m-비트 병렬 BCH 인코더의 새로운 설계 방법
A new design method of m-bit parallel BCH encoder

이 준*, 우 중 재**
June Lee , Choong-Chae Woo

요 약

차세대 멀티 레벨 셀 플래시 메모리들을 위해 복잡도가 낮은 에러 정정 코드 구현에 대한 요구가 커지고 있다. 일반적으로 부 표현 (sub-expression) 들을 공유하는 것은 복잡도와 칩 면적을 줄이기 위한 효과적인 방법이다. 본 논문에서는 직렬 선형 귀환 쉬프트 레지스터 구조를 기반으로 부 표현들을 이용한 저 복잡도 m-비트 병렬 BCH 인코더 구현 방법을 제안한다. 또한, 부 표현들을 탐색하기 위한 일반화된 방법을 제시한다. 부 표현들은 패리티 생성을 위해 사용하는 행렬(생성 행렬, generator matrix)의 부 행렬 (sub-matrix)과 다른 변수들의 합과의 행렬 연산에 의해 표현된다. 부 표현들의 수는 m개로 한정되며, 탐색된 부 표현들은 다른 병렬 BCH 인코더 구현을 위해 공유되어질 수 있다. 본 논문은 구현 과정에서 다수의 팬 아웃에 의해 발생하는 문제점(지연)의 해결이 아닌 복잡도(로직 사이즈) 감소에 그 목적이 있다.

Abstract

The design of error correction code with low complexity has a good attraction for next generation multi-level cell flash memory. Sharing sub-expressions is effective method to reduce complexity and chip size. This paper proposes a new design method of m-bit parallel BCH encoder based on serial linear feedback shift register structure with low complexity using sub-expression. In addition, general algorithm for obtaining the sub-expression is introduced. The sub-expression can be expressed by matrix operation between sub-matrix of generator matrix and sum of two different variables. The number of the sub-expression is restricted by m. The obtained sub-expressions can be shared for implementation of different m-parallel BCH encoder. This paper is not focused on solving a problem (delay) induced by numerous fan-out, but complexity reduction, expecially the number of gates.

Keywords : Parallel BCH encoder, sub-expression, error correction code.

I. 서 론

차세대 플래시 메모리(flash memory) 시스템에서 멀티 레벨 셀(multi-level cell)의 개념은 기록 밀도의 증가와 더불어 비트 당 비용을 줄이기 위한 매력적인 방법 중 하나이다[1][2]. 그러나, 인접 프로그램 레벨들 사이에 줄어든 간격은 멀티 레벨 셀 메모리들의 신뢰성을 크게 감소시킨다. 메모리 시스템에 적합한 에러 정정 코드(error correction code)를 사용함으로 데이터 복원의 신뢰성 확보가 가능하다. 메모리 시스템을 위해 가장 적합한 에러 정정 코드는 선형 블록 코드 (linear block code) 라고 알려져 있다[1]. 싱글 레벨 셀 메모리시스템에서는 에러 정정을 위한 선형 블록 코드로서 1비트 에러 정정 능력을 가진 해밍 코드(hamming code)가 사용되어졌다[3]. 그러나, 멀티 레벨 셀 메모리 시스템의 신뢰성 확보를 위해 해밍 코드는 충분한 에러 정정 능력을 가지지 못한다. 멀티 레벨 셀 메모리 시스템을 위해서는 BCH 코드와 같은 랜덤 에러 정정 (random error correction) 능력을 가진 선형 블록 코드가 바람직하다. BCH 코드는 멀티 레벨 셀 메모리 시스템을 위해 가장 널리 사용되어지고 있는 에러 정정 코드 중 하나이고 현재 고속 및 저 복잡도 BCH코드의 구현에 그 관심이 모아지고 있다[1][2][4][5][6].

BCH 인코더는 일반적으로 직렬 선형 귀환 쉬프트 레지스터 (serial linear feedback shift register) 아키텍처를 이용하여 구현되어진다[3][4][7]. 그러나 고속 처리가 요구되는 시스템에서는 이 구조를 기반으로 구현된 인코더의 클록 주파수는 요구되는 데이터 전송률을 따라 갈수가 없다. 따라서 고속 시스템을 위해 병렬 BCH 인코더의 적용은 필수적이다.

