 상세하게 나타내는 그림이고 색으로 표시한 부분은 축약 코드 (shorten code)[6]를 보상해 주는 Chien 전 처리부이다.

통상적으로 BCH 디코더에서 conventional 병렬 Chien 탐색 블록은 큰 계산의 복잡 도를 보이며 전체 디코더 회로 면적의 많은 면적을 차지 할 수 있다. 이를 줄이기 위해 다양한 아키텍처들이 제안되는데, 그중에서 곱셈기 (FFM: finite field multiplier) 및 곱셈기의 복잡 도를 줄이는 접근 방향이 주로 제안되었다.

종래의 직렬 Chien 탐지기 블록은 코드의 길이 n이고 오류 위치다항식 $\Lambda(x)$의 근이 α^i $(0 \le i \le n-1)$일 때 한 클록 사이클에 한 개의 근 α에 대해 오류 벡터 e를 계산하기 때문에 예를 들어 8196 비트의 코드라면 오류 위치를 찾는데 8196 사이클 (클록) 이 필요하다. 그런데, p비트씩 병렬 처리 경우에는 처리시간 n/p (또는 k/p) 사이클로 감축 할 수 있다, 이때, BCH 코딩의 곱셈이나 덧셈이 이루는 다항식으로 변형될 수 있다. 이를 연산 강도 강소 (strength reduced) 알고리즘[7]이라고 하고 설계에서는 inversion-less BM 알고리즘 출력에 맞게 변형하여 오류 위치다항식을 식 (12)로 나타낼 수 있다.

$$\begin{aligned}\Lambda(\alpha^i) &= \sum_{k=0}^{t+m} \alpha^k \lambda_k \\ &= \lambda_0 + \lambda_1\alpha^1 + \lambda_2\alpha^2 + \cdots + \lambda_{m-1}\alpha^{m-1} + \\ &\quad R_{f(x)}[\lambda_m\alpha^m + \lambda_{m+1}\alpha^{m+1} + \cdots + \lambda_{m+1}\alpha^{m+t}] \\ &= \sum_{k=0}^{m-1} \alpha^k\lambda_k + R_{f(x)} \sum_{k=m}^{m+t} \alpha^k\lambda_k \\ &= R_{f(x)} \sum_{k=m}^{m+t} \alpha^k\lambda_k \quad (\because \sum_{k=0}^{m-1} \alpha^k\lambda_k = R_{f(x)} \sum_{k=0}^{m-1} \alpha^k\lambda_k)\end{aligned} \tag{12}$$

식 (12)에서 $R_{f(x)}[a(x)]$는 $a(x)$을 원시다항식 $f(x)$로 나누었을 때의 나머지이다.

병렬 BCH 디코더에 인가되는 코드의 길이 n 은 실질적인 정보 코드와 정보에 포함된 redundancy 코드와 검사 코드이다. Chien 탐색전에 코드에서 redundancy 코드를 잘라내고 축약 코드를 만들어 주어야 한다.

BCH (n,k) 부호에서, 길이 l 만큼 정보코드를 잘라낸다면 (n-l, k-l) 축약코드가 만들어지는데 이 경우의 새로운 위치 다항식은 식 (13)과 같다.

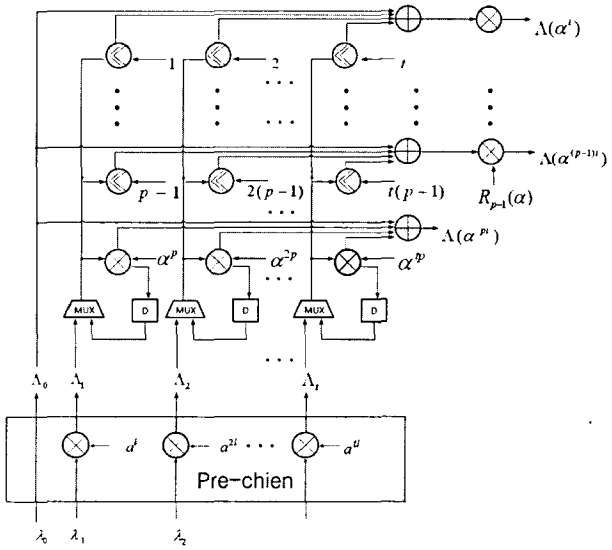

그림 4. 병렬 Chien 탐색 알고리즘 블록
Fig. 4. Parallel Chien search block.

$$\Lambda(x)=\sum_{i=0}^{t}(\lambda_i \cdot \alpha^{tl}) = \lambda_0+(\lambda_1 \cdot \alpha^{l})x+(\lambda_2 \cdot \alpha^{2l})x+\cdots+(\lambda_t \cdot \alpha^{tl})x^t \quad (13)$$

Ⅳ. 모의실험 결과

설계한 (n=4330,k=4096, t=18 및 n=4200, k=4096, t=8) BCH 인코더 및 디코더는 Verilog HDL로 구현되었고 0.18μm CMOS공정을 이용하여 Design compiler를 사용하여 합성을 하였다.

