論 文

Comparison of Parallel CRC Verification Algorithms for ATM Cell Delineation

Youn Hee Choi*, Sang Seob Song* *Regular Members*

ATM 셀 경계식별을 위한 병렬 CRC 검증 알고리즘의 비교

正會員 崔 允 姬* 正會員 宋 祥 燮*

ABSTRACT

In this paper we discuss three algorithms-*Direct, Successive*, and *Recursive*-on parallel CRC(Cyclic Redundancy Check) verification. The algorithms are derived by combining the byte-syndromes precomputed from the generator polynomial. These algorithms are compared in terms of the amount of hardware and the speed of operation. Since the algorithms can be generalized easily, we took the ATM cell delineation example for easier description.

As an application of the algorithm *Recursive*, an ATM cell delineation module suitable for STM-1 transmission has been successfully realized through commercially available field programmable gate arrays.

요 약

본 논문에서는 병렬적으로 CRC(Cyclic Redundancy Check)를 검증할 수 있는 세가지의 알고리즘-*Direct, Successive* 및 *Recursive*-를 연구하였다. CRC 검증에 필요한 각 알고리즘의 신드롬은 CRC 생성에 사용된 생성다항식으로부터 미리 계산된 보조 신드롬들을 조합함으로써 구해진다. 위 세가지 알고리즘은 구현에 필요한 하드웨어의 양과 전송지연에 따른 동작속도 관점에서 비교되었다. 한편 본 논문의 알고리즘들은 CRC 계산에 관여하는 데이타 바이트 수의 함수형태로 표현될 수 있으므로 이해를 돕기 위해 ATM 셀 경계식별 알고리즘을 예로 들어 설명하였다. 세가지 알고리즘 중 가장 구현이 간단한 *Recursive* 방법을 이용하여 STM-1급 전송에 적합한 ATM 셀 경계식별 알고리즘을 개발하고 이를 상용 FPGA로 실제 구현하였다.

* 全北大學校 電子工學科
Dept. of Electronics Engineering, Chonbuk National University
論文番號 : 93 167

I. Introduction

With the advent of B-ISDN(Broadband Inte-

grated Services Digital Netwrok), the forward error correction(FEC) schemes, which were restricted to specific applications such as deep space communications, high-speed computer memory systems, or storage devices, began to be employed in ordinary data transmission networks. In ATM network, for example, various FEC schemes are used at different layers. The ATM cell delineation at transmission and convergence(TC) sublayer in physical layer(PL) depends on the HEC(Header Error Control) field generated by a (40,32) sho rtened cyclic code. The protection of SN(Sequence Number) field is achieved by a (7,4) Hamming code at ATM adaptation layer(AAL)[1][2]. In addition, the mandatory use of a Reed-Solomon code has been confirmed to protect video and high-quality audio data for the type 1 AAL services [3]. The algorithms presented in this paper can be generalized to the case involving more than five octets in CRC computation. Therefore we take the ATM cell delineation example for easier description.

According to CCITT I-series documentation, the major functions of the transmission and convergence(TC) sublayer in physical layer(PL) include the ATM cell delineation whose mechanism depends on the HEC(Header Error Control) field of ATM cell header. Specifically, the transmitter computes the CRC corresponding to the first four octets of a header and puts the CRC into the fifth octet called the HEC. The CRC is generated by a (40,32) shortened cyclic code whose generator polynomial is specified as $g(x)=x^8+x^2+x+1$. With the CRC octet inserted, the whole header will be divisible by the generator polynomial. That is, the remainder of the header on being divided by $g(x)$ is zero. Checking whether this property is observed, the receiver determines the ATM cell boundary from the continuously arriving byte streams.

The actual ATM cell delineation mechanism will be controlled by a state diagram shown in Fig 1[2]. In the *HUNT* state, the delineation process is performed by checking the CRC corresponding to an assumed 5-octet header bit by bit or byte by byte. The agreement of the CRC tells the receiver that a valid cell header has been successfully located. For this purpose the receiver first computes the syndrome by dividing the assumed header by $g(x)$. If the syndrome, the remainder on division, is zero, then a CRC agreement has been found and the method enters into the *PRESYNC* state. Once the method leaves the *HUNT* state, the recevier checks the CRC cell by cell. On the other hand, a nonzero syndrome dictates the receiver remain in the *HUNT* state and continue to examine the CRC byte by byte until such an agreement is found.