선형 귀환 쉬프트 레지스터를 이용한 다양한 병렬 처리 구조들이 제안되어졌다[1][4][5]. 그 중 재귀 공식을 이용한 병렬 처리 방법이 비교적 병렬 BCH 인코더를 구현하기 위한 명확한 방법을 제시하였다[4]. 재귀 공식을 이용한 병렬 처리 구조는 패리티 생성을 위한 수식들에서 공통적으로 사용되는 계산 식(부 표현, sub-expression)을 찾아 공유함으로써 복잡도를 줄이는 방

* LG전자 D&S 연구소
** 한서대학교 컴퓨터공학과
투고 일자 : 2010. 6. 1 수정 일자 : 2010. 7. 24
게재확정일자 : 2010. 7. 29

법이다. 하지만, 본 구조에서 저 복잡도 구현을 위한 핵심인 최적화된 부 표현들의 획득은 알고리즘 기반이 아닌, 가능한 모든 경우의 탐색(전체 탐색)에 의해 수행되어지며, 이렇게 탐색된 부 표현들은 목표로 하고 있는 에러 정정 코드에 초점을 맞춰 선택되어졌기 때문에 다른 병렬 BCH 인코더 구현을 위해 공유되어질 수 없다는 단점이 있다. 부 표현들의 사용은 구현의 복잡도와 로직 크기를 효과적으로 줄이는 방법이다. 본 논문에서는 m-비트 병렬 BCH 인코더를 구현하기 위해 부 표현들을 탐색하기 위한 일반화된 방법을 제안한다. 부 표현들은 패러티 생성을 위해 사용하는 행렬 (생성 행렬)의 부 행렬(랭크가 풀인 정방 행렬)과 두 개의 다른 변수들의 합과의 행렬 연산의 의해 표현되며, 부 표현들의 수는 m개로 한정되어진다. 패리티 생성을 위한 각 수식(나머지 레지스터의 다음 상태를 나타내는 각 수식)들은 부 표현들의 선형 조합과 나머지 레지스터의 현재 상태와의 합에 의해 표현되어진다.

멀티 레벨 셀 메모리 시스템은 한 칩에 에러 정정 능력이 다른 다수의 BCH 코드들을 요구하고 있다[1][2]. 서로 공용하는 로직 부분 없이 한 칩에 다수의 BCH 코드 구현은 복잡도와 로직 크기 증가를 초래하는 비효율적인 방법이다. BCH 코드 구현에 있어서 디코딩 블록들은 약간의 변경을 통해 다른 BCH 코드 구현을 위해 공용이 가능한 반면 BCH 인코더는 그렇지 않다. 제안된 방법에 의해 탐색된 부 표현들은 서로 다른 BCH 인코더 구현을 위해 서로 공유되어 질 수 있음을 특징으로 하며, 결과적으로 한 칩에 복잡도와 로직 크기가 줄어든 다수의 BCH 인코더의 효과적인 구현을 가능하게 한다.

본 논문의 구성은 다음과 같다. II장에서는 선형 귀환 쉬프트 레지스터 기반의 병렬 BCH 인코더 구조에 대하여 기술하고, 제안된 부 표현을 이용한 저 복잡도 병렬 BCH 인코더 구현 방법을 소개한다. III 장에서는 예제를 통하여 제안한 방법을 이용한 BCH 인코더 구현 방법을 기술하고, IV장에서는 본 논문의 결론을 맺는다.

II. 부 표현을 이용한 저 복잡도 병렬 BCH 인코더 구현 기법

이진 BCH(n,k) 코드는 k-비트 메시지 $(v_1, v_2, ..., v_k)$를 n-비트 코드워드 $(c_1, c_2, ..., c_n)$로 부호화 한다[3][7]. 여기서, $c_i, v_i \in Galois\ field,\ GF(2)$이다. 메시지를 차수가 $k-1$인 다항식의 계수 $(V(x) = v_{k-1}x^{k-1} + v_{k-2}x^{k-2} +, ..., v_1x + v_0)$로서 고려하고, 코드워드를 차수가 $n-1$인 다항식의 계수 $(C(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} +, ..., c_1x + c_0)$로서 고려할 때, BCH 코드의 시스터 매틱(systematic) 인코더는 다음과 같은 수식에 의해 구현되어질 수 있다. $C(x) = V(x)x^{n-k} + REM(V(x)x^{n-k})_{G(x)}$, 여기서, $G(x) = x^R + g_{R-1}x^{R-1} + g_{R-2}x^{R-2}, ..., g_1x + 1$는 차수가 $R(=n-k)$인 생성 다항식이며, $REM(A)_B$는 A를 B로 나눈 나머지를 의미한다. 이 인코더는 선형 귀환 쉬프트 레지스터 (또는 나머지 레지스터)를 이용하여 구현할 수 있다. k 사이클 후 나머지 $(REM(V(x)x^{n-k})_{G(x)})$는 비트 형태로 R개의 레지스터들에 저장되며, R개의 레지스터들에 저장된 비트들은 패리티 비트를 형성한다.