1. 모의실험

그림 5는 입력받은 데이터를 이용하여 에러 위치다항식(LAMBDA0~LAMBDA18)을 구하고 에러 위치다항식의 차수인 LAMBDA_Degree는 에러의 수(6 비트 에러)이다. 화살표로 표시된 시점에서 6비트 에러를

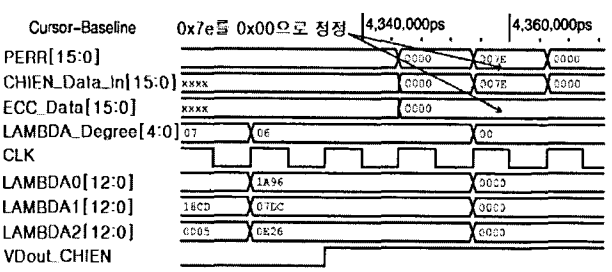

그림 5. 모의실험
Fig. 5. Simulation.

가진 CHIEN_Data_In (0x007E)에 PERR (에러 위치 데이터: 0x007E)을 더하여서 오류 정정된 ECC_Data (0x0000)을 출력한다.

2. Gate Counter 및 동작주파수

Design compiler에서 디자인을 합성 (Synthesis) 한 결과와 동작주파수와 Gate 수를 표 1에서 보여주고 있다.

표 1. 인코더와 디코더 동작 주파수 & Gate #
Table 1. Encoder and decoder operation frequency & Gate #.

		Gate #	동작주파수 (MHz)
인코더		5961	120
디코더	Syndrome 생성기	17440	110
	BM 블록	96308	
	Chien 탐색 블록	36487	

3. Throughput

BCH (n=4330,k=4096,t=18) 인코더의 수신된 프레임 512B (4096b) 를 513사이클에 CRC 코드를 출력되어 진다. 바이트 단위로 출력한다면 전송프레임은 532B가 되고 533 사이클 후에 프레임 인코딩이 완료된다.

디코더의 수신된 프레임을 532B를 신드롬 생성기에서 16bit 단위로 267사이클 동안 처리한고 이 신드롬을 BM (Berlekamp Massey) 블록에서 36 사이클 후에 에러위치다항식을 출력하면 Chien 탐색 블록에서 256 사이클 동안 에러정정 함으로 총 559 사이클에 동작을 완료한다. 아래의 표에서 괄호 안은 n=4200, k=4096, t=8 의 Throughput이다.

표 2. 프레임 처리 Throughput
Table 2. Frame process Throughput.

		Frame Cycle#		Throughput
인코더		533(526)		0.93 Gb/s
디코더	Syndrome 생성기	559 (535)	267(263)	0.85 Gb/s
	BM 블록		36(16)	
	Chien 블록		256	

3. 성능 비교

표 3은 제안된 설계와 2009년 IEEE VLSI 심포지엄에 발표된 논문[8]을 비교한 테이블이다. Clock rate 와 encoding latency에서는 VLSI 심포지엄 논문이 조금

표 3. ECC 성능비교
Table 3. Performance summary and Comparison.

Parameter	This Work	[8]
Area	1.55mm^2 (0.18μm tech)	0.808mm^2 (0.13μm tech)
User data size	512B	512B
Correction capability(t)	8, 18	9,14,19,24
Parity space	13B, 30B	16B,24B,32B,40B
Clock rate	120/110 Mhz (encoder/ decoder)	125Mhz
Encoding latency	4.4μs	4.1μs
Decoding latency	2.67μs(min) 5.59μs(max)	4.22μs(min) 12.7μs(typ) 13.3(max)
Gate count	156.2k	158.9k
Throughput	116MB/s (enc) 106MB/s (dec)	119MB/s(enc) 60MB/s(dec)