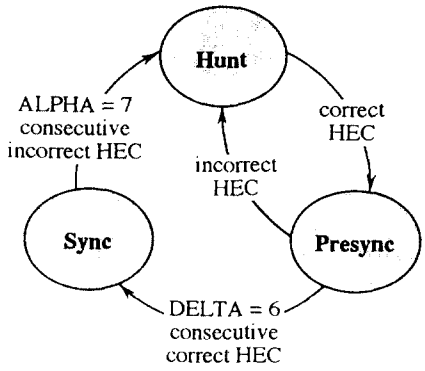


Fig 1. State transition diagram for ATM cell delin eation.

There is an abundant amount of literature on the cell by cell CRC verification algorithms which are either in bit-serial forms[4] or in byte-parallel realizations[5,6]. In this case, essentially the same algorithms can be used for both the CRC gener ation at the transmitter side and the syndrome computation at the reciever side. In the cell virification, the CRC is examined only once for each cell header. The cell header once checked will be discarded regardless of its CRC agree ment. In the byte by byte verification, however, a part of the assumed header should be examined many times. When the CRC agreement is not

found in the *HUNT* state, only the first octet of the assumed header is discarded. A new prospective header, consisting of the last four octets of the assumed header and the new octet following the assumed header, should be examined next time. Therefore the algorithms developed for the cell by cell verification can not be applied directly to this case.

This paper considers the byte by byte CRC verification with octet parallel realizations. The organization of this paper is as follows. In section II the three algorithms are developed. An ATM cell delineation module based on the algorithm *Recursive* will be presented in section III. Conclusion will be given in section IV.

II. Parallel CRC verification algorithms

Before discussing the algorithms, we first compute the *byte-syndromes* corresponding to a single octet B shifted i bits to the left. This is equivalent to multiplying $B=[d_7d_6\cdots d_0]$ by x^i when B is regarded as a binary polynomial of degree 7. Since the order of the specified generator polynomial is 8, the syndrome for $x^nB(n\geq 8)$ becomes a linear combination of the byte-syndromes for B, $xB,\cdots,x^7B$. That is,

$$\begin{aligned}R_{g(x)}[x^nB] &= R_{g(x)}[R_{g(x)}[x^n]\cdot B]\\ &= R_{g(x)}[(a_7x^7+a_6x^6+\cdots+a_0)B]\\ &= a_7R_{g(x)}[x^7B]+a_6R_{g(x)}[x^6B]+\cdots+a_0R_{g(x)}[B]\\ &= \sum_{i=0}^{7} a_iR_{g(x)}[x^iB],\ \text{for some binary constants } a_i\text{'s.}\end{aligned}$$

Here $R_{g(x)}[p(x)]$ is the remainder of a polynomial $p(x)$ when divided by $g(x)$. These byte-syndromes for $x^i\ B(i=0,1,\cdots,7)$ are tabulated in Table 1. Representing a single octet B as the polynomial $B=d_7x^7+d_6x^6+\cdots+d_1x+d_0$, the second column, for instance, has been obtained as

$$\begin{aligned}S(x) &= R_{g(x)}[xB]\\ &= R_{g(x)}[d_7x^8+d_6x^7+\cdots d_1x^2+d_0x]\\ &= d_7(x^2+x+1)+d_6x^7+\cdots d_1x^2+d_0x\\ &= d_6x^7+\cdots+(d_7\oplus d_1)x^2+(d_7\oplus d_0)x+d_7\\ &= s_7x^7+s_6x^6+\cdots+s_1x+s_0\end{aligned}$$