만약, 병렬 BCH 인코더가 m-비트를 동시에 처리할 수 있다면, 병렬 BCH 인코더의 처리 속도는 직렬 BCH 인코더 보다 m배가 빠르다. m-비트 병렬 BCH 인코더 $(m>1)$를 구현하기 위해서는, 우리는 시간 t의 나머지 레지스터의 상태와 m비트 입력 비트들을 이용하여, 시간 $t+m$에서의 나머지 레지스터의 상태를 알 수 있어야 한다. $X(t) = [x_{R-1}(t), ..., x_0(t)]^T$를 시간 t에서 나머지 레지스터의 상태라 하고,

$Z(t+m) = [z(t+1), z(t+2), z(t+m)|0, ..., 0]^T$를 m비트 입력이라고 가정할 때, 시간 $t+m$에서의 나머지 레지스터 상태는 다음과 같이 주어진다[4].

$$X(t+m) = F^m \times [X(t) + Z(t+m)] \tag{1}$$

여기서, 패리티 생성행렬 (F^m)은 F^1을 m번 곱함으로써 계산되어지며, F^1은 수식 (2)와 같다.

$$F^1 = \begin{bmatrix} g_{R-1} & 1 & 0 & 0 & \circ & \circ & \circ & 0 \\ g_{R-2} & 0 & 1 & 0 & \circ & \circ & \circ & 0 \\ g_{R-3} & 0 & 0 & 1 & \circ & \circ & \circ & 0 \\ \circ & \circ & \circ & \circ & & \circ & & \circ \\ \circ & \circ & \circ & \circ & & \circ & & \circ \\ \circ & \circ & \circ & \circ & & \circ & & \circ \\ g_1 & 0 & 0 & 0 & \circ & \circ & \circ & 1 \\ 1 & 0 & 0 & 0 & \circ & \circ & \circ & 0 \end{bmatrix}_{R(=n-k)\times R} \tag{2}$$

F^1의 첫 번째 열은 생성 다항식의 계수로 구성된다. F^i와 F^{i-1}의 관계는 다음과 같이 주어진다.

$$F^i = \left[F^{i-1} \begin{bmatrix} g_{R-1} \\ g_{R-2} \\ \circ \\ \circ \\ \circ \\ g_1 \\ 1 \end{bmatrix} \quad \begin{matrix} \text{the first } R-1 \\ \text{the columns of } F^{i-1} \end{matrix} \right]_{R\times R} \tag{3}$$

그림 1은 수식 (1)을 기반으로 한 기존 병렬 BCH 인코더 구조를 나타낸다. 나머지 레지스터의 상위 m-비트(현재 나머지 레지스터의 상태)는 m-비트 입력과 함께, 조합 로직 (Combinati-

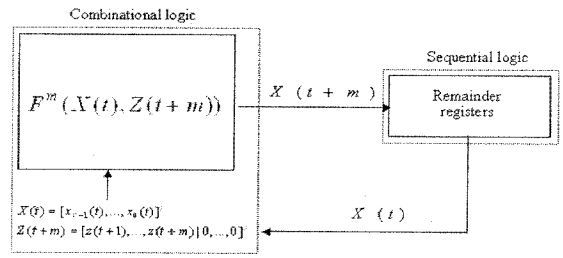

그림 1. 기존 병렬 BCH 인코더 구조
Fig.1. Conventional parallel BCH encoder structure

-onal logic) 에 입력되어지며, 조합 로직의 출력(다음 나머지 레지스터 상태)은 시간 $t+m$에서 나머지 레지스터의 상태로서 순차 회로(Sequential logic)에 입력되어진다. 이 구조는 두 블록을 포함한다. 하나는 간단한 순차 회로이고, 다른 하나는 복잡한 조합 로직 네트워크이다. 저 복잡도 병렬 BCH 인코더의 디자인은 이 복잡한 조합 로직 회로를 간소화하는 이슈에 집중해야 한다는 것을 쉽게 알 수 있다. 일반적으로 부 표현을 공유하는 것은 구현의 복잡도와 로직 사이즈를 줄이는 효과적인 방법이다[4]. 부 표현이란 다수의 수식이 공통으로 공유하는 수식을 의미한다.

수식 (2)의 F^1을 살펴보면 그것의 랭크(Rank)가 R이라는 것을 쉽게 확인할 수 있다. 따라서, F^1의 곱들에 의해 형성된 행렬 $(F^m(i=m))$의 랭크 또한 R이다. 이 사실은 F^m에는 랭크가 m인 부 행렬, 즉 m-차원 생성 집합(Span set)으로 구성된 부 행렬이 반드시 하나 이상 존재함을 의미한다. 본 논문은 이 특징을 찾아내었고, 이 특징을 이용하여, 부 표현들을 얻기 위한 일반화된 방법을 제안하였으며, 이 방법을 이용하여 조합 로직의 복잡도를 간소화 하였다.