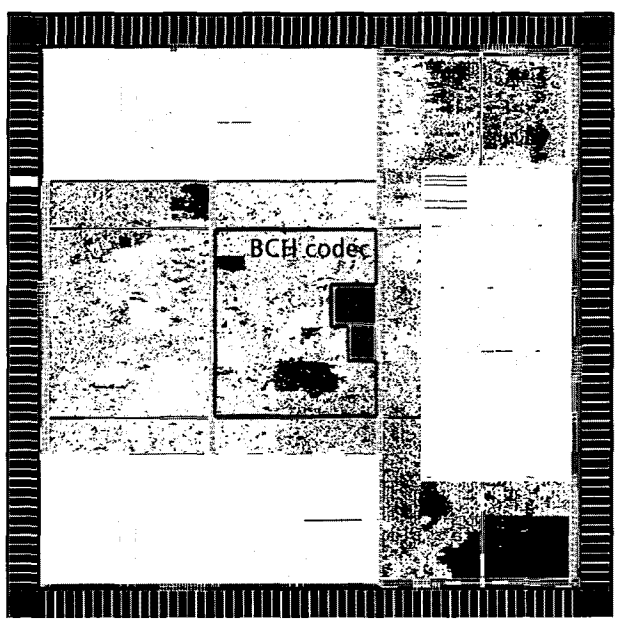

그림 6. 제안된 BCH ,CODEC ASIC의 레이아웃
Fig. 6. The layout of the proposed BCH CODEC.

더 낮은 결과를 보여 주는데 이에 대한 근거로는 제안된 설계의 경우는 0.18μm 공정이고 VLSI 심포지엄 논문의 경우는 0.13μm 공정이기 때문이다. 디코딩의 경우에는 지연시간을 줄이는 것이 성능개선사항이며 제안된 설계는 VLSI 심포지엄 논문에 비해 절반의 decoding latency를 가짐으로서 에러율이 높은 경우에서도 높은 throughput를 얻을 수 있다.

V. 결 론

본 논문에서는 NAND flash를 위한 메모리 영역의 활용에 따라 에러 정정을 선택적으로 할 수 있는 8/18 비트의 선택적인 BCH 인코더 및 디코더를 구조를 제안하고 설계하였다. 인코더에 비해 많은 지연시간을 갖는 디코더를 인코더의 입력에 두 배의 width로 처리함으로써 읽기 속도를 개선하였다.

향후 수행할 연구로 ECC 블록과 다수의 flash 메모리를 처리할 수 있는 flash control 블록과 데이터 교환을 효율적 처리를 연구를 진행할 것이다.

참 고 문 헌

[1] 윤상호, 이한호, "광통신용 40Gb/s Concatenated BCH 복호기 구조," 대한전자공학회 2009년 SoC학술대회, 158-161쪽, 전북대학교, 한국, 2009년 5월
[2] 류태규, 정용진, "DMB 휴대용 단말기를 위한 Reed-Solomon 복호기의 설계," 전자공학회 논문지, 제43권 SD편, 38-48쪽, 2006년 4월
[3] 이만영, "부호와 Reed-Solomon 부호," 민음사, 46-61쪽, 1990.
[4] Politecnico di Torino, "Parallel CRC generation," IEEE Micro, Vol. 10, pp. 63-71, Oct. 1990.
[5] Dilip V. Sarwate, "High-Speed Architectures for Reed-Solomon Decoders," IEEE Trans. VLSI Systems, Vol. 9, no. 5, pp. 641-655, Oct. 2001.
[6] Todd K. Moon, "Error Correction Coding," Wiely, pp. 143-144, Mar. 2005.
[7] Junho Cho, "Strength-Reduced Parallel Chien Search Architecture for Strong BCH Codes," IEEE Trans. Circuits and Systems II: express brief Vol. 55, no. 5, pp. 427-431, May 2008.
[8] Te-Hsuan Chen, Yu-Ying Hsiao, Yu-Tsao Hsing, Cheng-Wen Wu, "An Adaptive-Rate Error Correction Scheme for NAND Flash Memory," IEEE VLSI Test Symposium, pp. 53 - 58, May 2009.

저 자 소 개

이 상 혁(학생회원)
1999년 충북대학교 반도체공학과 학사 졸업.
2006년~2008년 매그나칩 ISD 사업부 선임 연구원
2009년~현재 중앙대학교 전자공학과 석사과정.
<주관심분야 : 디지털 회로설계, Mixed 회로 설계>

백 광 현(평생회원)
1990년 고려대학교 전자전산 공학과 공학사
1998년 고려대학교 전자공학과 공학석사
2002년 University of Illinois at Urbana-Champaign 공학박사
1990년~1996년 삼성전자 LSI 사업부 선임연구원
1998년~2001년 University of Illinois CSL Research Assistant
2001년~2006년 Rockwell Scientific Senior Scientist
2006년~현재 중앙대학교 전자전기공학부 부교수
<주관심분야 : 통신, 컴퓨터, 신호처리, 반도체>