Regular paper

Journal of the KIECS. pp. 333-340, vol. 13, no. 2, Apr. 30 2018, t. 88, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2018.13.2.333

병렬 CRC 생성 방식을 활용한 BCH 코드 복호기 설계

갈홍주* · 문현찬** · 이원영***

Design of BCH Code Decoder using Parallel CRC Generation

Hong-Ju Kal* · Hyun-Chan Moon** · Won-Young Lee***

요 약

본 논문은 병렬 CRC 생성 방식을 적용한 BCH 코드 복호기를 소개한다. 기존에 사용되는 병렬 신드롬 생성기로 LFSR(: Linear Feedback Shift Register)을 변형한 방식을 사용하면 짧은 길이의 코드에 적용하는 데 많은 면적을 차지한다. 제안하는 복호기는 짧은 길이 코드워드의 복호화를 위해 병렬 CRC(: Cyclic Redundancy Check)에서 체크섬을 계산하는 데 사용되는 방식을 활용하였다. 이 방식은 병렬 LFSR과 비교해 중복된 xor연산을 제거해 최적화된 조합회로로 크기가 작고 짧은 전파지연을 갖는다. 시뮬레이션 결과 기존 방식 대비 최대 2.01ns의 지연시간 단축 효과를 볼 수 있다. 제안하는 복호기는 0.35-μm CMOS 공정을 이용하여 설계하고 합성되었다.

ABSTRACT

This paper introduces a BCH code decoder using parallel CRC(: Cyclic Redundancy Check) generation. Using a conventional parallel syndrome generator with a LFSR(: Linear Feedback Shift Register), it takes up a lot of space for a short code. The proposed decoder uses the parallel CRC method that is widely used to compute the checksum. This scheme optimizes the a syndrome generator in the decoder by eliminating redundant xor operation compared with the parallel LFSR and thus minimizes chip area and propagation delay. In simulation results, the proposed decoder has accomplished propagation delay reduction of 2.01 ns as compared to the conventional scheme. The proposed decoder has been designed and synthesized in 0.35-μm CMOS process.

키워드

BCH Code, Parallel CRC Generation, Parallel BCH Decoder, Syndrome Generator
BCH 부호, 병렬 CRC 생성 방식, 병렬 BCH 복호기, 신드롬 생성기

Ⅰ. 서 론

데이터 통신에는 다양한 방식의 오류 비트 정정 방식이 존재한다[1-4]. 다양한 방식의 오류 비트 정정 코드 중 BCH(: Bose-Chaudhuri-Hocquenghem) 코드는 두 개 이상의 다중 오류를 정정할 수 있는 부호로 뛰어난 정정 능력을 갖는다. 저장매체나 통신회로 내 정보량이 증가함에 따라 신뢰성 확보를 위해 hamming 코드와 같이 단일 오류 정정에 중점을 둔 방식에 비해 BCH 코드가 ECC(: Error Correction Circuit)에 활용되고 있다. 예컨대 BCH 코드는 플래시 메모리, 이동통신, 암호화(stegnography), SRAM

*,** 서울과학기술대학교 전자IT미디어공학과
(qawsed9342@gmail.com*, mhcqwe92@naver.com**)
*** 교신저자 : 서울과학기술대학교 전자IT미디어공학과
· 접 수 일 : 2018. 02. 23
· 수정완료일 : 2018. 03. 20
· 게재확정일 : 2018. 04. 15
· Received : Feb. 23, 2018, Revised : Mar. 20, 2018, Accepted : Apr. 15, 2018
· Corresponding Author : Won-Young Lee
Dept. of Electronic and IT Media Engineering, Seoul National University of Science and Technology,
Email : wylee@seoultech.ac.kr

등에 주로 사용된다[5-6]. 그 중 단일 통신회로를 이용하는 회로는 직렬 방식의 복호기를 사용하는 데 주로 LFSR(: Linear Feedback Shift Register)방식이 쓰인다. 그러나 모든 정보데이터가 직렬로 입력되어 복호화에 많은 시간이 소요되는 단점이 있다. 만약 정보데이터가 병렬로 입력되어 병렬 회로를 통해 처리하는 방식을 이용할 수 있다면 복호화 시간을 줄일 수 있다. 따라서 복호기 구조에서 정보데이터를 입력받는 부분인 신드롬 생성기를 병렬 회로로 할 수 있어야만 한다. 긴 코드워드의 BCH코드를 병렬회로로 만드는 기존 방법 중에는 LFSR기반 신드롬 생성기가 있다. 그러나 이 방식을 모든 입력이 한 번에 처리되는 짧은 길이의 코드워드에 적용할 경우 불필요한 회로가 생성되는 단점이 있다.