F^i는 수식 (4)와 같이 다시 표현할 수 있다.

$$F^i = \begin{bmatrix} F^{i-1}\begin{bmatrix} g_{R-1} \\ g_{R-2} \\ \circ \\ \circ \\ \circ \\ g_1 \\ 1 \end{bmatrix} & \text{the first } R-1 \text{ columns of } F^{i-1} \end{bmatrix} \begin{bmatrix} I_{(R-i)\times(R-i)} \\ O_{i\times(R-i)} \end{bmatrix}_{R\times R}$$

$$= \begin{bmatrix} F^i_{R\times i} \end{bmatrix} \begin{bmatrix} I_{(R-i)\times(R-i)} \\ O_{i\times(R-i)} \end{bmatrix}_{R\times R} \quad (4)$$

여기서, F^i의 랭크는 항상 R 이며, I와 O은 각각 단위행렬과 제로행렬을 의미한다. 수식 (1)에 (4)를 대입한 후 계산 및 정리하면, 나머지 레지스터의 다음상태 $(X(t+i))$는 다음 두 행렬식들의 합에 의해 표현되어짐을 알 수 있다.

$$X(t+i) = X1(t+i) + X2(t+i) \quad (5)$$

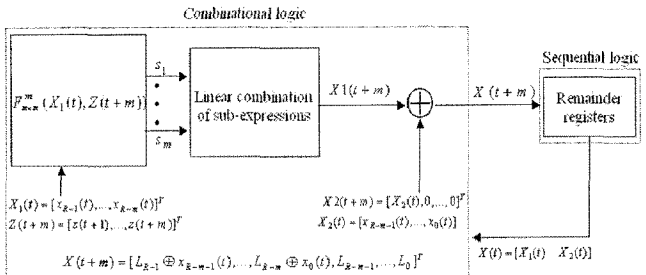

그림 2. 부 표현들을 이용한 m-비트 BCH 인코더 구조
Fig. 2. m-bit BCH encoder structure using sub-expression

수식 (5)에서 첫 번째 행렬식 $(X1(t+i))$은 아래 수식 (6)과 같이 주어진다.

$$X1(t+i) = F^i_{R\times i} \times \begin{bmatrix} x_{R-1}(t)+z(t+1) \\ \circ \\ \circ \\ \circ \\ x_{R-i}(t)+z(t+i) \end{bmatrix}_{i\times 1} = \begin{bmatrix} L_{R-1} \\ \circ \\ \circ \\ \circ \\ L_0 \end{bmatrix}_{R\times 1} \quad (6)$$

이 행렬은 i-비트 입력 및 나머지 레지스터 i개의 현재 상태들 간에 합과 $F^i_{R\times i}$와의 행렬 연산에 의해 구해지며, 이 행렬을 구성하는 R개의 수식들은 대부분은 다수의 항들로 구성될 것이다. 수식 (5)의 두 번째 행렬식 $(X2(t+i))$는 수식 (7)과 같이 주어진다.

$$X2(t+i) = \begin{bmatrix} I_{(R-i)\times(R-i)} \\ O_{i\times(R-i)} \end{bmatrix} \begin{bmatrix} x_{R-i-1}(t) \\ \circ \\ \circ \\ \circ \\ x_0(t) \end{bmatrix} = \begin{bmatrix} x_{R-i-1}(t) \\ x_{R-i-2}(t) \\ \circ \\ \circ \\ \circ \\ x_0(t) \\ 0 \\ \circ \\ \circ \\ \circ \\ 0 \end{bmatrix}_{R\times 1} \quad (7)$$

이 행렬은 0 또는 나머지 레지스터 $R-i$ 개의 현재 상태로 구성되어 있고, 이 행렬을 구성하는 $R-i$개의 수식들 각각은 단항으로 구성되어 있다. 따라서 수식 (5)를 구성하는R개 수식의 복잡도는 거의 수식 (6)을 구성하는 각 행 수식의 복잡도와 비례함을 알 수 있다. 수식 (6)은 조합 로직에 의해 구현가능하며, 부 표현들을 공유하여 수식 (6)을 구성하는 각 수식들을 표현함으로서 저 복잡도(게이트 수 감소)를 실현할 수 있다.