Table 1: Byte-syndromes for x^iB, $i=0,1,\cdots,7$

	B	xB	x^2B	x^3B
s^7	d_7	d_6	d_5	d_4
s^6	d_6	d_5	d_4	d_3
s^5	d_5	d_4	d_3	d_2
s^4	d_4	d_3	d_2	$d_1\oplus d_7$
s^3	d_3	d_2	$d_1\oplus d_7$	$d_0\oplus d_6\oplus d_7$
s^2	d_2	$d_1\oplus d_7$	$d_0\oplus d_6\oplus d_7$	$d_5\oplus d_6\oplus d_7$
s^1	d_1	$d_0\oplus d_7$	$d_6\oplus d_7$	$d_5\oplus d_6$
s^0	d_0	d_7	d_6	d_5
	x^4B	x^5B	x^6B	x^7B
s^7	d_3	d_2	$d_1\oplus d_7$	$d_0\oplus d_6\oplus d_7$
s^6	d_2	$d_1\oplus d_7$	$d_0\oplus d_6\oplus d_7$	$d_5\oplus d_6\oplus d_7$
s^5	$d_1\oplus d_7$	$d_0\oplus d_6\oplus d_7$	$d_5\oplus d_6\oplus d_7$	$d_4\oplus d_5\oplus d_6$
s^4	$d_0\oplus d_6\oplus d_7$	$d_5\oplus d_6\oplus d_7$	$d_4\oplus d_5\oplus d_6$	$d_3\oplus d_4\oplus d_5$
s^3	$d_5\oplus d_6\oplus d_7$	$d_4\oplus d_5\oplus d_6$	$d_3\oplus d_4\oplus d_5$	$d_2\oplus d_3\oplus d_4$
s^2	$d_4\oplus d_5\oplus d_6$	$d_3\oplus d_4\oplus d_5$	$d_2\oplus d_3\oplus d_4$	$d_1\oplus d_2\oplus d_3\oplus d_7$
s^1	$d_4\oplus d_6$	$d_3\oplus d_4$	$d_2\oplus d_3$	$d_1\oplus d_2\oplus d_7$
s^0	d_4	d_3	d_2	$d_1\oplus d_7$

Note that the symbol $\oplus$ denotes the binary exclusive OR(XOR) operation used for the addition of two bits. With the byte-syndromes in Table 1, we are now ready to derive the three algorithms, termed *Direct*, *Successive*, and *Recursive*, respectively.

2.1 Algorithm I : Direct

The first algorithm termed *Direct* determines the syndrome as the sum of 5 subsyndromes, each corresponding to a single octet in the assumed cell header. Let $S(x)$ be the syndrome of an assumed 5-byte header $H=[B_1B_2B_3B_4B_5]$ with the B_1 being arrived first. Note that the notation $[B_1\cdots B_5]$ will be used to denote either the entire 40-bit header or the header polynomial of degree 39, with the MSB d_7 of B_1 as the coefficient of x^{39}. The meaning should be clear from the context.

$$S(x) = R_{g(x)}[H]$$

$$= R_{g(x)}[B_1 B_2 B_3 B_4 B_5]$$
$$= R_{g(x)}[x^{32} B_1] + R_{g(x)}[x^{24} B_2] + \cdots + B_5 \quad (1)$$
$$= S_1(x) + S_2(x) + S_3(x) + S_4(x) + S_5(x)$$

where

$S_i(x) = R_{g(x)}[x^{40-8i} B_i]$, $i = 1,2,\cdots,5$

The ith subsyndrome $S_i(x)$ is computed directly from the ith octet B_i. For example, the fourth subsyndrome $S_4(x)$ due to B_4 is determined as

$$S_4(x) = R_{g(x)}[x^8 B_4]$$
$$= R_{g(x)}[R_{g(x)}[x^8] \cdot B_4]$$
$$= R_{g(x)}[(x^2 + x + 1) \cdot B_4]$$
$$= R_{g(x)}[x^2 B_4] + R_{g(x)}[x B_4] + R_{g(x)}[B_4]$$

The three terms in the above equation have been precomputed Table 1. Therefore the full expression for this can be obtained as just the sum of corresponding columns. That is, summing the byte-syndromes in columns 1, 2, and 3 of Table 1, we have the expression for the subsyndrome $S_4(x)$ directly from the fourth octet B_4.