이러한 문제를 해결하기 위해서 본 논문에서는 병렬 CRC 생성 방식을 사용한 병렬 복호기를 제안하고 있다. 본문의 2장에서 BCH 코드의 기본적인 설명과 병렬 CRC 생성 방식을 적용한 예시를 통해 짧은 코드워드의 병렬 신드롬 생성기를 만들어보고 긴 코드워드에 적용해 LFSR 기반의 신드롬 생성기를 변형시키는 방식을 제안한다. 3장에서는 실험을 통해 얻은 내용에 대해 논의하고, 4장에서 결론을 맺는다.

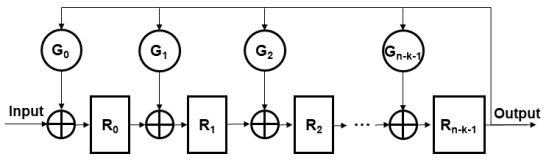

그림 1. BCH 부호화 과정에 사용되는 LFSR
Fig. 1 Block diagram of LFSR for BCH encoding

III. 본론

2.1 BCH 코드의 부호화와 복호화

BCH 코드에서는 비트 계산을 용이하게 하기 위해 데이터를 다항식으로 나타낸다. 코드워드는 데이터 비트와 그 뒤에 패리티 비트가 합쳐진 것을 의미한다. k비트의 정보비트를 가지고 t비트의 오류를 정정할 수 있는 길이 n비트의 코드워드(n, k, t) 다항식 R(x)는 식 (1)과 같이 표현할 수 있다.

$$
\begin{aligned}
&R(x) = x^{n-k}d(x) + \gamma(x)\\
&d(x) = d_{k-1}x^{k-1} + d_{k-2}x^{k-2} + \ldots + d_1x + d_0\\
&\gamma(x) = \gamma_{n-k-1}x^{n-k-1} + \ldots + \gamma_1x + \gamma_0\\
&R = (d_k, d_{k-1}, \ldots, d_1, d_0, \gamma_{n-k}, \gamma_{n-k-1}, \ldots, \gamma_1, \gamma_0)\\
&d = (d_k, d_{k-1}, \ldots, d_1, d_0)\\
&\gamma = (\gamma_{n-k}, \gamma_{n-k-1}, \ldots, \gamma_1, \gamma_0)
\end{aligned}
\tag{1}
$$

d(x)는 데이터 다항식으로 부호화하려는 정보비트이다. g(x)는 패리티 다항식으로 $d(x)x^{n-k}$를 생성 다항식 G(x)로 나누는 부호화 과정에서 생성된다. 이때 생성 다항식 G(x)은 최소다항식의 최소공배수이다. 최소다항식은 최고항이 x^m (m = $\log_2$n)이며 유한체(galois field) GF(2^m)에 속하는 원시원소 중 일부를 해로 갖는 다항식을 의미한다. 수정 가능한 오류의 개수가 t라고 할 때, 최소다항식 $m_i(x)$는 α^j(j = i, 2i, ..., 2^t·i)를 근이 된다. $m_i(x)$를 전개하여 계수를 계산할 때 합은 모듈로 2 연산이므로 계수는 GF(2)의 원소(0 또는 1)가 된다. 정정할 수 있는 오류의 개수가 t개 일 때 존재하는 최소다항식 개수는 2t개이다. $m_1(x) = m_2(x) = m_4(x) = \ldots = m_{2t}(x)$ 이므로 식 (2)와 같이 표현할 수 있다.

$$
\begin{aligned}
G(x) &= LCM\{m_1(x), m_2(x), \cdots, m_{2t}(x)\}\\
&= LCM\{m_1(x), m_3(x), \cdots, m_{2t-1}(x)\}
\end{aligned}
\tag{2}
$$

다시 말해서, 부호화 과정은 데이터 다항식을 생성 다항식으로 모듈로 나누기 연산한 나머지 값인 패리티 다항식을 뒤에 붙여 코드워드 다항식을 만드는 것을 의미한다. 일반적인 부호화 과정에는 그림 1과 같