F^i가 풀 랭크(Full rank)라는 사실로 부터 수식 (6)에 포함된 행렬 $F^i_{R\times i}$는 랭크가 i인 부 행렬 $F^i_{i\times i}$를 한 개 이상 포함하게 된다. 따라서 $F^i_{R\times i}$의 모든 행들은 $F^i_{i\times i}$각 행들의 선형 조합(Linear combination)에 의해 표현되어질 수 있다. 결과적으로, 수식 (6)을 구성하는 각 행(수식)들은 i-비트 입력 및 나머지 레지스터 i개의 현재 상태들 간에 합과 $F^i_{i\times i}$와의 행렬 연산에 의해 얻어진 부 표현들의 선형 조합으로 나타낼 수 있다. 부 표현 행렬(S)은 수식 (8)과 같이 주어진다.

$$S = F^i_{i\times i} \times \begin{bmatrix} x_{R-1}(t)+z(t+1) \\ \circ \\ \circ \\ \circ \\ x_{R-i}(t)+z(t+i) \end{bmatrix}_{i\times 1} = \begin{bmatrix} s_1 \\ \circ \\ \circ \\ \circ \\ s_i \end{bmatrix}_{i\times 1} \quad (8)$$

부 표현의 개수는 i이며, 다수의 $F^i_{i\times i}$가 존재할 때 그 중 부 표현(s)들을 표현하기 위해 필요로 하는 게이트 수를 최소로 하는 것을 선택한다. 수식 (6)의 각 행은 수식 (8)을 구성하는 부 표현들의 선형 조합에 의해 표현될 수 있다. 그림 2는 제안한 방

법에 의해 얻어진 부 표현들을 이용한 m-비트 병렬 BCH 인코더 구조를 보여준다. 조합 로직은 두 개의 입력을 기반으로 나머지 레지스터의 다음 상태를 출력한다. 이 두 입력 중 하나는, m-비트 병렬 BCH 인코더 입력 $Z(t+m)=[z(t+1),...,z(t+m)]^T$이며, 다른 하나는 순차회로의 출력인 나머지 레지스터의 현재 상태 $X(t)=[X_1(t)^T\ X_2(t)]$이다. 이 두 입력들은 $F^m_{R\times m}$으로 부터 얻어진 부 행렬 $F^m_{m\times m}$과 행렬 연산에 의해 m개의 부 표현들을 생성한다. 이 부 표현들의 선형 조합은 수식 (6)를 표현할 수 있다. 따라서, 나머지 레지스터의 다음 상태 $X(t+m)$는 부 표현들의 선형 조합으로 표현된 수식 (6)과 수식 (7)의 $\oplus$ (XOR) 연산에 의해 얻어지며, 수식 (9)으로 표현되어 진다.

$$X(t+m)=[L_{R-1}\oplus x_{R-m-1}(t),...,L_{R-m}\oplus x_0(t),L_{R-m-1},...,L_0]^T \quad (9)$$

수식 (9)을 기반으로 BCH 인코더가 각 사이클 마다 m-비트 메시지를 처리할 때, 패리티 비트들은 k/m사이클 후 R개의 나머지 레지스터들에 저장되어 진다. m-비트 병렬 BCH 코드의 시스터매틱 인코딩은 k-비트 메시지 후 R개의 패러티 비트를 더함으로서 완료된다.

기존 방법에 의한 m-비트 병렬 BCH 인코더 구현은 그림 1의 방법에 의해 얻어진 수식들을 기반으로 저 복잡도와 저 로직 사이즈를 위해 부 표현의 최적화를 수행 한다[4]. 그러나 이 부 표현의 최적화는 명확한 알고리즘에 의해서가 아닌 시행착오 의해 수행되며 에러 정정 능력이 큰 코드의 구현에서는 조사해야할 가짓수가 너무 방대하기 때문에 이 방법의 적용은 매우 비효율적이다. 또한 획득된 부 표현들은 목표로 하고 있는 에러 정정 코드에 초점을 맞춰 선택되어졌기 때문에 다른 m-비트 병렬 BCH 인코더 구현을 위해 공유되어 질 수 없다. 반면에 본 방법은 m-비트 병렬 BCH 인코더 구현을 위해 명백한 알고리즘을 기반으로 얻어진 m-차원 생성 집합으로 구성된 부 표현들을 사용하기 때문에, 나머지 레지스터의 다음 상태들을 표현하기 위해 공통으로 사용할 저 복잡도(최적화) 부 표현들의 지루한 탐색을 요구하지 않는다. 또한, 제안된 기법에 의해 생성된 부 표현들은 m-차원 공간의 생성 집합을 형성하기 때문에 다른 종류의 m-비트 병렬 BCH 인코더와 관계된 수식 (6)을 구현하기 위해 공유가 가능하다. 따라서 한 칩에 특징이 다른 다수의 m-비트 병렬 BCH 코드들을 요구하는 시스템에 제안된 방법의 적용은 시스템의 저 복잡도 달성을 위한 효율적인 솔루션을 제공할 수 있다.

