셀룰라 오토마타 기반의 이중 오류정정부호†

조성진*, 허성훈**
*부경대학교 수리과학부
**부경대학교 정보보호학(협)

Double-Error Correcting Code based on Cellular Automata†

Sung-Jin Cho*, Seong-Hun Heo**
*Division of Mathematical Sciences, Pukyong National Univ.
**Dept. of Information Security, Pukyong National Univ.

요 약

오류정정부호는 메모리 시스템 설계, 다양한 디지털 데이터 통신 시스템 설계 등에 널리 이용된다. 특히 오늘날과 같이 통신망의 확대와 컴퓨터의 발전으로 인한 대용량 데이터의 교환과 처리가 요구되는 디지털 데이터 통신 시스템 설계에서는 통신 채널에서 발생하는 오류를 효율적으로 제어하기 위한 오류정정부호 장치가 필수불가결한 요소가 되었다. 본 논문에서는 셀룰라 오토마타의 간단하고, 규칙적이며, 모듈화한 특성을 이용하여 이중 오류정정부호의 효율적인 부호화 및 복호화 방법을 제안한다.

1. 서론

오늘날 많은 양의 데이터가 다양한 컴퓨터 시스템과 서브시스템 사이에서 디지털 논리 회로와 상호 연결선을 통하여 전송된다. 시스템의 신뢰성은 회로 모듈 사이에서 데이터 전송의 무오류성(error-free)에 의존한다[1]. 하지만 전자적 잡음, 장치 결함, 시간 오류 등으로 인하여 시스템에는 항상 언제 발생할지 모르는 잠재적 오류가 존재한다[1][2]. 따라서 시스템의 신뢰성을 향상시키기 위해서 오류정정부호 장치가 필수불가결한 요소가 되었다. 일반적으로 오류정정부호는 (n, k, d)부호로 나타내며, n은 부호어의 길이, k는 메시지(정보어)의 길이, d는 최소거리를 나타낸다[2][3]. 기존의 오류정정부호는 k 값이 커짐에 따라 부호화 및 복호화 회로가 복잡해지는 단점이 있다. 본 논문에서는 셀룰라 오토마타(Cellular Automata, 이하 CA)의 간단하고, 규칙적이며, 모듈화한 특성을 이용하여 이중 오류정정부호의 효율적인 부호화 및 복호화 방법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서는 셀룰라 오토마타의 정의 및 용어에 대해서 설명한다. 3장에서는 셀룰라 오토마타 기반의 오류정정부호의 부호화 방법과 복호화 방법을 제안한다. 그리고, 4장에서 결론을 맺는다.

† 본 연구는 정보통신부 2002 기초기술연구지원사업(IITA:C1-2002-053-03)의 연구비에 의해 연구되었음

2. 셀룰라 오토마타

CA는 동역학계(dynamical system)를 해석하는 하나의 방법으로 공간과 시간을 이산적으로 다루는 시스템이며, 셀룰라 공간(cellular space)의 기본 단위인 각 셀(cell)이 취할 수 있는 상태를 유한하게 처리하며, 각 셀들의 상태가 국소적인 상호작용에 의해서 동시에 갱신되는 시스템이다. 가장 간단한 구조를 가지는 1차원 CA에서는 모든 셀들이 선형으로 배열되어 있고 국소적 상호작용이 세 개의 셀, 즉 자신과 인접한 두 셀에 의해 이루어지는 CA를 3-이웃(3-neighborhood) CA라 한다. 본 논문에서 다루는 CA는 3-이웃 1

차원 CA에 국한시킨다.

CA를 설명하기 위해서 다음 기호들이 사용된다.

- i : 일차원으로 배열되어 있는 각 셀들의 위치
- t : 시간 단계
- $q_i(t)$: 시간 t에서 i번째 셀의 상태
- $q_i(t+1)$: 시간 $t+1$에서 i번째 셀의 상태

세 개의 이웃을 가지는 CA에 대한 다음상태 전이함수(transition function)는 다음과 같이 나타낸다.