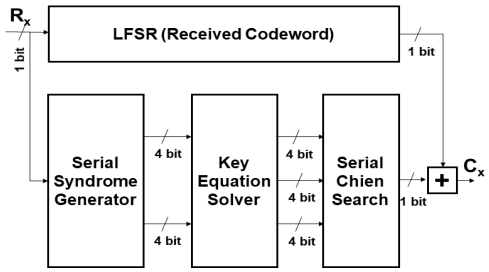

그림 2. BCH 복호기의 전체 구조
Fig. 2 Block diagram of the BCH decoder

은 레지스터 n-k단을 가지고 있는 LFSR을 사용한다. LFSR은 d(x)의 최고차항부터 각 항의 계수가 1인지 확인한다. 어떠한 항의 계수가 1이면 G(x)로 나눠질 수 있다는 것을 의미하고 다음 시행에 G(x)만큼을 xor 연산을 통해 빼준다. 이처럼 모든 자리 수에 대해서 빼기 계산을 완료하면 나머지 값이 남는다. 회로를 통해 보면 데이터 비트가 직렬로 들어오며 k개의 정보데이터가 쉬프트 된다. n-k 단에서 나오는 값이 1이면 G(x)의 계수로 xor연산을 해준다. 데이터비트 n-k 비트와 k개의 0비트가 입력되고 레지스터에 남은 연산 결과가 패리티비트 값이다.

복호화과정은 오류의 유무 및 오류 패턴을 분석한다. 각각의 최소다항식으로 모듈러 연산을 수행하여 나머지 값인 신드롬 구한다. 신드롬을 통해 오류 위치를 구할 수 있다. 일반적인 BCH 코드의 복호화 과정은 4가지 단계로 나눈다.

1. 코드워드 R(x)로부터 신드롬을 계산한다.
2. 신드롬(s-bit) 으로 오류위치 다항식 σ(x)를 결정한다.
3. σ(x)의 근인 오류위치를 구한다.
4. 오류위치의 오류를 정정한다.

일반적인 복호기는 신드롬 생성기, key equation solver, chein search로 구성되어 있다. 수신된 코드워드는 직렬로 복호기에 입력되고 부호기와 같은 원리로 나누기 연산을 시행해 신드롬 값을 얻는다. 신드롬 생성기로부터 4비트의 신드롬 두개를 받아 Key equation solver에서 식 (3)을 계산해 오류위치 다항식을 구하게 된다.

$$\sigma(x) = 1 + s_1 + (s_1^2 + \frac{s_3}{s_1})x^2 \quad (3)$$

이 식을 계산하기 위해 곱셈기, 덧셈기 및 역원을 구하는 연산기가 사용된다. 오류위치 다항식 σ(x)로부터 오류위치를 구할 수 있다. Chien search에서 σ(x)의 근이 될 수 있는 모든 값 1, α, ... , α^{n-1}을 σ(x)에 직접 대입해 오류의 위치를 찾을 수 있다. σ(x)의 근을 찾는 알고리즘은 Chien이 제안한 방식이 널리 사용되고 있다. 16비트 직렬 복호기는 그림 2와 같이 표현된다.

2.2 병렬 CRC 생성 방식을 활용한 제안하는 신드롬 생성기

2.2.1 병렬 CRC 생성 방식

제안하는 BCH 복호기에서는 BCH 코드를 복호화하는데 필요한 신드롬을 계산하기 위해 병렬 CRC 생성 방식을 적용하였다. CRC(: Cyclic Redundancy Check)는 간단한 오류 확인 방법으로써 전송하는 데이터에 대한 CRC 계산 값인 체크섬을 데이터에 붙여 송신한다. 수신기에서는 수신된 데이터의 데이터 비트에 대한 체크섬 값을 다시 계산하여 기존의 CRC값과 차이가 있는지 확인해 오류가 생겼는지 확인 한다. 이러한 CRC를 계산하는 방식은 LFSR을 주로 이용하지만, 데이터를 순서대로 입력되는 방식이 아닌 데이터 전체를 병렬 입력을 받아 체크섬을 계산할 경우, 최적화된 조합 논리회로를 통해 기존 대비 빠른 시간 내에 CRC 계산이 가능하다[7-8]. 이와 같은 병렬 CRC 생성 방식은 LFSR의 작동 원리를 토대로 구한다. 이때 결과 값인 레지스터 값은 입력 비트와 현재 레지스터의 값에 따라 결정된다. 결과 값은 입력 비트와 현재 레지스터 값에 대해 유일한 값을 갖고 선형적으로 합이 가능하다.