III. 제안된 방법을 이용한 BCH 인코더 구현

이 장은 간단한 m-비트 병렬 BCH 인코더를 제안한 알고리즘을 이용하여 구현함으로서 제안한 방법의 유용성을 보이는 데 있다. 구현의 예로 사용한 BCH 코드는 $(n=80,k=72,t=1)$이고 $m=4$이다. 여기서 t는 에러 정정 능력을 나타낸다. BCH(80, 72, 1) 코드는 BCH(255, 247, 1)의 짧은 형태의 코드(Shortened code) 이고, 메시지 및 코드 워드 심벌은 $GF(2^8)$ 상의 원소이다. BCH(80, 72, 1) 코드의 인코딩을 위한 생성 다항식은 $G(x)=1+x^2+x^3+x^4+x^8$이다.

4-비트 병렬 BCH (80, 72, 1) 인코더 구현을 위한 생성 행렬 (F^4)은 F^1을 4번 곱 합으로써 얻어진다. 행렬 F^1과 F^4는 아래와 같다.

$$F^1=\begin{bmatrix}01000000\\00100000\\00010000\\10001000\\10000100\\10000010\\00000001\\10000000\end{bmatrix},\ F^4=\left[\begin{array}{c|c}1000&1000\\1100&0100\\1110&0010\\0111&0001\\1011&0000\\0101&0000\\0010&0000\\0001&0000\end{array}\right]=\left[F^4_{8\times4}\ \middle|\ \begin{matrix}I_{4\times4}\\0_{4\times4}\end{matrix}\right] \quad (10)$$

F^4는 풀 랭크 $(R=n-k=8)$를 가진다. 따라서 행렬 $F^4_{8\times4}$에는 랭크가 4인 부 행렬$(F^4_{4\times4})$들이 반드시 존재한다. $F^4_{8\times4}$ 안에 존재하는 부 행렬들의 수는 55개이다. 55개의 부 행렬들 중 부 표현들을 구현하기 위해 사용되어지는 게이트 수를 최소로 하는 부 행렬을 선택하였다. 선택되어진 부 행렬은 $F^4_{8\times4}$의 1, 2 , 7 및 8 행들로 구성된다. 부 행렬을 이용한 부 표현 행렬은 수식 (8)을 통해 다음과 같이 주어진다.

$$S=\begin{bmatrix}s_1\\s_2\\s_3\\s_4\end{bmatrix}=\begin{bmatrix}x_7(t)+z(t+1)\\(x_7(t)+z(t+1))+(x_6(t)+z(t+2))\\x_5(t)+z(t+3)\\x_4(t)+z(t+4)\end{bmatrix} \quad (11)$$

부 표현의 수는 $m(=4)$와 동일하며, 이 부 표현들의 선형 조합들에 의해 $X1(t+4)$는 아래와 같이 주어진다.

$$X1(t+4)=\begin{bmatrix}L_7\\L_6\\L_5\\L_4\\L_3\\L_2\\L_1\\L_0\end{bmatrix}=\begin{bmatrix}s_1\\s_2\\s_2+s_3\\s_1+s_2+s_3+s_4\\s_1+s_3+s_4\\s_1+s_2+s_4\\s_3\\s_4\end{bmatrix} \quad (12)$$

또한, 수식 (12)와 4-비트 병렬 BCH(80, 72, 1) 인코더와 관계된 $X2(t+4)$과의 합에 의해 나머지 레지스터의 다음 상태 $(X(t+4))$를 나타낼 수 있고 그 결과는 다음과 같다.

$$X(t+4)=\begin{bmatrix}x_7(t+4)\\x_6(t+4)\\x_5(t+4)\\x_4(t+4)\\x_3(t+4)\\x_2(t+4)\\x_1(t+4)\\x_0(t+4)\end{bmatrix}=\begin{bmatrix}s_1+x_3(t)\\s_2+x_2(t)\\s_2+s_3+x_1(t)\\s_1+s_2+s_3+s_4+x_0(t)\\s_1+s_3+s_4\\s_1+s_2+s_4\\s_3\\s_4\end{bmatrix} \quad (13)$$

그림 1의 방법에 의해 구현된 4-비트 병렬 BCH (80, 72, 1) 인코더의 게이트(XOR) 수는 36인 반면, 제안한 방법에 의해 구현된 4-비트 병렬 BCH (80, 72, 1) 인코더의 게이트 수는