$q_i(t+1) = f[q_{i-1}(t), q_i(t), q_{i+1}(t)]$ 여기서 f는 결합논리를 가지는 국소 전이함수이다. f는 3개의 변수를 가지는 Boolean 함수이므로 2^{2^3}, 즉 256개의 다음 상태 전이함수들이 있으며 이것을 CA의 rule이라고 한다.

CA의 셀들의 상태를 0과 1의 두 가지 값으로 다루고 주어진 CA의 다음상태 전이함수를 아래와 같이 표현한다. 여기서 첫 행은 시간 t에서 인접한 세 개의 셀들의 가능한 8가지 상태의 배열이고 다음 행들은 시간 $t+1$에서 i번째 셀의 갱신된 상태이다.

이웃 상태	000	100	010	001	110	011	101	111
다음 상태	0	0	1	1	1	0	1	0
(rule 102)								
다음 상태	0	1	1	1	0	0	0	1
(rule 150)								

rule 102: $q_i(t+1) = q_i(t) \oplus q_{i+1}(t)$

rule 150: $q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$

위의 rule에 대한 논리결합은 다음 식으로 표현할 수 있고 ⊕는 XOR을 나타낸다.

CA는 적용되는 rule의 논리의 종류에 따라 선형 CA(Linear CA), 가산 CA(Additive CA), 비가산 CA(Nonadditive CA)로 분류되는데 각 셀에 적용된 rule이 XOR 논리로만 이루어진 CA는 선형 CA이다. 선형 CA의 상태 전이 함수는 행렬로 표현 될 수 있고 이 행렬을 특성행렬이라 한다. 또한 셀에 적용되는 rule이 XNOR과 XOR 논리로 이루어진 CA를 여원 CA(Complemented CA)라 하고 선형 CA와 여원 CA를 가산 CA라 한다. 셀들의 rule이 AND-OR논리로 이루어진 CA를 비가산 CA라 한다. <표 1>은 가산 CA에서 선형 rule을 보여준다.

rule 60	$q_i(t+1) = q_{i-1}(t) \oplus q_i(t)$
rule 90	$q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$
rule 102	$q_i(t+1) = q_i(t) \oplus q_{i+1}(t)$
rule 150	$q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$
rule 170	$q_i(t+1) = q_{i+1}(t)$
rule 204	$q_i(t+1) = q_i(t)$
rule 240	$q_i(t+1) = q_{i-1}(t)$

<표 1> CA의 선형 rule

CA의 각 셀에 모두 같은 rule이 적용된 CA를 uniform CA, 그렇지 않고 2가지 이상의 서로 다른 rule이 적용된 CA를 hybrid CA라 한다. CA의 rule에 의해 변화되는 상태를 나타낸 상태전이 그래프의 형태에 따라 Group CA와 Nongroup CA로 분류할 수 있다. Group CA는 모든 셀들의 상태가 몇 개의 사이클을 이루며 반복되는 CA로 임의의 한 상태에 대한 이전상태가 유일하다. 이와 달리 Nongroup CA는 상태전이 그래프가 트리 구조를 이루고 있으며 상태전이 함수에 의해 얻어질 수 있는 상태인 도달 가능한 상태와 상태전이 함수에 의해 나타날 수 없는 도달 불가능한 상태로 나누어진다. CA에서 가장 왼쪽과 오른쪽의 셀은 2개의 이웃만을 가지므로 세 번째 이웃의 상태를 결정해 주어야 한다. 이것을 CA의 경계조건이라 하고 일반적으로 다음 세 가지의 경계조건을 이용한다. 제일 왼쪽과 오른쪽의 셀들이 0상태에 연결되어 있는 NBCA (Null Boundary CA), 양끝의 셀들이 서로 연결되어 있는 PBCA(Periodic Boundary CA), 가장 왼쪽(오른쪽) 셀의 다음 상태가 그 자신과 그것의 오른쪽(왼쪽) 이웃, 두 번째 오른쪽(왼쪽) 이웃 셀의 상태에 의존하는 IBCA(Intermediate Boundary CA)이다[5][6].

n개의 셀을 가지는 선형 1차원 CA에서는 현재 상태를 다음 상태로 전이시키는 전이함수를

$n \times n$ 행렬로 나타낼 수 있으며, 이것을 특성행렬이라 한다[7].