2.2.2 짧은 길이 코드워드의 병렬 신드롬 생성기

데이터의 암호화 또는 SRAM과 같은 어플리케이션에서 BCH 코드를 사용하여 데이터를 처리하는 경우, 32비트 이하의 BCH 코드가 많이 사용되며 짧은 길이의 코드워드를 받아 병렬적으로 한 번에 계산하는 방식이 존재한다. BCH 코드 (n, k, t) = (16, 8, 2)를 디코딩하는 과정을 예시로 병렬 신드롬 계산기를 생성 방식을 확인할 수 있다. 그림 3은 기존 LFSR를 사용한 신드롬 생성기를 보여주고 있다. LFSR에 활

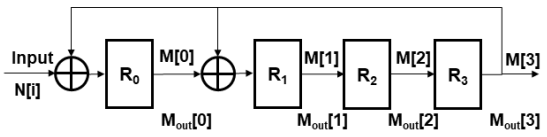

그림 3. (16, 8, 2) 코드워드 복호화에 사용되는 기존 방식의 LFSR 구조
Fig. 3 Block diagram of the conventional LFSR for decoding (16, 8, 2) codeword

용되는 최소 다항식은 x^4+x^1+1이다. 제안하는 BCH 복호기 구조는 LFSR 회로를 사용한 CRC회로를 병렬 CRC회로로 변경하는 방식을 적용함으로써 기존 LFSR를 이용한 신드롬 생성기와 동일한 출력 값을 유지하면서 병렬데이터 처리가 가능케 하였다. 예로, 그림 3과 같이 16자리 코드워드를 5자리 최소 다항식으로 나눌 경우 총 1비트씩 12번의 입력이 들어가게 된다. 이 BCH 코드의 신드롬 생성기를 3개의 입력 비트에 대한 병렬 CRC 구조로 바꾸는 방법은 아래와 같은 순서이다.

1단계, 코드워드에 대한 LFSR 루틴을 확인한다. 코드워드의 길이와 레지스터의 단수를 확인한다. LFSR에 들어가는 코드워드의 길이는 12 비트이다. LFSR의 레지스터는 4단으로 이루어져있다.

2단계, 입력에 따른 레지스터 출력 값을 표로 정리한다. 입력에 대한 출력 신호는 비트 간 선형적인 관계를 가지고 있기 때문에 입력을 각 자리 비트로 구분해서 계산할 수 있다. 예를 들어 정보다항식 R(x)의 각 비트 자리를 N[i] (i = 0, 1, 2, ...) 라고 하고, 3비트씩 나눠서 LFSR로 입력된다고 할 때, 입력되는 시간 순서대로 N[2], N[1], N[0]이라고 표시할 수 있다(N[2] : MSB). 이때 각 자리 레지스터에 남는 결과 값은 출력 단에 가까운 쪽부터 $M_{out}[3]$, $M_{out}[2]$, $M_{out}[1]$, $M_{out}[0]$으로 표시하며, 초기값 M[3]는 1이고 M[2], M[1], M[0]은 전부 0으로 설정한다. 이러한 환경에서 N[2], N[1], N[0] 중 하나만 1인 세가지의 경우에 대해서 LFSR 동작을 수행하면 표 1과 같은 결과를 얻을 수 있다. 예를 들어, 표 1의 세 번째 행인 N[2]는 N[2] = 1, N[1] = 0, N[0] = 0이 순차적으로 입력되는 경우의 수행된 결과를 보여주고 있는 것이다.