18으로 감소함을 수식 (13)을 통해 확인할 수 있다. 그림 1의 방법(기존 방법)을 이용하여 수식 (13)를 구현하기 사용되는 총 XOR개수, 36은 다음과 같이 산출된다. 수식 (11)에서 부 표현 s_1, s_2, s_3 및 s_4를 구현하기 위해 사용되는 XOR수는 각각 1, 3, 1 및 1개이다. 수식 (13)에서 사용되는s_1, s_2, s_3 및 s_4의 개수는 각각 4개이다. 따라서 수식 (14)에서 부 표현들을 표현하기 위해 사용되는 XOR 개수는 24(=4×1 + 4×3 + 4×1 + 4×1)개 이며, 부 표현들과 이전 나머지 레지스터 상태들과의 연산을 위해 필요한 XOR개수가 12개임을 확인할 수 있다. 결과적으로, 그림 1의 방법을 이용하여 수식 (13)을 구현하기 위해 사용되는 XOR개수는 36(=24+12)개가 된다. 그림 2의 방법(제안한 방법)을 이용하여 수식 (13)를 구현하기 사용되는 총 XOR개수, 16은 다음과 같이 산출된다. 제안한 방법인 부 표현을 공유하여 수식 (13)을 구현할 때, 부 표현들 각각 구현한 후 부 표현들의 결과 값을 이용하기 때문에 부 표현들의 구현을 위해 사용되는 총 XOR 개수는 6(=1+3+1+1)개 된다. 따라서 부 표현들의 구현을 위해 필요한 XOR 개수 6(=1+3+1+1)개와 부 표현들과 이전 나머지 레지스터 상태들과의 연산을 위해 필요한 XOR개수12개를 더함으로써 제안한 방법을 이용하여 수식 (13)를 구현하기 위해 사용되는 총 XOR개수 18(=6+12)을 얻을 수 있다.

4-비트 병렬 BCH (80, 72, 1) 인코더가 매번 4-비트 씩 처리할 때, 패리티 비트들 (8-비트)은 수식 (13)을 18(=72/4)번 반복함으로 얻어지며 그 결과는 8개의 나머지 레지스터에 저장된다. 이 8-비트 패러티를 72비트 메시지 다음에 부가함으로써, 4-비트 병렬 BCH (80, 72, 1) 인코더는 시스터매틱 인코딩을 완료하게 된다.

4-비트 병렬 BCH (80, 72, 1) 인코더 구현을 위해 생성된 부 표현들은 $m(=4)$차원 공간의 생성 집합이기 때문에 에러 정정이 다른 4-비트 병렬 BCH 인코더 구현을 위해 공유되어 질 수 있다. 본 장에서는 제안된 방법으로 4-비트 병렬 BCH (80, 72, 1) 인코더 구현을 위해 획득된 부 표현들을 이용하여 에러 정정 능력이 다른 4-비트 병렬 BCH (88, 72, 2) 인코더를 구현하였다. BCH (88, 72, 2)의 생성 다항식은 $G(x)=1+x^2+x^3+x^4+x^8+x^9+x^{10}+x^{11}+x^{13}+x^{14}+x^{16}$이며, BCH(255, 239, 1)의 짧은 형태의 코드 (Shortened code)이다. 4-비트 병렬 BCH (88, 72, 2)인코더 관계된 $X1(t+4)$은 4-비트 병렬 BCH (80, 72, 1) 인코더 구현을 위해 획득된 부 표현들의 선형 조합에 의해 표현되어질 수 있고, 그 결과는 다음과 같다.

$$X1(t+4)=\begin{bmatrix}L_{15}\\L_{14}\\L_{13}\\L_{12}\\L_{11}\\L_{10}\\L_{9}\\L_{8}\\L_{7}\\L_{6}\\L_{5}\\L_{4}\\L_{3}\\L_{2}\\L_{1}\\L_{0}\end{bmatrix}=\begin{bmatrix}s_2+s_3\\s_2+s_2+s_4\\s_4\\s_1+s_2+s_3\\s_1+s_3+s_4\\s_3+s_4\\s_2+s_3+s_4\\s_4\\s_2+s_3\\s_1+s_2+s_3+s_4\\s_2+s_4\\s_1\\s_2\\s_2+s_3\\s_1+s_2+s_3+s_4\\s_2+s_4\end{bmatrix} \tag{14}$$

수식 (14)와 BCH (88, 72, 2) 인코더와 관계된 $X2(t+4)$와의 합에 의해 나머지 레지스터의 다음 상태 ($X(t+4)$)를 나타낼 수 있고 그 결과는 다음과 같다.

$$X(t+4)=\begin{bmatrix}L_{15}\\L_{14}\\L_{13}\\L_{12}\\L_{11}\\L_{10}\\L_{9}\\L_{8}\\L_{7}\\L_{6}\\L_{5}\\L_{4}\\L_{3}\\L_{2}\\L_{1}\\L_{0}\end{bmatrix}=\begin{bmatrix}s_2+s_3+x_{11}(t)\\s_2+s_2+s_4+x_{10}(t)\\s_4+x_9(t)\\s_1+s_2+s_3+x_8(t)\\s_1+s_3+s_4+x_7(t)\\s_3+s_4+x_6(t)\\s_2+s_3+s_4+x_5(t)\\s_4+x_4(t)\\s_2+s_3+x_3(t)\\s_1+s_2+s_3+s_4+x_2(t)\\s_2+s_4+x_1(t)\\s_1+x_0(t)\\s_2\\s_2+s_3\\s_1+s_2+s_3+s_4\\s_2+s_4\end{bmatrix} \tag{15}$$