$$S_4(x) = (d_5 \oplus d_6 \oplus d_7)x^7 + (d_1 \oplus d_5 \oplus d_6)x^6$$
$$+ (d_3 \oplus d_4 \oplus d_5)x^5 + (d_2 \oplus d_3 \oplus d_4)x^4$$
$$+ (d_1 \oplus d_2 \oplus d_3 \oplus d_7)x^3 + (d_0 \oplus d_1 \oplus d_2 \oplus d_6)x^2$$
$$+ (d_0 \oplus d_1 \oplus d_6)x + (d_0 \oplus d_6 \oplus d_7)$$

Note that the XOR operation makes an even number of repeated bits in each term to be cancelled out in combining columns. The equation above can be implemented through a set of XOR network. In this manner, it is immediate to find all the XOR networks required for equation (1) as shown in Table 2. With Table 2, the algorithm *Direct* can be implemented by a 5-byte buffer to store the header, 4 different sets of XOR networks (multipliers), and an adder-another set of XOR network-as shown in Fig 2.

It is obvious from Fig 2 that the amount of hardware required for the algorithm *Direct* increases linearly with the number of octets involved in the CRC computation. In addition, the maximum speed of operation may be limited significantly due to the excessive propagation delay accumulated along the long paths up to the addition circuitry.

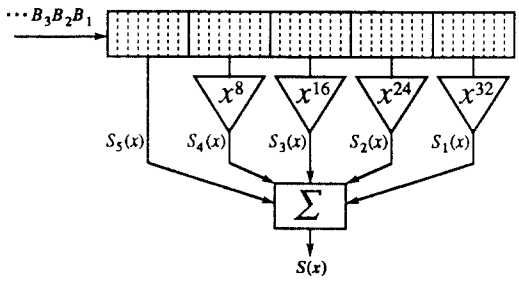


Fig 2. Block diagram of the algorithm *Direct*.

Table 2: XOR networks in algorithm I

	$S_1(x)$ $x^{32}B$	$S_2(x)$ $x^{24}B$	$S_3(x)$ $x^{16}B$	$S_4(x)$ $x^{8}B$	$S_5(x)$ B
s_7	$d_{3,5,6}$	$d_{1,2,4,6}$	$d_{3,5,7}$	$d_{5,6,7}$	d_7
s_6	$d_{2,4,5}$	$d_{0,1,3,5}$	$d_{2,4,6}$	$d_{4,5,6}$	d_6
s_5	$d_{1,4,7}$	$d_{0,2,4}$	$d_{1,3,5,7}$	$d_{3,4,5}$	d_5
s_4	$d_{0,2,3,6,7}$	$d_{1,3}$	$d_{0,2,4,6,7}$	$d_{2,3,4}$	d_4
s_3	$d_{1,2,5,6}$	$d_{0,2,7}$	$d_{1,3,5,6}$	$d_{1,2,3,7}$	d_3
s_2	$d_{0,1,4,5}$	$d_{1,6}$	$d_{0,2,4,5,7}$	$d_{0,1,2,6}$	d_2
s_1	$d_{0,4,5,6}$	$d_{0,1,2,4,5,6,7}$	$d_{1,4,5,6,7}$	$d_{0,1,6}$	d_1
s_0	$d_{4,6,7}$	$d_{0,2,3,5,7}$	$d_{0,4,6}$	$d_{0,6,7}$	d_0
B	B_1	B_2	B_3	B_4	B_5

2.2 Algorithm II : Successive

The second algorithm termed *Successive* is similar to the first algorithm *Direct* except the former uses only one kind of XOR network. That is, the subsyndrome for each octet is successively updated each time it passes the multplier x^8 as shown in Fig 3. To derive this algorithm, the syndrome $S(x)$ of an assumed header $H = [B_1 B_2 B_3 B_4 B_5]$ is rewritten as

$$S(x) = R_{g(x)}[H]$$
$$= R_{g(x)}[B_1 B_2 B_3 B_4 B_5]$$

$$= R_{g(x)}[B_1 OOOO] + R_{g(x)}[B_2 OOO] + R_{g(x)}[B_3 OO] + R_{g(x)}[B_4 O] + R_{g(x)}[B_5]$$
$$= S_1(x) + S_2(x) + S_3(x) + S_4(x) + S_5(x)$$