표 1. 3 비트 입력 값에 따른 LFSR 출력
Table 1. Outputs of the LFSR according to 3-bit input code

	$M_{out}[3]$	$M_{out}[2]$	$M_{out}[1]$	$M_{out}[0]$
N[0]	0	0	1	1
N[1]	0	1	1	0
N[2]	1	1	0	0

표 2. 레지스터 초기 출력 값에 따른 LFSR 출력
Table 2. Outputs of the LFSR according to initial outputs of registers

	$M_{out}[3]$	$M_{out}[2]$	$M_{out}[1]$	$M_{out}[0]$
M[0]	1	0	0	0
M[1]	0	0	1	1
M[2]	0	1	1	0
M[3]	1	1	0	0

3단계, 초기 레지스터 상태에 따른 레지스터 출력 값의 변화를 계산하고 이에 대한 매트릭스를 표 2와 같이 작성한다. 입력뿐만 아니라 레지스터의 초기 값 또한 출력에 영향을 주고 선형적인 합으로 표현 가능하다. 레지스터 출력 값을 계산하기 위해서는 입력 비트는 모두 0으로 설정하고 입력 N의 비트 길이만큼 LFSR을 동작 시킨 후 결과를 확인하면 된다. 예를 들어, 그림 3과 같이 4개의 레지스터로 LFSR이 구성된 환경에서 3 비트 길이의 입력인 N[2], N[1], N[0]은 모두 0이며, 레지스터 초기 값은 M[0] = 1, M[1] = M[2] = M[3] = 0이라고 하자. 이 때, 입력 비트의 길이인 3 클럭 사이클만큼 동작 시키면 레지스터의 값은 $M_{out}[3]$ = 1, $M_{out}[2]$ = 0, $M_{out}[1]$ = 0, $M_{out}[0]$ = 0 이 된다.

4단계, 매트릭스를 토대로 각 레지스터별 출력에 영향을 주는 것을 확인하여 병렬 CRC 방정식을 작성한다. 3 비트 입력에 대한 병렬 CRC 방정식은 식 (4)와 같이 작성할 수 있다.

$$\begin{aligned} M_{out}[0] &= N[0] + M[1] \\ M_{out}[1] &= N[0] + N[1] + M[1] + M[2] \\ M_{out}[2] &= N[1] + N[2] + M[2] + M[3] \\ M_{out}[3] &= N[2] + M[0] + M[3] \end{aligned} \tag{4}$$

$M_{out}[0]$의 경우 초기값 M[1]과 N[0]에 영향을 받는 것을 알 수 있다. 선형적인 합으로 표현되기 때문에 XOR 연산을 사용하여 구현할 수 있다. 만약, M[3]를 제외한 모든 레지스터 초기 값이 0이라면 입력만으로 결과 수식을 간단히 표현할 수 있다. 16비트 예시에

표 3. 12 비트 입력 값에 따른 LFSR 출력
Table 3. Outputs of the LFSR according to 12-bit input code

	$M_{out}[3]$	$M_{out}[2]$	$M_{out}[1]$	$M_{out}[0]$
N[0]	0	0	1	1
N[1]	0	1	1	0
N[2]	1	1	0	0
N[3]	1	0	1	1
N[4]	0	1	0	1
N[5]	1	0	1	0
N[6]	0	1	1	1
N[7]	1	1	1	0
N[8]	1	1	1	1
N[9]	1	1	0	1
N[10]	1	0	0	1
N[11]	0	0	0	1

서, 정보 다항식 R(x)은 최고차항이 15이고 생성다항식 G(x)은 최고차항이 4이다. 신드롬 다항식은 정보 다항식을 생성다항식으로 나눈 나머지 값을 의미한다. 따라서 신드롬 다항식은 최고차항이 3이고 4비트로 표현할 수 있다. 정보다항식의 상위 12비트에 대한 LFSR로 나누기를 실시해 나머지를 구할 수 있다. 12개의 병렬입력에 관해서 표 3과 같이 매트릭스를 작성하면 그림 4와 같이 입력에 관한 식을 구해 신드롬 생성기를 만들 수 있다. 짧은 코드워드의 경우, CRC 생성 방식으로 병렬 신드롬 생성기를 구현하는 것이 LFSR을 병렬로 바꾼 방식과 비교했을 때, 동일한 정보 비트와 오류 정정 가능 비트 수를 가지면서도 불필요한 논리 연산식 및 레지스터가 사라져 구현하는데 필요한 면적이 줄어드는 장점이 있음을 알 수 있다.

