論 文

병렬 CRC 코드 생성기 및 Syndrome 계산기의 구현

正會員 金 永 燮* 正會員 崔 松 仁* 正會員 朴 弘 植* 正會員 金 在 均**

Implementation of Parallel Cyclic Redundancy Check Code Encoder and Syndrome Calculator

Young Sup Kim*, Song In Choi*, Hong Shik Park*, Jae Kyoon Kim** *Regular Members*

要 約

디지틀 전송 시스템에서 순방향 에러 제어(Forward Error Control) 방식으로 에러를 검출할 수 있는 성능과 구현의 용이함에 의해 Cyclic Redundancy Check(CRC) code가 널리 사용되고 있다. 즉, 간단한 몇개의 shift register와 modulo 2 가산기를 이용하여 회로를 구성하고 입력 데이터 열을 직렬로 입력하면 최종적으로 shift register에 남아 있는 값이 CRC code가 되어 입력 데이터 열을 전송한 뒤 shift register의 값들을 순차적으로 전송하는 방식으로 전송 상의 에러를 검출하고 수정한다. 그러나 전송속도가 높아짐에 따라 직렬 데이터를 이용하여 CRC code를 생성하는 회로를 구현하는 것은 반도체 소자의 속도 제약 때문에 많은 어려움이 따른다. 따라서 본 논문에서는 주문형 반도체 개발시 반도체 소자의 속도 제약 문제를 해소하기 위하여 입력데이터 열을 병렬로 입력하여 직렬로 수행하는 방식과 동일한 방식으로 동작하는 병렬 CRC code 생성방식 및 syndrome 계산방식을 제안하였다.

ABSTRACT

In the digital transmission system, cyclic redundancy check(CRC) code is widely used because it is easy to be implemented and has good performance in error detection. CRC code generator consists of several shift registers and modulo 2 adders. After manipulation of input data stream in the encoder, the remaining value of shift registers becomes CRC code. At the receiving side, error can be detected and corrected by CRC codes immediately transmitted after data stream. But, in the high speed system such as an ATM switch, it is difficult to implement the serial CRC encoder because of speed limitation of available semiconductor devices. In this paper, we propose the efficient parallel CRC encoder and syndrome calculator to solve the speed problem in implementing these functions using the existing semiconductor technology.

*韓國電子通信研究所
ETRI
**韓國科學技術院 電氣및 電子工學科
KAIST
論文番號 : 93-9 (接受1992. 7. 1)

I. 서 론

디지틀 전송 시스템에서 순방향 에러 제어(Forward Error Control) 방식으로 에러를 검출할 수 있

는 성능이 좋고, 구현이 용이한 Cyclic Redundancy Check(CRC) code가 널리 사용되고 있다. 즉, CRC code 생성다항식에 따라 몇개의 shift register와 modulo 2 가산기를 이용하여 회로를 구성하고 입력 데이터 열을 직렬로 입력하면 최종적으로 shift register에 남아 있는 값이 CRC code가 되어 입력 데이터 열을 전송한뒤 shift register의 값들을 순차적으로 전송하는 방식으로 전송 상의 에러를 검출하고 수정할 수 있다[1-3]. 이러한 CRC code를 이용할 경우 생성 다항식에 따라 에러의 검출 확율은 거의 100%에 이르고, 단일 비트 에러인 경우 수신단에서 에러의 정정도 가능하다. 그러나 전송속도가 높아짐에 따라 직렬 데이터를 이용하여 CRC code를 생성하는 회로를 구현하는 것은 반도체 소자의 속도 제약때문에 많은 어려움이 따른다. 예를들어 ATM(Asynchronous Transfer Mode) 망에서는 155.520Mbps 혹은 622.080Mbps의 속도로 정보를 전송하므로 이를 처리할 수 있는 반도체 소자에 의해 회로를 구현하여야 하는데 적용 가능한 반도체 소자에는 한계가 있으며, 있다 하더라도 전력소모의 과다등 많은 문제점이 있다. 반면에 입력 데이터열을 병렬로 변환하여 동작속도를 낮추게 되면 기존의 반도체 소자들을 이용하여 쉽게 구현할 수 있다. 즉, 155.520Mbps의 전송 속도를 8bit 병렬 처리하면 19.44Mbps가 되어 기존의 반도체 소자를 적용할 수 있게 된다.

Pei 등은 최근 참고문헌(4)에서 CRC code 길이가 병렬 데이터 워드 길이보다 길거나 같은 경우의 병렬 CRC code 생성방식을 제안하였으나 본 논문에서는 CRC code와 병렬 입력 데이터 및 CRC code 생성다항식에 대한 일반 관계식을 구함으로써 CRC code 길이가 병렬 데이터 워드 길이보다 적은 경우에도 적용가능한 병렬 CRC code 생성방식을 제안하고, 이 개념을 활용한 병렬 Syndrome 계산기 구현 방식도 함께 제안한다. 아울러 본 방식을 ATM 기반 사용자-망 인터페이스의 물리계층에 적용한 예를 제시한다.