The symbol O denotes the all-zero byte, *i.e.*, $O = [00000000]$ and the number of O's following an octet equals the number of times that the octet passes the multplier x^8. As depicted in Fig 3, the first octet B_1 will pass the multiplier x^8 four times during the 5 octet cycles. This is equivalent to multiplying B_1 by x^{32}. The second octet B_2 passes x^8 three times to achieve $x^{24}B_2$, and so on. Each time a new octet is received, the subsyndromes are multiplied by x^8 and passed to the next stages. Let $s_i(t)$, $i=0, 1, \cdots, 7$ be the contents of a syndrome register at time t. Denoting T as the byte-clock period, the contents $s_i'(t+T)$, $i=0, 1, \cdots, 7$ of the subsequent syndrome register at time $t+T$ is obtained from $s_i(t)$ by multiplying $s_i(t)$ by x^8. That is,

$$\begin{aligned}
s_0'(t+T) &= s_0(t) \oplus s_6(t) \oplus s_7(t) \\
s_1'(t+T) &= s_0(t) \oplus s_1(t) \oplus s_6(t) \\
s_2'(t+T) &= s_0(t) \oplus s_1(t) \oplus s_2(t) s_6(t) \\
s_3'(t+T) &= s_1(t) \oplus s_2(t) \oplus s_3(t) s_7(t) \\
s_4'(t+T) &= s_2(t) \oplus s_3(t) \oplus s_4(t) \\
s_5'(t+T) &= s_3(t) \oplus s_4(t) \oplus s_5(t) \\
s_6'(t+T) &= s_4(t) \oplus s_5(t) \oplus s_6(t) \\
s_7'(t+T) &= s_5(t) \oplus s_6(t) \oplus s_7(t)
\end{aligned} \tag{2}$$

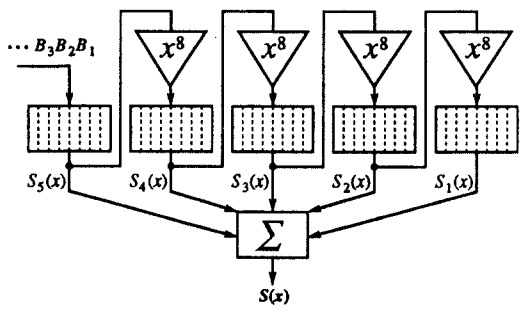


Fig 3. Block diagram of the algorithm *Successive*.

As noted this method is similar to the algorithm *Direct* in terms of the amount of hardware and the speed of operation. One advantage over *Direct* is the possibility of a modular implementation using the identical XOR network at each stage. This fact leads to the development of the third algorithm with much reduced hardware.

2.3 Algorithm III : Recursive

The third algorithm termed *Recursive* computes the syndrome in a radically different way. Note that the *successive* use of the four identical multipliers in series for the algorithm *Successive* can be replaced with the *recursive* use of a single multiplier x^8. To accomplish this, the output of the first syndrome register in Fig 3 is fed back to the input as depicted in Fig 4.

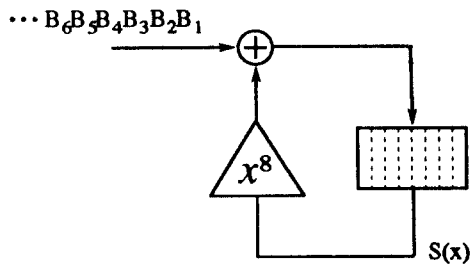


Fig 4. Updating syndrome based on a recursive multiplier x^8.