2.2.3 긴 길이 코드워드의 병렬 신드롬 계산기

긴 코드워드의 BCH 코드는 주로 낸드 플래시 메모리에 주로 쓰인다[9]. 낸드플래시 중 MLC 타입은 저장용량이 크지만 잡음이나 인접 셀의 간섭에 취약하다. 낸드 플래시 메모리는 한 페이지 단위로 엑세스한다. 한 페이지는 4224바이트이고 그중 4096바이트는 정보데이터를 저장하는 데이터 영역이고 128바이트는 부호화된 페리티를 저장하는 여분 영역이다. 정보데이터는 512바이트의 섹터로 나눠지며 이 섹터에 대해

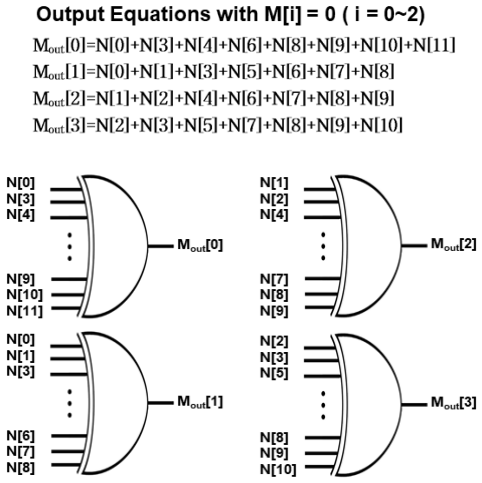

그림 4. 제안하는 병렬 CRC 방식을 사용한 (12, 8, 2) 신드롬 생성기
Fig. 4 Schematic of the proposed (12, 8, 2) syndrome generator using parallel CRC generation scheme

오류 검사를 실시한다. 즉, ECC로 한 섹터 내 2개의 오류를 수정할 수 있는 BCH (4122,4096,2)가 가장 간단한 오류정정 모델이다. 이때 복호기에 주로 바이트 단위 병렬 BCH 신드롬 생성기를 사용한다. 신드롬을 계산하는데 필요한 최소다항식은 식 (5)와 같다.

$$m_1(x) = 1 + x + x^3 + x^4 + x^{13}$$
$$m_3(x) = 1 + x + x^3 + x^4 + x^5 + x^7 + x^9 + x^{13} \quad (5)$$

최소다항식 $m_1(x)$에 대한 기존의 1바이트 입력 병렬 LFSR 신드롬 생성기는 그림 5와 같다. 기존 방식의 회로는 4122비트에 대한 신드롬 13개를 계산하는데 516 사이클 (≒ 4122/8)이 필요하다.

이 같이 긴 연산 시간을 단축하기 위해 병렬 CRC 생성 방식을 사용하여 신드롬 생성기를 만들 수 있다. 최대 8개의 병렬 입력을 받아 13개의 레지스터를 통해 처리하는 구조에 대해 병렬 CRC 생성 방식을 이용해서 매트릭스를 작성한다. LFSR 구조 생성기의 입력에 따른 레지스터 출력 값을 정리하고, 초기 레지스터 상태에 따른 레지스터 출력 값의 변화를 계산한다. 입력 값과 레지스터 초기 값에 대한 LFSR 연산 결과 값을 바탕으로 선형적인 특징을 이용하여 병렬

CRC 방정식을 작성한다. 출력이 M[0]부터 M[12]이므로 식 (6)과 같이 총 13개의 방정식이 생성된다.

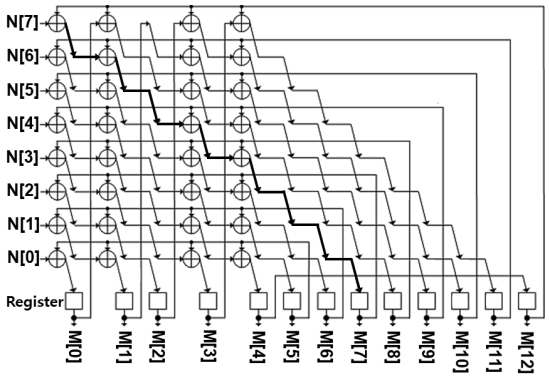

그림 5. (4224, 4096, 2) 코드워드 복호화를 위한 기존 방식의 병렬 LFSR 신드롬 생성기
Fig. 5 Block diagram of the conventional parallel LFSR syndrome generator for decoding (4224, 4096, 2) codeword