Ⅱ. 수학적 해석

CRC code 생성기는 일반적으로 생성다항식에 따라 다수의 shift register와 modulo 2 가산기로 구성된다. 그림 1에 (n,k) CRC code를 생성하는 일반적인 CRC code 생성기를 나타내었다[1-3]. 여기서 n은 CRC code를 포함한 전송 데이터의 길이를 나타내고, k는 CRC code를 제외한 정보데이터의 길이를 나타낸다.

CRC code 생성기의 각 shift register 상태를 $d_{j,i}$라 하자. 이는 j번째 shift register가 i번 shift를 수행한 뒤의 상태를 나타낸다. 전체 shift register 상태를 (n-k)×1의 상태 벡터로 나타낼 수 있다. 즉,

$$\mathbf{d}_i = (d_{1,i}\, d_{2,i} \cdots d_{j,i} \cdots d_{n-k,i})^T \tag{1}$$

i번 shift를 수행한 뒤의 상태 벡터 $\mathbf{d}_i$는 다음과 같이 표현될 수 있다. 즉,

$$\mathbf{d}_i = \mathbf{T}\mathbf{d}_{i-1} + \mathbf{U}a_k \tag{2}$$

여기서 T는 다음과 같은 형태의 (n-k)×(n-k) 상태천이 행렬을 나타내며 U는 (n-k)×1의 행렬이다.

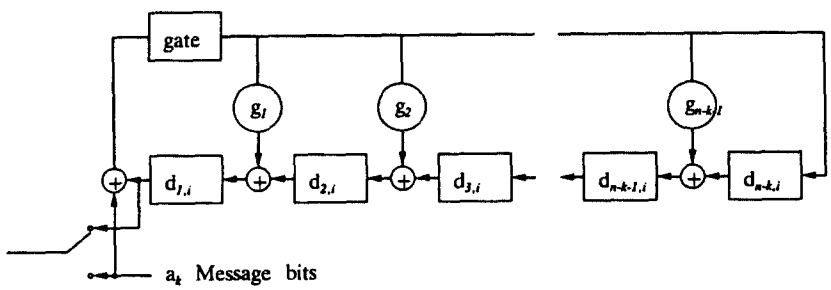


그림 1. (n,k) CRC code 생성기
Fig. 1. (n,k) CRC code encoder

$$\mathbf{T}=\begin{bmatrix} g_1 & 1\ 0 & \cdots\cdots & 0\ 0 \\ g_2 & 0\ 1 & \cdots\cdots & 0\ 0 \\ g_3 & 0\ 0 & \cdots\cdots & 0\ 0 \\ g_4 & 0\ 0 & \cdots\cdots & 0\ 0 \\ \cdot & \cdot\ \cdot & \cdots\cdots & \cdot\ \cdot \\ \cdot & \cdot\ \cdot & \cdots\cdots & \cdot\ \cdot \\ g_{n-k-1} & 0\ 0 & \cdots\cdots & 0\ 1 \\ 1 & 0\ 0 & \cdots\cdots & 0\ 0 \end{bmatrix}, \quad \mathbf{U}=\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ \cdot \\ \cdot \\ g_{n-k-1} \\ 1 \end{bmatrix}$$

a_k는 입력 데이터 열이고, g_i는 0 혹은 1이다.

CRC를 생성하기 위한 $\mathbf{d}_0$는 **(0)**이며 $\mathbf{d}_k$가 입력 데이터 $a_1a_2\ldots a_k$에 대한 CRC 값이된다. 즉,

$$\begin{aligned}\mathbf{d}_k &= \mathbf{T}\mathbf{D}_{k-1}+\mathbf{U}a_k \\ &= \mathbf{T}^2\mathbf{d}_{k-2}+\mathbf{T}\mathbf{U}a_{k-1}+\mathbf{U}a_k \\ &= \mathbf{T}^3\mathbf{d}_{k-3}+\mathbf{T}^2\mathbf{U}a_{k-2}+\mathbf{T}\mathbf{U}a_{k-1}+\mathbf{U}a_k \\ &\cdots\cdots\cdots\cdots \\ &= \mathbf{T}^k\mathbf{d}_0+\mathbf{T}^{k-1}\mathbf{U}a_1+\mathbf{T}^{k-2}\mathbf{U}a_2+\ldots+\mathbf{T}^2\mathbf{U}a_{k-2} \\ &\quad +\mathbf{T}\mathbf{U}a_{k-1}+\mathbf{U}a_k \\ &= \mathbf{G}\mathbf{a} \end{aligned} \tag{3}$$

로 된다. 여기서,

$$\mathbf{G}=(\ \mathbf{T}^{k-1}\mathbf{U}\quad \mathbf{T}^{k-2}\mathbf{U}\quad \ldots\quad \mathbf{T}^1\mathbf{U}\quad \mathbf{U}\ )$$

인 $(n-k)\times k$인 행렬이고 a는 입력 데이터 열로 이루어진 $k\times 1$ 입력 벡터이다. 즉,

$$\mathbf{a}=(\ a_1\ a_2\ \ldots\ a_{k-1}\ a_k\ )^T$$

이를 m bit의 길이를 갖는 데이터 워드로 병렬처리하기 위해서는 k개의 메모리에 입력데이터를 저장하여 실제로 CRC 생성규칙에 따라 나눗셈을 수행하여 CRC code를 얻는 방법과 식(3)에 의해 modulo 2 가산을 수행하는 방법이 있다. 이 두 경우에 입력 데이터 열을 k개 만큼 저장하고 k개의 입력이 들어온 순간 복잡한 연산에 의해 CRC가 생성된다.