3. 셀룰라 오토마타 기반의 이중 오류정정 부호

(1) 부호화 방법

CA의 특성행렬을 이용하여 이중 오류정정부호를 생성하는 방법을 알아본다.

[정리 1] 행렬 T가 다음 두 조건을 만족한다면, CA는 $(n, k, 5)$부호를 생성한다:

(a) T의 모든 열벡터는 적어도 네 개의 1을 포함한다.

(b) T의 서로 다른 네 개 이하의 열벡터의 합은 영벡터가 아니다. □

부호화의 첫 번째 과정은 정리 1을 만족하는 T를 찾는 것이다. Mathematica를 이용하여 정리 1의 조건을 검사하는 프로그램을 작성하여 T를 구성한다.

다음은 최소거리가 5인 부호를 생성하기 위하기 T를 구성하는 알고리즘이다.

<알고리즘 I>

Step 1. rule이 <102, 102, ···, 102, 150>인 CA의 특성행렬 T_k을 구성한다.

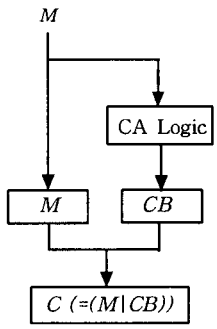
Step 2. $T_k{}^3$을 구한다.

Step 3. $T_k{}^3$의 $(1, k)$성분을 1로 바꾸고, 행렬 $\begin{bmatrix} 101010\cdots \\ 010101\cdots \end{bmatrix}$을 덧붙인다.

Step 4. Step 3에서 구한 행렬을 T_h라 하면 같은 위치에서 1을 갖지 않는 행끼리 더하여 [정리 1]을 만족하는지 검사하여 행의 수를 최대한으로 줄인다.

Step 5. Step 4에서 구한 행렬을 T로 둔다.

<알고리즘 I>에 의해서 구성된 T를 이용하여 부호어를 생성한다. k비트의 메시지(M)가 로드(load)되면 T를 이용하여 검사비트를 생성하고 메시지에 검사비트(CB)를 연결하여 부호어(C)를 생성한다. <그림1>은 제안된 방법을 보여준다.



<그림 1> 최소거리가 5인 부호어 생성

CA Logic은 TM^T을 수행하여 검사비트를 생성한다.

<예제 1> 메시지가 10비트(1010101010)일 때, 부호어를 구해보자.

<알고리즘 I>의 Step 1과 Step 2에서 구성된 T_{10}과 $T_{10}{}^3$은 각각 다음과 같다.

$$T_{10} = \begin{bmatrix} 1100000000 \\ 0110000000 \\ 0011000000 \\ 0001100000 \\ 0000110000 \\ 0000011000 \\ 0000001100 \\ 0000000110 \\ 0000000011 \\ 1000000011 \end{bmatrix}_{10\times 10}, \quad T_{10}{}^3 = \begin{bmatrix} 1111000000 \\ 0111100000 \\ 0011110000 \\ 0001111000 \\ 0000111100 \\ 0000011110 \\ 0000001111 \\ 1000000101 \\ 1100000000 \\ 0110000000 \end{bmatrix}_{10\times 10}$$

Step 3에서 (1,10)성분을 1로 바꾸고 행렬 $\begin{bmatrix} 1010101010 \\ 0101010101 \end{bmatrix}$을 연결하면,

$$T_h = \begin{bmatrix} 1111000001 \\ 0111100000 \\ 0011110000 \\ 0001111000 \\ 0000111100 \\ 0000011110 \\ 0000001111 \\ 1000000101 \\ 1100000000 \\ 0110000000 \\ 1010101010 \\ 0101010101 \end{bmatrix}_{12\times 10}$$