$$
\begin{aligned}
M_{out}[0] &= M[5]+N[0]] \\
M_{out}[1] &= M[5]+M[6]+N[0]+N[1] \\
M_{out}[2] &= M[6]+M[7]+N[1]+N[2] \\
M_{out}[3] &= M[5]+M[7]+M[8]+N[0]+N[2] \\
&\quad +N[3] \\
M_{out}[4] &= M[5]+M[6]+M[8]+M[9]+N[0] \\
&\quad +N[1]+N[3]+N[4] \\
M_{out}[5] &= M[6]+M[7]+M[9]+M[10]+N[1] \\
&\quad +N[2]+N[4]+N[5] \\
M_{out}[6] &= M[7]+M[8]+M[10]+M[11]+N[2] \\
&\quad +N[3]+N[5]+N[6] \\
M_{out}[7] &= M[8]+M[9]+M[11]+M[12]+N[3] \\
&\quad +N[4]+N[6]+N[7] \\
M_{out}[8] &= M[0]+M[9]+M[10]+M[12]+N[4] \\
&\quad +N[5]+N[7] \\
M_{out}[9] &= M[1]+M[10]+M[11]+N[5]+N[6] \\
M_{out}[10] &= M[2]+M[11]+M[12]+N[6]+N[7] \\
M_{out}[11] &= M[3]+M[12]+N[7] \\
M_{out}[12] &= M[4]
\end{aligned}
\tag{6}
$$

그림 6은 병렬 CRC 생성 방식으로 만들어진 방정식 중 7번 출력 ($M_{out}[7]$)의 회로 구현 예를 보여주고 있다. 연산에 필요한 크리티컬 패스를 줄이고 비슷한 길이의 tree 모양 회로로 구성하는 하는 방식으로서 방정식에서의 덧셈 연산은 XOR 게이트로 구현되었다[10]. 병렬 CRC 방식을 사용한 방정식을 구하는 과정에서 피드백 연산이 고려되었기 때문에 신호의 흐름이 한 방향으로만 되어 있다. 따라서 바이트 크기 병렬 LFSR을 전파지연이 적은 최적화된 회로로 바꿀 수 있다.

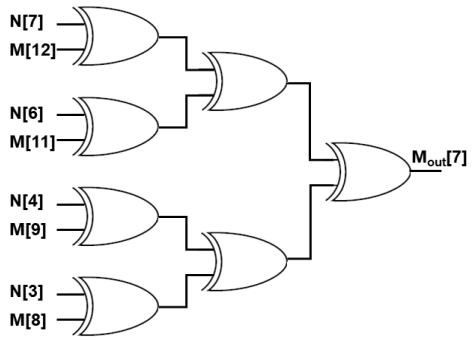

그림 6. (4224, 4096, 2) 코드워드 복호화를 위한 병렬 CRC 방식의 신드롬 생성기의 $M_{out}[7]$ 구현
Fig. 6 $M_{out}[7]$ implementation of the proposed (4224, 4096, 2) syndrome generator using parallel CRC generation scheme

표 4. 합성된 회로를 사용한 신드롬 생성 시간 시뮬레이션 결과
Table 4. Simulation results of syndrome generation time using synthesized circuits

	Propagation time	
	LFSR	CRC Tree
S1 계산	2.24 ns	1.25 ns
S3 계산	3.61 ns	1.60 ns

III. 시뮬레이션 결과

병렬 LFSR 방식과 CRC 생성 방식 구현한 신드롬 생성기의 전파 지연 특성을 비교하기 위해서 회로를 verilog로 디자인하고 0.35μm CMOS 공정 라이브러리를 이용하여 합성했다. 표 4는 합성한 결과를 사용한 전파지연에 대한 시뮬레이션 결과이다. 신드롬 값 중 지연시간이 긴 S1과 S3 값을 비교했을 때, S1의 경우 LFSR 방식 대비 0.99ns의 지연시간이 단축되었으며, S3의 경우는 2.01ns의 지연시간 단축 효과를 볼 수 있다.

CRC 생성 방식을 적용하면 지연시간 단축 효과 외에도 병렬 입력을 늘릴 수 있는 장점이 있다. 수 있다. 기존의 병렬 LFSR을 구성하는 방식에서는 13개의 레지스터에 대해 최대 13개의 병렬 입력만이 가능하다. 하지만 CRC 생성 방식을 이용하면 16비트 32비트에 대해서도 신드롬 생성기를 쉽게 구성할 수 있다. 이는 [9]에서 제안한 leap ahead 방식을 이용한 16비트 32비트 입력의 신드롬 생성기 설계와 동일한 논리식을 갖는다.