위 식과 같이 4-비트 병렬 BCH (88, 72, 2) 인코더는 4-비트 병렬 BCH (80, 72, 1) 인코더 구현을 위해 생성된 부 표현들을 공유함으로써 복잡한 절차 없이 해 쉽게 구현될 수 있다. 그림 1의 방법에 의해 구현된 4-비트 병렬 BCH (88, 72, 2) 인코더의 게이트(XOR) 수는 82인 반면, 제안한 방법에 의해 구현된 4-비트 병렬 BCH (88, 72, 2) 인코더의 게이트 수는 36으로 감소함을 수식 (15)를 통해 확인할 수 있다. 그림 1 및 2 방법들을 이용하여 수식 (15)을 구현하기 위해 필요한 총 XOR개수, 82개 및 36개라는 값은 수식 (13)을 구현하기 필요한 총 XOR개수를 구하는 방법과 동일하다.

Ⅳ. 결 론

본 논문에서는 부 표현들을 이용하여 저 복잡도 병렬 BCH 인코더를 구현하기 위한 일반화된 방법을 제시하였다. 부 표현들은 패러티 생성 행렬 및 부 행렬의 수학적 특징을 기반으로 명확한 알고리즘에 의해 얻어진다. 이 생성된 부 표현들은 에러 정정

능력이 다른 BCH 인코더 구현을 위해 공유가 가능하다. 제안한 방법은 기존 방법과 비교해 저 복잡도 m-비트 병렬 BCH 코드 구현을 위해 기여할 수 있고, 이 기여도는 m과 t의 값과 비례한다. 4-비트 병렬 BCH (80, 72, 1) 및 BCH (80, 72, 2) 인코더들의 구현을 통해 제안한 방법의 실현 가능성과 특징을 살펴보았다. 이 구현 결과에서, 제안한 방법은 기존 방법과 비교해 구현에 필요한 게이트 수가 거의 반임을 확인할 수 있었다. 따라서 제안된 방법은 한 칩에 특징이 다른 다수의 BCH 코드를 요구하는 시스템의 저 복잡도 실현을 위한 솔루션으로서 제공될 수 있다.

참 고 문 헌

[1] W. Liu, J. Rho, and W. Sung, "Low-power high-throughput BCH error correction VLSI design for multi-level cell NAND flash memories," in *Proc. Int. Workshop SiPS*, pp. 248 - 253, 2006.

[2] X. Wang et al., "A high-speed two-cell BCH decoder for error correnting in MLC NOR flash memories,", IEEE Transaction on Circuit and Systems II: Express Briefs, vol. 56, no. 11, pp 865-869, 2009.

[3] S. Lin and D. J. Costello, Error control coding: Fundamentals and application (second edition), Prentice Hall, 2004.

[4] J. Zhang, Z. Wang, Q. Hu, and J. Xiao, "Optimized design for highspeed parallel BCH encoder," in *Proc. IEEE Int. Workshop. VLSI Des. & Video Tech.*, pp. 97 - 100, May 2005.

[5] X. Zhang and K. K. Parhi, "High-speed architecture for parallel long BCH encoders," *IEEE Trans. On Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 7, pp. 872-877, July 2005.

[6] K. K. Parhi, "Eliminating the Fanout Bottleneck in Parallel Long BCH Encoders,"*IEEE Tran. on Circuits and Systems*, vol. 51, no. 3, pp. 512-516, Mar. 2004.

[7] S. B. Wicker, "Error Control Systems for Digital Communication and Storage," Prentice Hall, Upper addle River, New Jersey, 1995.

이 준 (June Lee)

1998년 2월 동국대 전자공학과 (공학사)
2000년 3월 동국대 전자공학과 (공학석사)
2003년 2월 동국대 전자공학과 (공학박사)
2003년 3월 ~ 2006년 2월 삼성종합기술원 책임연구원
2007년 1월 ~ 현재 LG전자 D&S연구소 선임연구원
※주관심분야 : 디지털신호처리, 채널코딩, 저장기록장치

우 중재 (Choong-Chae Woo)

2000년 2월 순천대 전자공학과 (공학사)
2002년 2월 연세대 전기전자공학과(공학석사)
2007년 2월 연세대 전기전자공학과(공학박사)
2007년 9월 ~ 2009년 2월 삼성전자 정보통신총괄 책임연구원
2009년 2월 ~ 현재 한서대학교 전자컴퓨터통신학과 교수
※주관심분야 : 이동통신, 디지털신호처리, 저장기록장치