이 된다. T_h는 Step 4에 의해서 12행에서 9행으로 줄어든다. Step 5에서 T는 다음과 같다.

$$T=\begin{bmatrix}1111000001\\0111100000\\1011110101\\0001111000\\1100111100\\0110011110\\0000001111\\1010101010\\0101010111\end{bmatrix}_{9\times 10}$$

메시지에 대응하는 검사비트(CB)는 TM^T에 의해서 생성된다.

$$CB^T = TM^T = \begin{bmatrix}1111000001\\0111100000\\1011110101\\0001111000\\1100111100\\0110011110\\0000001111\\1010101010\\0101010111\end{bmatrix}\begin{bmatrix}1\\0\\1\\0\\1\\0\\1\\0\\1\\0\end{bmatrix}=\begin{bmatrix}0\\0\\1\\0\\1\\1\\0\\1\\1\end{bmatrix}$$

검사비트(CB)는 9비트(001011011)이된다. 따라서 부호어(C)는 19비트 (1010101010001011011)이 된다. □

다음 표는 최소거리가 5인 부호에서 메시지에 대한 검사비트의 크기를 Chowdhury, et, al.[4]의 결과와 본 연구를 비교한 것이다. 표에 의해서 <알고리즘 I>에 의해서 구성된 T를 이용한 방법이 더 우수함을 알 수 있다.

	Chowdhury [4]	본 연구
메시지 (k)	검사비트($n-k$)	검사비트($n-k$)
8	9	**8**
9	9	**9**
10	10	**9**
11	10	**9**
12	11	**9**
13	11	**10**
14	11	**10**
15	11	**10**
16	12	**11**
17	13	**11**
18	13	**12**
19	13	**12**
20	13	**12**
32	17	**16**

<표 2> 최소거리가 5인 부호의 검사비트

(2) 복호화 방법

수신된 부호어에서 오류가 있는 비트를 정정하기 위한 복호화 방법을 알아본다. 복호화하는 첫 번째 단계는 신드롬을 계산하는 것이다.

수신된 부호어가 $C'(=(M'|CB'))$이라면 신드롬은 $S(C')=HC'^T=H\,[M'|CB']^T$이다. 여기서, H는 <알고리즘 I>에 의해서 생성된 T와 단위행렬 I_k를 연결하여 형성된 패리티 검사행렬이다. 즉, $H=[T|I_k]$이다.

따라서,

$$S(C')=H\,[M'|CB']^T=[T|I_k]\,[M'|CB']^T$$
$$=TM'^T\oplus I_k\,CB'^T=TM'^T\oplus CB'^T$$

로 표현될 수 있다.

신드롬의 값이 0이면 오류가 없음을 나타내고 0이 아니면 수신된 부호어에 오류가 있음을 나타낸다.

수신된 부호어에서 M_e과 CB_e를 각각 메시지와 검사비트에 대응되는 오류 벡터라고 하자. 그러면, 수신된 메시지는 $M'=M\oplus M_e$이고 , 수신된 검사비트는 $CB'=CB\oplus CB_e$ 이다.

$$H\,[M|CB]^T\oplus H\,[M_e|CB_e]^T=S$$

여기서, $H\,[M|CB]^T=0$ 이므로,

$H\,[M_e|CB_e]^T=S$이 된다.