Let $S(t)$ and $S(t+T)$ be the syndrome at time t and $t+T$, respectively. Then for each byte B applied the syndrome will be updated as

$$S(t+T) = R_{g(x)}[x^8 S(t)] + B$$

and this expression can be expanded as follows.

$$\begin{aligned}
s_0(t+T) &= s_0(t) \oplus s_6(t) \oplus s_7(t) \oplus d_0 \\
s_1(t+T) &= s_0(t) \oplus s_1(t) \oplus s_6(t) \oplus d_1 \\
s_2(t+T) &= s_0(t) \oplus s_1(t) \oplus s_2(t) \oplus s_6(t) \oplus d_2 \\
s_3(t+T) &= s_1(t) \oplus s_2(t) \oplus s_3(t) \oplus s_7(t) \oplus d_3 \\
s_4(t+T) &= s_2(t) \oplus s_3(t) \oplus s_4(t) \oplus d_4 \\
s_5(t+T) &= s_3(t) \oplus s_4(t) \oplus s_5(t) \oplus d_5
\end{aligned}$$

$$s_6(t+T) = s_4(t) \oplus s_5(t) \oplus s_6(t) \oplus d_6$$
$$s_7(t+T) = s_5(t) \oplus s_6(t) \oplus s_7(t) \oplus d_7 \quad (3)$$

Conscious readers would notice that the set of these recursive equations is similar to the formula employed for HEC generations such as in Bellcore's ATM chip[5]. These equations can also be used for the *cell by cell* CRC verification.

Note that the configuration in Fig 4 does not have seperate registers to store the subsyndromes of individual bytes. The content of the register is the syndrome of *the entire bytes applied up to present*. This is in a sharp contrast to the cases of algorithms *Direct* and *Successive*, where their outputs are determined by *the most recent five octets only*. For this reason, the syndrome register in Fig 4 should be initialized to zero before the first byte B_1 is applied. After five byte-clock($5T$) periods, the register will contain the syndrome of $H=[B_1B_2B_3B_4B_5]$. If the syndrome turns out to be nonzero, the syndrome of the reassumed header $H=[B_2B_3B_4B_5B_6]$ must be checked. The content of the register at $6T$, however, will be the syndrome of $[B_1B_2B_3B_4B_5B_6]$, six octets altogether. Therefore we must remove the effect of the oldest octet B_1. This subtracting process can be derived as follow.

$$\begin{aligned} R_{g(x)}[B_2\cdots B_6] &= R_{g(x)}[B_1\cdots B_6] - R_{g(x)}[B_1 00000] \\ &= R_{g(x)}[B_1\cdots B_6] \oplus R_{g(x)}[x^{40}B_1] \\ &= R_{g(x)}[B_1\cdots B_6] \oplus R_{g(x)}[(x^6+x^5+x)\cdot B_1] \end{aligned}$$

By combining the columns 2, 6, and 7 of Table 1, the XOR network for multiplying the oldest octet by x^{40} is obtained as

$$\begin{aligned} R_{g(x)}[B_1x^{40}] = {} & (d_1\oplus d_2\oplus d_6\oplus d_7)x^7 + (d_0\oplus d_1\oplus d_5\oplus d_6)x^6 \\ & + (d_0\oplus d_4\oplus d_5)x^5 + (d_3\oplus d_4\oplus d_7)x^4 \\ & + (d_2\oplus d_3\oplus d_6)x^3 + (d_1\oplus d_2\oplus d_5\oplus d_7)x^2 \\ & + (d_0\oplus d_2\oplus d_4\oplus d_7)x + (d_2\oplus d_3\oplus d_7) \end{aligned}$$

As can be seen from Fig 5, the significance of the algorithm *Recursive* is that the required amount of hardware is independent of the block size. It uses only the two sets of XOR networks with a little additional circuitry : x^8 for updating the syndrome and x^{40} for removing the effect of the oldest octet. Out of the three algorithms on parallel CRC verification, the algorithm *Recursive* requires the least amount of XOR networks and can operate at a higher speed than others. Accounting these, we have concluded that the algorithm *Recursive* is the most suitable in the parallel verification of CRC for ATM cell delineation.