Ⅳ. 결 론

본 논문에서는 CRC에서 체크섬을 구하기 위한 병렬 회로 설계 방식을 차용해 쉽게 BCH 복호기 중 병렬 신드롬 생성기를 구성하는 방식을 제안한다. 짧은 코드워드를 갖는 BCH 코드에서 병렬 LFSR 방식을 이용한다면 상쇄되는 불필요한 연산이 포함되기 때문에 크기가 크고 속도가 느린 반면, 병렬 CRC 생성 방식을 활용하면 병렬 입력에 대한 가장 최적화된 회로를 구성할 수 있다. 긴 코드워드를 갖는 BCH코드에서 병렬 LFSR 방식은 신드롬 레지스터 수만큼의 병렬 입력만을 사용할 수 있다. 이를 해결하기 위해 CRC 생성 방식을 이용하여 병렬 입력의 수를 늘렸다. 이처럼 CRC 생성 방식을 이용하면 자유롭게 병렬 계수를 설정하고 쉽게 최적화된 논리를 구성할 수 있다.

감사의 글

이 연구는 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. NRF-2016R1C1B1013321). 본 연구는 IDEC에서 EDA Tool를 지원받아 수행하였습니다.

References

[1] E. Jang, “LDPC Coding for image data and FPGA Implementation of LDPC Decoder," *J. of the Korea institute of Electronic Communication Science*, vol. 12, no. 4, 2017, pp. 569-574.

[2] J. Jung, Y. Lee, and H. Shin, “A Study on Forward Error Correction of Long-Distance Submarine Optical Communication Systems," *J. of the Korea institute of Electronic Communication Science*, vol. 3, no. 3, 2008, pp. 170-176.

[3] A. Doniyor and H. Suh, “An Efficient Algorithm for finding Optimal Spans to determine R=1/2 Rate Systematic Convolutional Self-Doubly Orthogonal Codes," *J. of the Korea institute of Electronic Communication Science*, vol. 10, no. 11, 2015, pp. 1239-1244.

[4] S. Lin and D. J. Costello, *Error Control Coding*, Saddle River: Prentice Hall, 2004.

[5] Y. Wang and X. Zhang, "Research on the Steganography Robustness Method of BCH Single Coefficient," *The Open Automation and Control Systems J.*, vol. 7, no. 1, 2015, pp. 910-915.

[6] H. Wei, X. Cui, Q. Zhang, and Y. Jin, “An Enhanced Decoder for Multiple-bit Error Correcting BCH Codes,” In *Proc. IEEE 11th Int. Conf. on ASIC*, Chengdu, China, Nov. 2015.

[7] E. Stavinov, “A Practical Parallel CRC Generation Method,” *Circuit Cellar*, Issue 234, no. 1, Jan. 2010, pp. 38-45.

[8] G. Albertengo and R. Sisto, “Parallel CRC Generation,” *IEEE Micro*, vol. 10, no. 5, Oct. 1990, pp. 63-71.

[9] W. Choi, J. Lee, and W. Sung “Design of High-performance Parallel BCH Decoder for Error Collection in MLC Flash Memory,” *J. of the Korea Contents Association*, vol. 16, no. 3, Mar. 2016, pp. 91-101.

[10] J. Zhang, Z. Wang, Q. Hu, and J. Xiao, “Optimized Design for High-speed Parallel BCH Encoder,” In *Proc. 2005 IEEE Int. Workshop on VLSI Design and Video Technology*, Suzhou, China, May 2005.

저자 소개

갈홍주(Hong-Ju Kal)

2018년 서울과학기술대학교 전자IT미디어공학과 졸업(공학사)
2018년~현재 연세대학원 전기전자공학과 재학
※ 관심분야 : 컴퓨터 구조, 메모리 구조

문현찬(Hyun-Chan Moon)

2012년~현재 서울과학기술대학교 전자IT미디어공학과 학사과정
※ 관심분야 : Error correction code, Hardware networking

이원영(Won-Young Lee)

2006년 KAIST 전기 및 전자공학과 졸업(공학사)
2008년 KAIST 대학원 전기 및 전자공학과 졸업(공학석사)
2012년 KAIST 대학원 전기 및 전자공학과 졸업(공학박사)
2012~2015년 삼성전자 메모리사업부 책임연구원
2015년~현재 서울과학기술대학교 전자IT미디어공학과 조교수
※ 관심분야 : VLSI, High-speed Serial Interface