예를들어 생성다항식이 x^4+x+1인 (16,12)의 CRC를 4bit의 데이터 워드로 생성하는 경우를 고려하여 본다. 이 경우 **T** 및 **U**는 다음과 같다.

$$\mathbf{T}=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{U}=\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

이를 이용하여 G 행렬을 구하면 다음과 같다.

$$\mathbf{G}=\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

이를 하드웨어로 구현하려면 입력되는 12bit의 데이터를 저장하기 위한 12개의 메모리와 25개의 modulo 2 가산기가 필요하다.

수신단에서는 CRC code를 포함한 n개의 입력데이터 열을 수신하여 syndrome 계산기에 의해 CRC code 생성 다항식으로 나눗셈을 수행하여 나머지를 얻는다. 송신부에서 전송한 데이터는 CRC code 생성 다항식으로 나누어 나머지가 생기지 않도록 만들어 준 것이므로 계산된 나머지는 0이 되어야 하고, 0이 아닌 경우는 수신된 데이터에 에러가 있음을 나타낸다. 단일 비트 에러인 경우에는 어느 위치에서 에러가 발생하였는지 syndrome을 분석하여 찾아낼 수 있고 에러의 정정이 가능하다. 그림 2에 일반적인 직렬 syndrome 계산기를 나타내었다. 이를 CRC code 생성기를 분석한 방법과 동일하게 분석하여 본다.

먼저 syndrome 계산기의 shift register 상태를 $r_{j,i}$로 나타내면 다음과 같은 $(n-k)\times 1$의 상태 벡터로 shift register의 상태를 표시할 수 있다. 즉,

$$\mathbf{r}_i=(\ r_{1,i}\ r_{2,i}\ \cdots\ r_{j,i}\ \cdots\ r_{n-k,i}\ )^T \tag{4}$$

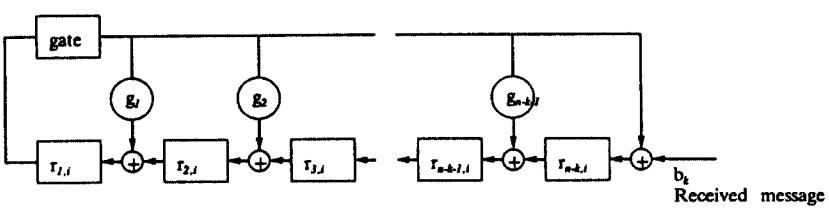


그림 2. (n,k) Syndrome 계산기
Fig. 2. (n,k) syndrome caculator

i번 shift 뒤의 상태 벡터 r_i는 다음과 같은 상태 방정식으로 나타낼 수 있다. 즉,

$$\mathbf{r}_i = \mathbf{T}\mathbf{r}_{i-1} + \mathbf{V}b_k \tag{5}$$

여기서 **T**는 식(2)의 **T**와 동일한 형태이고, **V**는 다음과 같은 형태의 (n−k)×1의 행렬이다.

$$\mathbf{V} = (\ 0\ 0\ 0\ \ldots\ 0\ 1\ )^T$$

b_k는 수신된 입력 데이터 열이다.

syndrome을 계산하기 위한 $\mathbf{r}_0$는 **(0)**이고 $\mathbf{r}_n$이 입력 데이터열 $b_1 b_2 b_3 \ldots b_n$에 대한 syndrome이 된다.

즉,

$$\begin{aligned}\mathbf{r}_n &= \mathbf{T}\mathbf{d}_{n-1} + \mathbf{V}b_n \\ &= \mathbf{T}^2\mathbf{r}_{n-2} + \mathbf{T}\mathbf{U}b_{n-1} + \mathbf{V}b_n \\ &= \mathbf{T}^3\mathbf{r}_{n-3} + \mathbf{T}^2\mathbf{V}b_{n-2} + \mathbf{T}\mathbf{U}b_{n-1} + \mathbf{V}b_n \\ &\cdots\cdots\cdots\cdots \\ &= \mathbf{T}^n\mathbf{r}_0 + \mathbf{T}^{n-1}\mathbf{V}b_1 + \mathbf{T}^{n-2}\mathbf{V}b_2 + \ldots + \mathbf{T}^2\mathbf{V}b_{n-2} \\ &\quad + \mathbf{T}\mathbf{U}b_{n-1} + \mathbf{V}b_n \\ &= \mathbf{H}\mathbf{b}\end{aligned} \tag{6}$$

이다. 여기서,

$$\mathbf{H} = (\ \mathbf{T}^{n-1}\mathbf{V}\quad \mathbf{T}^{n-2}\mathbf{V}\quad \ldots\quad \mathbf{T}\mathbf{V}\quad \mathbf{V}\ )$$

인 (n−k)×n인 행렬이고 **b**는 입력 데이터 열로 이루어진 n×1 입력 벡터이다. 즉,

$$\mathbf{b} = (\ b_1\ b_2\ \ldots\ b_{n-1}\ b_n\ )^T$$

이를 m bit의 길이를 갖는 데이터 워드로 병렬 처리하기 위해서는 n개의 메모리에 입력데이터를 저장하여 복잡한 연산에 의해 syndrome을 계산하여야 한다.