즉, $H[E]^T=[S]$ 에서 오류벡터 E는

$[E]^T=H^{-1}[S]$이 된다. 하지만 H^{-1}이 존재하려면 H는 $n\times n$정방행렬이어야 한다. 하지만, H가 정방행렬이 아니므로 패리티 검사 행렬 $H_{(n-k)\times n}$를 정방행렬 $T_{aug\ n\times n}$로 변환하기 위해서 k개의 추가 행을 $H_{(n-k)\times n}$에 덧붙인다. 이때 $T_{aug\ n\times n}$가 역행렬이 존재(즉, group CA)하도록 구성한다. 즉, $det[T_{aug}]=1$ 이다.

$$T_{aug}=\begin{bmatrix}[H]_{(n-k)\times n}\\\cdots\\[\text{추가행}]_{k\times n}\end{bmatrix}_{n\times n}$$

$$T_{aug}[E]^T=\begin{bmatrix}[H]\\\cdots\\[\text{추가행}]\end{bmatrix}_{n\times n}[M_e|CB_e]^T$$
$$=\begin{bmatrix}S\\\cdots\\S_{aug}\end{bmatrix}_{n\times 1}$$

$$[E]=[T_{aug}]^{-1}\begin{bmatrix}S\\S_{aug}\end{bmatrix}$$

<정의1> 복호화에 사용되는 T_{aug}는 <알고리즘 I >에 의해서 구성된 T를 이용하여 다음과 같이 구성한다.

$$T_{aug} = \begin{bmatrix} T_{(n-k)\times k} & I_{n-k} \\ I_k & O_{k\times(n-k)} \end{bmatrix}_{n\times n}$$

□

[정리 2] 정의 1에서 구성된 T_{aug}의 역행렬은 다음과 같다.

$$T_{aug}^{-1} = \begin{bmatrix} O_{k\times(n-k)} & I_k \\ I_{n-k} & T_{(n-k)\times k} \end{bmatrix}_{n\times n}$$

□

[정리 3] T_{aug}의 역행렬이 존재 ($det[T_{aug}]=1$) 하기 위한 필요충분조건은 $det[T]=1$ 이다.

□

정리 3에 의해서 T_{aug}를 구성할 때 $det[T]=1$ 인 T를 선택해야 한다.

<예제 2> 예제 1에서 생성된 부호어 $C=(10\ 10101010001011011)$가 전송 도중 4번째와 13번째 비트에서 두개의 오류가 발생했다고 하자.
즉, $C'=(M'|CB')=(10111010100000011011)$이 수신되었다고 하자.
복호화 방법에서 쓰이는 T_{aug}^{-1}는 다음과 같다.

$$T_{aug}^{-1} = \begin{bmatrix} 000000000\ 1000000000 \\ 000000000\ 0100000000 \\ 000000000\ 0010000000 \\ 000000000\ 0001000000 \\ 000000000\ 0000100000 \\ 000000000\ 0000010000 \\ 000000000\ 0000001000 \\ 000000000\ 0000000100 \\ 000000000\ 0000000010 \\ 000000000\ 0000000001 \\ \hline 100000000\ 1111000001 \\ 010000000\ 0111100000 \\ 001000000\ 1011110101 \\ 000100000\ 0001111000 \\ 000010000\ 1100111100 \\ 000001000\ 0110011110 \\ 000000100\ 0000001111 \\ 000000010\ 1010101010 \\ 000000001\ 0101010111 \end{bmatrix}_{19\times 19}$$

오류벡터는 $[E]=T_{aug}^{-1}\begin{bmatrix} S \\ S_{aug} \end{bmatrix}$ 이며 오류벡터를 구하기 위해서 먼저 신드롬 S를 구한다.

$$S = TM'^{T} \oplus CB'^{T} = \begin{bmatrix} 1\\1\\0\\1\\1\\1\\0\\1\\0 \end{bmatrix} \oplus \begin{bmatrix} 0\\0\\0\\0\\1\\1\\0\\1\\1 \end{bmatrix} = \begin{bmatrix} 1\\1\\0\\1\\0\\0\\0\\0\\1 \end{bmatrix}_{9\times 1}$$

S가 구해지면 복호화 회로에서 S_{aug}는 PLA에 의해서 자동으로 대응되도록 구성한다.

$$S_{aug} = \begin{bmatrix} 0\\0\\0\\1\\0\\0\\0\\0\\0\\0 \end{bmatrix}_{10\times 1}$$