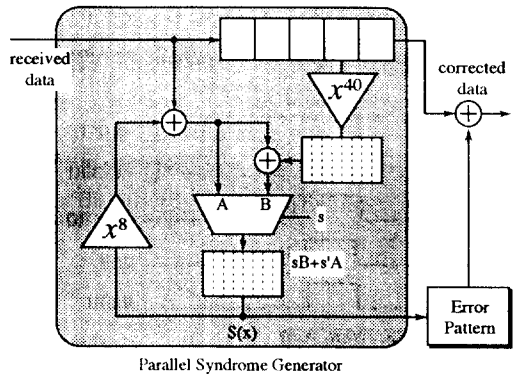


Fig 5. Block diagram of the algorithm *Recursive*.

III. An ATM cell delineation module

In this section an ATM cell delineation module is realized to show the effectiveness of the algorithm *Recursive*. According to CCITT recommendations, the cell delineation and scrambling/descrambing block performs the following functions of PL-TC sublayer : (1) HEC generation and scrambling the payloads at the transmitter side and (2) cell delineation and descrambling the payloads at the receiver side. The SDH(Synchronous Digital Hierarchy)-based transmission with STM-1(155.520 Mbps) framing is assumed. Therefore the self-synchronous scrambler/descrambler ($x^{43}+1$) is adopted. To achieve the speed of 155.520 Mbps required for STM-1 transmission, an octet parallel implementation has

been considered. For the design and simulation of logic circuits, CAD tools including WorkView and ALS have been used. The circuits have been placed within two chips of Actel FPGAs(Field Programmable Gate Array), called CDSB-Tx and CDSB-Rx, respectively[7].

The required timing diagram of the control signals is shown in Fig 6. The control signals are generated under the following assumptions. (1) The CRC verification is performed *byte by byte* in the *HUNT* state. Once the receiver enters into the *PRESYNC* or *SYNC* state, however, the CRC is checked *cell by cell*. (2) Error correction will not be carried out in the *HUNT* state since the receiver can leave the *HUNT* state only when a CRC agreement is found, that is, only when an error free header is encountered. In the *PRESYNC* or *SYNC* state, the receiver of course should correct any single bit errors by subtracting(XOR*ing*) the suitable error patterns from headers during the error correction cycle.

The signal *SYN-EN* remains *HIGH* during the syndrome computation. With the signal *ERR-DETECT* set to *HIGH*, the value of syndrome is examined to set the signal *ERR*. In the *PRESYNC* or *SYNC* state, the receiver checks the CRC cell by cell. The corrected header being aligned with the signal *CELL-SYNC* is to be delivered to the upper layer. In the *HUNT* state, on the contrary, the receiver keeps checking the CRC byte by byte until it sees the zero syndrome. Let us consider the case illustrated in Fig 6(b). The receiver first computes the syndrome for the assumed header $H=[B_1B_2H_1H_2H_3]$, not a genuine header. Since the syndrome can not be zero, the signals *ERR* and *HERR*—meaning a HEC error in the *HUNT* state—are set to *HIGH*. Therefore the signal *ERR-DETECT* remains *HIGH* extending the syndrome computation cycle one clock period further. At the next clock, the syndrome for the reassumed header $H=[B_2H_1H_2H_3H_4]$ will not be zero again, and all the signals retain their previous values. Finally at the next clock the receiver encounters a genuine header $H=[H_1H_2H_3H_4H_5]$. Since the syndrome will be zero this time, the control signals *ERR*, *HERR*, and *ERR-DETECT* change their values to *LOW* in that order and the receiver enters into the *PRESYNC* state.