앞에 설명한 예를 적용하여 syndrome을 계산할 경우의 **H** 행렬을 구하면 다음과 같다.

$$\mathbf{H} = \begin{bmatrix} 0&1&1&1&1&0&1&0&1&1&0&0&1&0&0&0 \\ 0&0&1&1&1&1&0&1&0&1&1&0&0&1&0&0 \\ 0&0&0&1&1&1&1&0&1&0&1&1&0&0&1&0 \\ 1&1&1&1&0&1&0&1&1&0&0&1&0&0&0&1 \end{bmatrix}$$

이를 구현하려면 입력되는 16bit의 데이터를 저장하기 위한 16개의 메모리와 29개의 modulo 2 가산기가 필요하다.

Ⅲ. CRC Code 생성기의 병렬구현

본 논문에서는 메모리 및 modulo 2 가산기 수를 줄이기 위한 병렬 CRC code 생성기를 구현하고자 한다. 입력데이터 열이 m의 길이를 갖는 병렬 데이터 워드로 입력된다면 입력 데이터 열은 다음과 같은 입력 벡터로 표시할 수 있다. 즉,

$$\mathbf{c}_i = (\ c_{1,i}\ c_{2,i}\ \cdots\ c_{m,i}\ )^T$$

여기서 입력 데이터 열의 길이 k가 데이터 워드의 길이 m의 배수라면, 식(2)로부터

$$\begin{aligned}\mathbf{d}_k &= \mathbf{T}^m\mathbf{d}_{k-m} + \mathbf{T}^{m-1}\mathbf{U}c_{1,l} + \mathbf{T}^{m-2}\mathbf{U}c_{2,l} + \ldots \\ &\quad + \mathbf{T}\mathbf{U}c_{m-1,l} + \mathbf{U}c_{m,l} \\ &= \mathbf{T}^m\mathbf{d}_{k-m} + \mathbf{P}\mathbf{c}_l\end{aligned} \tag{7}$$

이다. 여기서 l은 입력 데이터 열의 수 k를 m으로 나눈 몫이고 **P**는 다음과 같은 형태의 (n−k)×m 행렬이다.

$$\begin{aligned}\mathbf{P} &= (\ \mathbf{T}^{m-1}\mathbf{U}\quad \mathbf{T}^{m-2}\mathbf{U}\quad \ldots\quad \mathbf{T}\mathbf{U}\quad \mathbf{U}\ ) \\ &= (\ \mathbf{P}_1\ \mathbf{P}_2\ \ldots\ \mathbf{P}_i\ \mathbf{P}_m\ )\end{aligned}$$

여기서 $\mathbf{P}_i$는 행렬 **P**의 i번째 열로 이루어진 열행렬이다.

식(7)을 이용하여 m의 크기를 갖는 병렬데이터가 입력되면 입력된 데이터를 P에 따라 modulo 2 가산을 수행하여 CRC register에 저장하고 다음 데이터가 입력되면 입력데이터의 modulo 2 가산 결과와 CRC register 값을 이용하여 연산을 수행하게 되면 CRC code의 데이터 워드 크기의 메모리와 식(3)을 이용한 방식보다 적은 수의 modulo 2 가산기를 이용하여 병렬 CRC code 생성회로를 구성할 수 있음을 알 수 있다.

(정리 1)병렬 입력 데이터 워드의 길이 m이 CRC code의 데이터 워드 n−k보다 작고 입력데이터 열의 길이 k가 m의 배수이면, 상태 벡터 $\mathbf{d}_k$는 다음과 같다.

$$\mathbf{d}_k = \mathbf{T}^m \left[\frac{\mathbf{D}_{A,k} + \mathbf{c}_l}{\mathbf{D}_{B,k}} \right] \tag{8}$$

여기서,

$$\mathbf{d}_k = \left[\frac{\mathbf{D}_{A,k}}{\mathbf{D}_{B,k}} \right]$$

로서

$$\mathbf{D}_{A,k} = (\, d_{1,k} \; d_{2,k} \; \cdots \; d_{m,k} \,)^T$$
$$\mathbf{D}_{B,k} = (\, d_{m+1,k} \; d_{m+2,k} \; \cdots \; d_{n,k} \,)^T$$

를 나타낸다.