오류벡터는

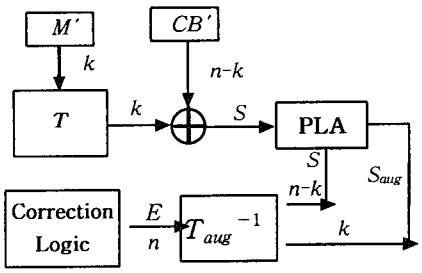
$$[E] = T_{aug}^{-1}\begin{bmatrix} S \\ S_{aug} \end{bmatrix}$$

$$= \begin{bmatrix} 000000000\ 1000000000 \\ 000000000\ 0100000000 \\ 000000000\ 0010000000 \\ 000000000\ 0001000000 \\ 000000000\ 0000100000 \\ 000000000\ 0000010000 \\ 000000000\ 0000001000 \\ 000000000\ 0000000100 \\ 000000000\ 0000000010 \\ 000000000\ 0000000001 \\ 100000000\ 1111000001 \\ 010000000\ 0111100000 \\ 001000000\ 1011110101 \\ 000100000\ 0001111000 \\ 000010000\ 1100111100 \\ 000001000\ 0110011110 \\ 000000100\ 0000001111 \\ 000000010\ 1010101010 \\ 000000001\ 0101010111 \end{bmatrix} \begin{bmatrix} 1\\1\\0\\1\\0\\0\\0\\0\\1\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0 \end{bmatrix} = \begin{bmatrix} 0\\0\\0\\1\\0\\0\\0\\0\\0\\0\\0\\0\\1\\0\\0\\0\\0\\0\\0 \end{bmatrix}$$

에 의해서 $E=(0001000000001000000)$임을 알 수 있다.

따라서 $C=C'\oplus E=(10101010100011011)$이 된다. □

<그림 2>는 이중 오류정정부호의 복호화 방법을 나타낸다.



<그림 2> 복호화

4. 결론

본 논문에서는 rule이 <102, 102, …, 102, 150>인 PBCA의 특성행렬을 이용하여 이중 오류를 정정하기 위하여 최소거리가 5인 부호를 생성하는 방법을 제안하였다. 또한 이중 오류를 효율적으로 복호화하는 방법을 제안하였다. 본 연구에서 제안된 방법은 기존의 연구에서 제안된 방법보다 뛰어남을 보였다. 앞으로 최소거리가 3 이상인 오류정정부호에 대한 연구가 필요할 것이며, 또한 이 연구를 바탕으로 다중 오류정정부호의 설계에 대한 일반화 연구가 필요할 것으로 사료된다.

참고문헌

[1] P. P. Chaudhuri, et. al., "Additive cellular automata theory and applications", Vol. 1, IEEE Computer Society Press, California, USA, 1997.

[2] T.R.N. Rao, E. Fujiwara, "Error-Control Coding for Computer System", Prentice Hall, Englewood Cliffs, N.J., 1989.

[3] S. Lin, D.J. Costello, "Error Control Coding: Fundamentals and Applicaitons", rentice Hall, Englewood Cliffs, N.J., 1983.

[4] D.R. Chowdhury, S. Basu, I.S. Gupta, P.P. Chaudhuri, "Design of CAECC - Cellular Automata Based Error Correcting Code", IEEE Trans. Computers, Vol. 43, June 1994, pp. 759-764.

[5] S.J. Cho, U.S. Choi and H.D. Kim, "Linear nongroup one-dimensional cellular automata characterization on GF(2)", J. Korea Multimedia Soc., Vol. 4, No. 1, 2001, pp. 91-95.

[6] S.J. Cho, H.D. Kim and U.S. Choi, "Behavior of Complemented CA Whose Complement Vector is Acyclic in a Linear TPMACA", Mathematical and Computer Modelling, July, 2002, pp. 980-986.

[7] S.J. Cho, H.D. Kim and U.S. Choi, "Analysis of complemented CA derived from a linear TPMACA", Computers & Mathematics with Applications, Vol. 45, 2003, pp. 689-698.