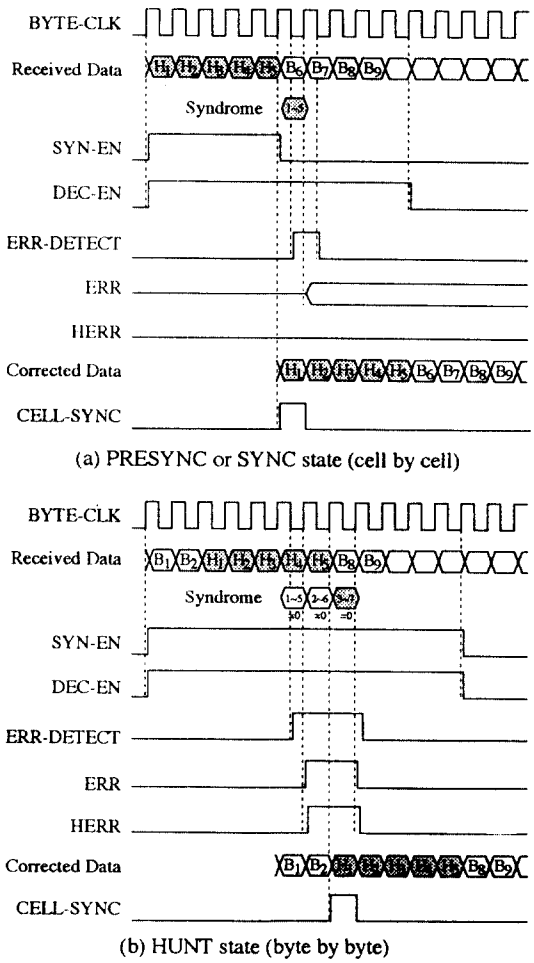


Fig 6. Required timing for the ATM cell delineation.

Ⅳ. Conclusions

This paper discussed three algorithms on parallel CRC verification, termed *Direct*, *Successive*, and *Recursive*, respectively. The first and second algorithms are similar in terms of the amount of hardware and the speed of operation. The required

amount of hardware increases linearly with the number of octets involved in CRC computation. One advantage of the algorithm *Successive* over *Direct* is the possibility of modular implementaion, *i.e.*, the identical XOR network can be used in each stage. On the other hand, the third algorithm *Recursive* requires a fixed amount of hardware independent of the block size.

As an application of the algorithm *Recursive*, an ATM cell delineation module suitable for STM 1 transmission operated at a rate 155.520 Mbps has been successfully realized with field programmable gate arrays.

It is not difficult to generalize the algorithms presented in this paper to the case involving more than five octets in CRC computation. The bitwise algorithms can also be derived from these bytewise algorithms with a minor modification.

References

1. CCITT Draft Recommendation I.432 : B-ISDN User-Network-Interface (UNI) Physical Layer specification.
2. CCITT Draft Recommendation I.362 : B-ISDN ATM Adaptation Layer (AAL) functional description.
3. CCITT Draft Recommendation I.363 : B-ISDN ATM Adaptation Layer (AAL) specification.
4. Richard E. Blahut, "Theory and Practice of Error Control Codes," Addison-Wesley Publishing Company, 1983.
5. Cesar A Johnston and H. Jonathan Chao, "The ATM layer Chip : An ASIC for B-ISDN Applications," *IEEE JSAC*, Vol. 9, No. 5, pp. 741 750, June 1991.
6. Y. S. Kim, S. I. Choi, H. S. Park, and J. K. Kim, "Implementation of Parallel Cyclic Redundancy Check Code Encoder and Syndrome Calculator," The Journal of the Korean Institiute of Communication Sciences, Vol. 18, No. 1, pp. 83-91, Jan. 1993.
7. S. Song, *et al.*, "The ATM Cell Delineation and Forward Error Control," Research report, Feb. 1993.

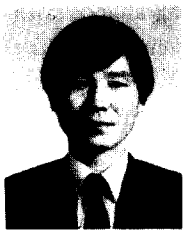
崔 允 姬(Youn Hee Choi) 학생회원

1968년 12월 22일생

1991년 2월 : 전북대학교 전자공학과 졸업

1992년 3월 ~ 현재 : 전북대학교 전자공학과 석사과정

※주관심분야 : 채널코딩이론, 정보이론, 이동통신기술 등



宋 祥 燮(Sang Seob Song) 종신회원

1954년 6월 15일생

1978년 2월 : 전북대학교 전기공학과 졸업

1980년 2월 : 한국과학기술원 전기 및 전자공학과(석사)

1990년 8월 : 아나나 마니토바 대학 전기 및 컴퓨터공학과 (박사)

1981년 4월 ~ 현재 : 전북대학교 전자공학과 교수

※주관심분야 : 채널코딩이론, 정보이론, 이동통신기술, High speed network 신호처리 등