(증명)
행렬 $\mathbf{T}^m$를 다음과 같이 표시하자. 즉,

$$\mathbf{T}^m = (\, \mathbf{T}_1^m \; \mathbf{T}_2^m \; \ldots \; \mathbf{T}_i^m \; \ldots \; \mathbf{T}_{n-k}^m \,)$$
$$= (\, \mathbf{T}_A^m \mid \mathbf{T}_B^m \,)$$

여기서, $\mathbf{T}_i^m$은 행렬 $\mathbf{T}^m$의 i번째 열로 이루어진 열행렬이며, $\mathbf{T}_A^m$은

$$\mathbf{T}_A^m = (\, \mathbf{T}_1^m \; \mathbf{T}_2^m \; \ldots \; \mathbf{T}_m^m \,)$$

인 $(n-k)\times m$ 행렬이고, $\mathbf{T}_B^m$은

$$\mathbf{T}_B^m = (\, \mathbf{T}_{m+1}^m \; \mathbf{T}_{m+2}^m \; \ldots \; \mathbf{T}_{n-k}^m \,)$$

인 $(n-k)\times(n-k-m)$ 행렬이다.

그러면,

$\mathbf{T}_1 = \mathbf{U}$이고,
$\mathbf{T}_1^m = \mathbf{T}^{m-1}\mathbf{T}_1 = \mathbf{T}^{m-1}\mathbf{U}$
$\mathbf{T}_2^m = \mathbf{T}_1^{m-1} = \mathbf{T}^{m-2}\mathbf{T}_1 = \mathbf{T}^{m-2}\mathbf{U}$
. . . . . .
$\mathbf{T}_{m-1}^m = \mathbf{T}_{m-2}^{m-1} = \mathbf{T}_{m-3}^{m-2} = \ldots = \mathbf{T}_3^2 = \mathbf{T}_2 = \mathbf{T}\mathbf{T}_1 = \mathbf{T}\mathbf{U}$
$\mathbf{T}_m^m = \mathbf{T}_{m-1}^{m-1} = \mathbf{T}_{m-2}^{m-2} = \ldots = \mathbf{T}_2^2 = \mathbf{T}_1 = \mathbf{U}$

이다. 따라서

$$\mathbf{P} = (\, \mathbf{T}_1^m \; \mathbf{T}_2^m \; \mathbf{T}_3^m \; \ldots \; \mathbf{T}_m^m \,) = \mathbf{T}_A^m$$

이므로 식(7)로부터

$$\mathbf{d}_k = (\, \mathbf{T}_A^m \mid \mathbf{T}_B^m \,) \left[\frac{\mathbf{D}_{A,k}}{\mathbf{D}_{B,k}} \right] + \mathbf{T}_A^m \mathbf{c}_l$$
$$= (\, \mathbf{T}_A^m \mid \mathbf{T}_B^m \,) \left[\frac{\mathbf{D}_{A,k} + \mathbf{c}_l}{\mathbf{D}_{B,k}} \right]$$
$$= \mathbf{T}^m \left[\frac{\mathbf{D}_{A,k} + \mathbf{c}_l}{\mathbf{D}_{B,k}} \right]$$

(정리 2)병렬 입력 데이터 워드의 길이 m이 CRC code의 데이터 워드 n−k와 같고, 입력 데이터 열의 길이 k가 m의 배수이면,

$$\mathbf{P} = \mathbf{T}^m$$

이고, 상태 벡터 $\mathbf{d}_k$은 다음과 같다.

$$\mathbf{d}_k = \mathbf{T}^m(\mathbf{d}_{k-m} + \mathbf{c}_l) \qquad (9)$$

(증명)
정리1에서와 동일한 방법으로

$\mathbf{P}_1 = \mathbf{T}^{m-1}\mathbf{U} = \mathbf{T}_1^m$
$\mathbf{P}_2 = \mathbf{T}^{m-2}\mathbf{U} = \mathbf{T}_2^m$
. . . . . .
$\mathbf{P}_{m-1} = \mathbf{T}\mathbf{U} = \mathbf{T}_{m-1}^m$
$\mathbf{P}_m = \mathbf{U} = \mathbf{T}_m^m$

임을 쉽게 보일 수 있다. 따라서, $\mathbf{T}^m = \mathbf{H}$ 이고, 식(9)는 성립한다.

예를 들어 입력 데이터 워드의 길이 m이 4인 경우 $\mathbf{T}^4$ 및 $\mathbf{P}$를 구해보면,

$$\mathbf{T} = \begin{bmatrix} g_1 & 1 & 0 & 0 \\ g_2 & 0 & 1 & 0 \\ g_3 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{T}\mathbf{U} = \begin{bmatrix} g_1+g_2 \\ g_1g_2+g_3 \\ g_1g_3+1 \\ g_1 \end{bmatrix}$$

$$\mathbf{T}^2 = \begin{bmatrix} g_1+g_2 & g_1 & 1 & 0 \\ g_1g_2+g_3 & g_2 & 0 & 1 \\ g_1g_3+1 & g_3 & 0 & 0 \\ g_1 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{T}^2\mathbf{U} = \begin{bmatrix} g_1+g_3 \\ g_1g_2+g_1g_3+g_2+1 \\ g_1+g_1g_3+g_2g_3 \\ g_1+g_2 \end{bmatrix}$$

$$\mathbf{T}^3 = \begin{bmatrix} g_1+g_3 & g_1+g_2 & g_1 & 1 \\ g_1g_2+g_1g_3+g_2+1 & g_1g_2+g_3 & g_2 & 0 \\ g_1+g_1g_3+g_2+g_3 & g_1g_3+1 & g_3 & 0 \\ g_1+g_2 & g_1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{T}^3\mathbf{U} = \begin{bmatrix} g_1+g_1g_2+g_2+1 \\ g_1+g_1g_2+g_1g_3 \\ g_1+g_2+g_3+g_1g_3 \\ g_1+g_3 \end{bmatrix}$$

$$\mathbf{T}^4=\begin{bmatrix} g_1+g_1g_2+g_2+1 & g_1+g_3 & g_1+g_2 & g_1 \\ g_1+g_1g_2+g_1g_3 & g_1g_2+g_1g_3+g_2+1 & g_1g_2+g_3 & g_2 \\ g_1+g_2+g_3+g_1g_3 & g_1+g_1g_2+g_2g_3 & g_1g_3+1 & g_3 \\ g_1+g_3 & g_1+g_2 & g_1 & 1 \end{bmatrix}$$

그러므로 $\mathbf{T}^4=\mathbf{P}$이다.

식(9)에 의하여 병렬 CRC code 생성기를 구현하면 단지 데이터 워드 크기의 메모리와 적은 수의 modulo 2 가산기로 구현할 수 있음을 알 수 있다.

앞에서 설명한 예에서 입력데이터 열이 4bit 길이의 데이터 워드로 입력된다면 식(9)로 부터

$$\mathbf{d}_k=\mathbf{T}^4(\mathbf{d}_{k-4}+\mathbf{c}_l)$$

$$=\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}(\mathbf{d}_{k-4}+\mathbf{c}_l)$$

이 되며, 이를 이용한 CRC code 생성 회로는 그림3과 같다. 그림3에 나타낸 바와 같이 4bit의 메모리와 9개의 modulo 2 가산기로 병렬 CRC 생성기를 구현할 수 있다.

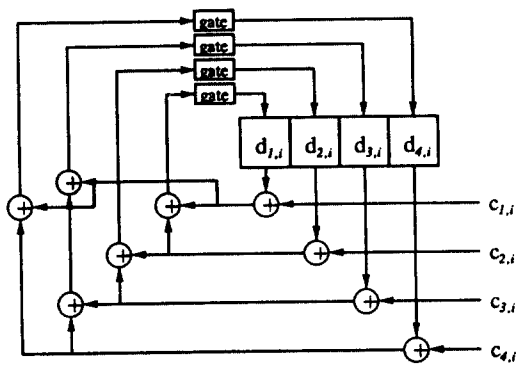


그림 3. 4bit 병렬 입력 (16,12) CRC code 생성회로
Fig. 3. 4bit parallel input (16,12) CRC code encoder

(정리 3)병렬 입력 데이터 워드의 길이 m이 CRC code의 데이터 워드 n-k보다 크고, 입력 데이터 열의 길이 k가 m의 배수이면, 상태 벡터 $\mathbf{d}_k$는 다음과 같다.

$$\mathbf{d}_k=\mathbf{P}\begin{bmatrix}\mathbf{d}_{k-m}+\mathbf{C}_{A,l}\\ \mathbf{C}_{B,l}\end{bmatrix} \tag{10}$$

여기서 입력벡터 c_l은,

$$\mathbf{c}_l=\begin{bmatrix}\mathbf{C}_{A,l}\\ \mathbf{C}_{B,l}\end{bmatrix}$$

로서

$$\mathbf{C}_{A,l}=(\ c_{1,l}\ c_{2,l}\ \cdots\ c_{n-k,l}\ )^T$$
$$\mathbf{C}_{B,l}=(\ c_{n-k+1,l}\ c_{n-k+2,l}\ \cdots\ c_{m}\ )^T$$

를 나타낸다.

(증명)
정리 1과 2에서 증명한 방법과 동일한 방법으로

$$\mathbf{T}^m=(\ \mathbf{P}_1\ \mathbf{P}_2\ \ldots\ \mathbf{P}_i\ \ldots\ \mathbf{P}_{n-k}\ )$$

임을 알 수 있다. 따라서 식(10)은 성립한다.

Ⅳ. Syndrome 계산기의 병렬구현

병렬 syndrome 계산기에 대해서도 CRC 생성기와 동일한 방법으로 구현할 수 있다. 수신단에서 수신한 입력 데이터 열이 m bit의 길이를 갖는 병렬 데이터 워드로 입력된다면 입력 데이터열을 다음과 같은 입력 벡터로 표시할 수 있다. 즉,

$$\mathbf{s}_i=(\ s_{1,i}\ s_{2,i}\ \ldots\ s_{m,i}\ )^T$$

여기서 수신 데이터열의 길이 n이 데이터 워드의 길이 m의 배수라면 식(5)로 부터 다음과 같은 관계식을 얻을 수 있다.

$$\begin{aligned}\mathbf{r}_n&=\mathbf{T}^m\mathbf{r}_{n-m}+\mathbf{T}^{m-1}\mathbf{V}s_{1,l}+\mathbf{T}^{m-2}\mathbf{V}s_{2,l}+\ldots\\&\quad+\mathbf{T}\mathbf{V}s_{m-1,l}+\mathbf{V}s_{m,l}\\&=\mathbf{T}^m\mathbf{r}_{n-m}+\mathbf{Q}\mathbf{s}_l\end{aligned} \tag{11}$$

여기서 $\mathbf{Q}$는 다음과 같은 형태의 (n-k)×m 행렬이다.

$$\mathbf{Q}=(\ \mathbf{T}^{m-1}\ \mathbf{T}^{m-2}\mathbf{V}\ \ldots\ \mathbf{T}\mathbf{V}\ \mathbf{V}\ )$$

식(11)을 통해 m의 크기를 갖는 병렬 데이터가 입력되면 입력된 데이터를 $\mathbf{Q}$에 따라 modulo 2 가산을 수행하여 syndrome register에 저장하고 다음 데이터가 입력되면 다시 입력 데이터와 syndrome regis-

ter의 값을 이용하여 연산을 수행하는 방식으로 syndrome을 계산한다. 이와 같이 하면 식(6)을 이용한 방법보다 적은 수의 메모리와 modulo 2 가산기로 syndrome 계산기를 구현할 수 있다.

앞에서 설명한 예에서 입력 데이터 열이 4bit 길이의 데이터 워드로 입력된다면 식(11)로 부터

$$\mathbf{r}_n = \mathbf{T}^4 \mathbf{r}_{n-4} + \mathbf{Q}\mathbf{s}_l$$

$$= \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \mathbf{r}_{n-4} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{s}_l$$

이 된다. 이를 이용한 병렬 syndrome 계산기 회로를 그림4에 나타내었다. 그림4에 나타낸 바와 같이 4bit의 메모리와 9개의 modulo 2 가산기를 이용하여 병렬 syndrome 계산기를 구현할 수 있다.

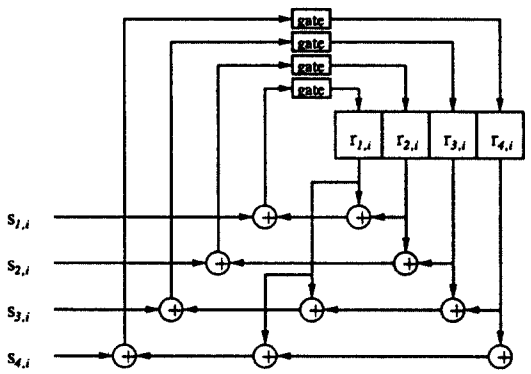


그림 4. 4bit 병렬 입력 (16,12) Syndrome 계산기
Fig. 4. 4bit parallel input (16,12) syndrome caculator

V. ATM 물리 계층에 적용

ATM 가입자 정합부에서는 4 octet의 cell 헤더에 대해 x^8+x^2+x+1의 다항식을 이용하여 (40,32)의 CRC를 계산하여 다섯번째 octet으로 사용하도록 권고하고 있으며 전송 속도를 155.520Mbps 혹은 622.080Mbps로 규정하고 있다[5]. 이러한 전송속도의 직렬데이터를 기존의 반도체 소자로 처리하기는 어렵다. 이를 8 bit 크기를 갖는 데이터 워드로 병렬처리를 수행하면 155.520Mbps의 경우 19.44Mbps로 동작 속도를 낮출 수 있으며 식(3)을 이용하여 병렬 CRC code 생성기를 구현하면 32개의 메모리와 약 110개의 modulo 2 가산기가 필요하다. 한편, 식(9)를 이용한 병렬 CRC 생성기를 구현해 보면 **T**, **U**, **P**는 다음과 같게 되고,

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

이를 이용한 8bit 병렬 CRC 생성기는 그림5와 같게 된다. ATM cell header의 HEC는 구해진 CRC에 B'01010101을 더하여 전송한다.

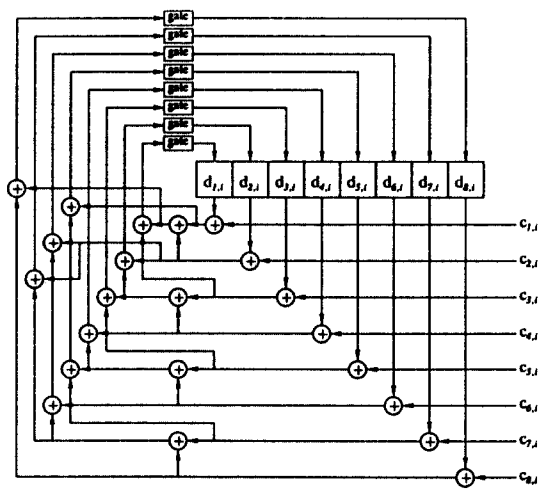


그림 5. ATM cell header의 병렬 CRC code 생성회로
Fig. 5. Parallel CRC code encoder of ATM cell header

그림5에 나타낸 바와 같이 식(9)을 이용하여 병렬 CRC code 생성기를 구현하면 8bit의 메모리와 22개

의 modulo 2 가산기로 구성할 수 있어 식(3)에 의한 병렬 CRC code 생성기와 비교할 때 하드웨어를 현저히 감소시킬 수 있음을 알 수 있다.

또한, ATM cell 동기를 찾거나 header의 에러를 제어하기 위한 syndrome 계산기는 그림6에 나타내었다.

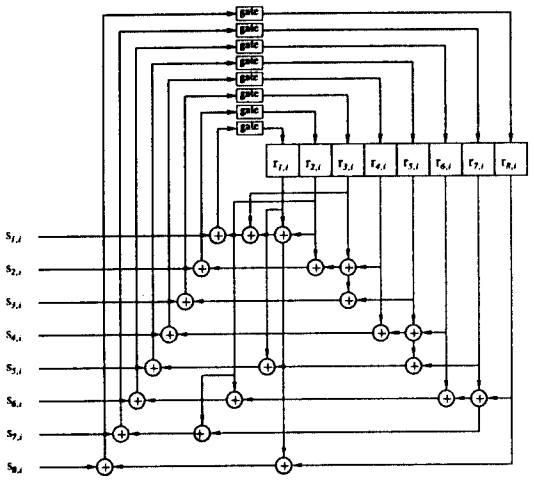


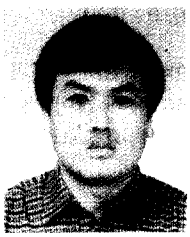
그림 6. ATM cell header의 syndrome 계산기
Fig. 6. Syndrome calculator of ATM cell header

Ⅵ. 결 론

본 논문에서는 수학적 해석을 통하여 병렬 CRC code 생성기 및 병렬 syndrome 계산기를 간단하게 구현할 수 있는 방법을 제안하고 이를 ATM 물리계층의 HEC code를 생성하기 위한 CRC code 생성회로와 syndrome 계산기에 적용하여 보았다. 이제까지 병렬로 CRC code를 생성하거나 syndrome을 계산하려면 병렬로 입력된 데이터를 순차적으로 메모리에 저장하여 나눗셈을 수행하거나 생성행렬에 의해 복잡한 연산을 통해 이루어 졌으나 제안된 방식으로 병렬 CRC code 생성기와 syndrome 계산기를 구현하면 CRC code의 크기 만큼의 메모리와 적은 수의 가산기를 이용해 구현할 수 있다.

참 고 문 헌

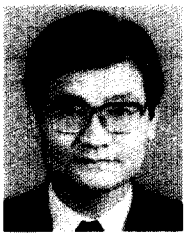
1. Simon Haykin, *Digital Communications*, John Wiley & Sons, 1988.
2. Shu Lin and Daniel J. Costello Jr., *Error Control Coding*, Prentice Hall, 1983.
3. T. R. N. Rao and E. Fujiwara, *Error Control Coding for Computer System*, Prentice Hall, 1989.
4. Tong-Bi Pei and Chales Zukowski, "High-Speed Parallel CRC Circuits in VLSI," *IEEE Trans. on Comm.*, Vol.40, No.4, April, 1992.
5. CCITT Recommendation I.432, Dec. 1991.



金 永 燮(Young Sup Kim) 정회원
1961年 5月 21日生
1984年 2月 : 홍익대학교 전자공학과(학사)
1988年 8月 : 홍익대학교 대학원 전자공학과(석사)
1987年 2月 ~ 현재 : 한국전자통신 연구소 ATM정합연구실

崔 松 仁(Song In Choi) 정회원
1957年 1月 26日生
1982年 2月 : 광운대 응용전자공학과(학사)
1987年 2月 : 광운대 대학원 전자계산기공학과(석사)
1982年 7月 ~ 현재 : 한국전자통신 연구소 ATM정합연구실

朴 弘 植(Hong Shik Park) 정회원
1953年 8月 16日生
1977年 2月 : 서울대학교 졸업(학사)
1986年 8月 : 한국과학기술원(석사)
1977年 3月 ~ 현재 : 한국전자통신 연구소 ATM 정합연구실 실장



金 在 均(Jae Kyoon Kim) 正會員
1938年 9月 17日生
1958年 4月～1962年 2月 : 韓國航空大學 應用電子科(工學士)
1962年 3月～1967年 2月 : 서울大學校 大學院 電子工學科(工學碩士)
1967年 9月～1971年 8月 : 美國 남가주 大學校 大學院 電氣工學科(工學博士)
1962年 4月～1966年 7月 : 空軍服務 中尉
1967年 9月～1972年 3月 : 美國 남가주 大學校(研究院)
1972年 4月～1973年 3月 : 美國 NASA GSFC(研究院)
1973年 4月～現在 : 韓國科學技術院 教授
1984年 1月～1985年 6月 : 科學技術處 電氣·電子조정관
1993年 現在 : 本學會 會長