분산표본혼화기의 병렬구현

*정헌주 김재형 정성현 박승철
현대전자산업주식회사 정보통신연구소
서울시 강남구 대치동 996-1

Parallel Implementation of Distributed Sample Scrambler

*HeonJoo Joeng JaeHyeong Kim SungHyoun Jeong SeungChul Park
Hyundail Electronics Industries Co. Ltd.
996-1, Daechi-dong, Kangnam-ku, Seoul, Kerea
heonjoo@hei.co.kr

Abstract

This paper presents a method and implementation of the parallel distributed sample scrambler(DSS) in the cell-based ATM transmission environment. In the serial processing, it requires very high speed clock because the processing clock of the serial DSS is equal with the data transmission speed. In this paper, we develop a conversion method of the serial SRG(shift register generator) to 8bit parallel realization. In this case, it has a sample data processing problem which is a character of DSS.

So, a theory of correction time movement is presented to solve this problem. We has developed a ASIC using this algorithm and verified the recommendation of ITU-T, I.432

1. 서 론

DSS 방식은 SDH 전송기반의 FSS(Frame Synchronous Scrambling)방식과 기본적으로는 같지만 동기를 잡는 방법에 있어서는 서로 다르다.[1] FSS 방식은 프레임단위로 동기가 이루어 지지만 DSS 는 송신측에서 샘플값을 보내고 수신측에서 이를 이용하여 동기를 잡는다. 즉, 수신측에서는 수신 데이터에서 샘플값을 추출하고 자체적으로 동작하는 SRG(Shift Register Generator)에서 샘플링한 값과 비교하여 정정과정을 거침으로써 동기 성취 과정을 수행한다.

본 논문에서는 이를 병렬로 처리하는 방법으로써 정정시간 변경 과정과 8 비트 단위로 데이터를 변환하는 과정을 제시하고자 한다.

2. 직렬 DSS 방식의 스크램블러의 개요

BISDN 물리계층의 기능은 상위 ATM 계층으로부터 내려온 ATM 셀들을 수집 및 정리하여 물리 매체에 전송하고, 그 역과정을 수행하는 것이다. 이 때 SDH (Synchronous Digital Hierarchy)기반 전송인 경우 각종 오버헤드를 첨가하여 프레임을 구성하여 데이터가 전송되고 FSS 방식에 의해 혼화되지만, 셀 기반 전송인 경우 프레임을 사용하지 않고 ATM 셀 흐름 그대로를 전송하며 DSS 방식에 의해 혼화된다. 또한 FSS 방식이 프레임단위로 혼화 되는 것과는 달리 DSS 방식은 SRG 를 이용한다는 면에서는 같지만 송신측에서 샘플을 보내고 수신측에서 이 샘플값을 이용하여 동기를 잡는다는 차이가 있다. 아래 그림 1 은 혼화기와 역혼화기의 기본 개념도이다.

그림 2 는 송수신의 SRG 의 특성다항식이 $x^{31}+x^{28}+1$ 인 ITU-T. I_432 에서 권고한 직렬 역혼화기의 구조이고 정정벡터 CV' 는

$$CV' = [\,0110100110111001100111011000100\,]$$

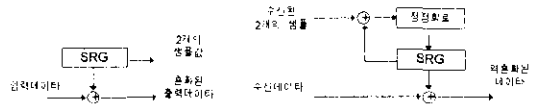
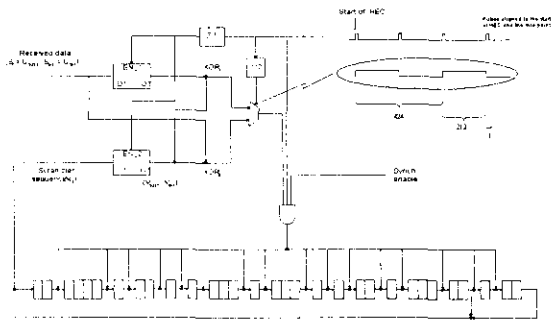


그림 1. 혼화기와 역혼화기의 개념도



그 림 2. ITU-T. I_432 의 직렬 역혼화기구조

이다. 또한 그림 3 은 위의 직렬 역혼화기가 동작했을 때 타이밍도이다. 송신측에서는 헤더 오류 제어 (HEC)구간을 제외한 전체에 SRG 랜덤수열이 한 비트씩 2 진 연산으로 더해지고, HEC 구간 첫 비트에는 현재 셀 유료부하보다 211 비트 전의 SRG 수열값 (U_{t-211})이, 두번째 비트에는 현재의 SRG 수열값 (U_{t+1})이 더해진다. 수신측에서는 송신측 SRG 와 똑같은 방식으로 랜덤수열를 만들어 내서 1/2 셀 간격마다 추출하여 이 값(V_{t-211}, V_{t+1})을 전달된 샘플값과 비교하여 서로 다른 경우 정해진 정정시간에 맞추어 위에 제시된 정정벡터 CV'에 의해 정정된다. 여기서 정정시간은 셀의 시작점으로부터 33 번째와 이로부터 212 번 떨어진 245 번째이다. [2][3][4]

이러한 직렬 DSS 방식을 병렬로 구현하기 위해서는 우선 정정시간의 변경이 필요하고 실제 직렬을 8 비트단위로 전환하는 일을 수행하게 된다.

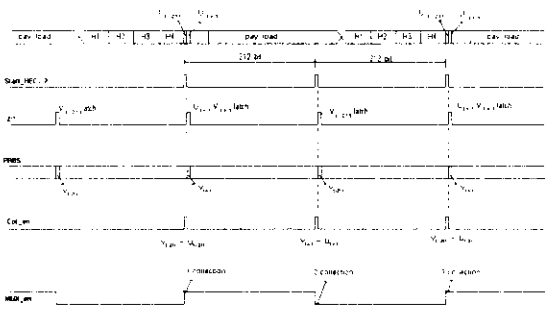


그림 3. 직렬 DSS의 타이밍도

3. 정정시간의 변경

그림 4 는 특성다항식이 $x^{31}+x^{28}+1$ 인 직렬 SRG 이며 31 개의 레지스터로 구성되고 한 클럭에 하나의 랜덤 비트 값이 한 비트의 데이터화 2 진 연산되어 혼화된다.

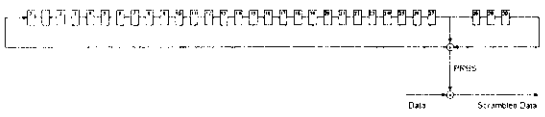


그림 4. 특성다항식 $x^{31}+x^{28}+1$ 의 SRG 구성

이러한 SRG 에서 초기 레지스터값을 알면 앞으로의 레지스터 상태를 예측할 수 있고(Forward), 전의 레지스터 상태를 알 수도 있다(Backword). 이를 이용하여 정정시간을 변경하고자 한다. 그림 5 와 같은 x^3+x+1 의 SRG 를 예로 들어보면

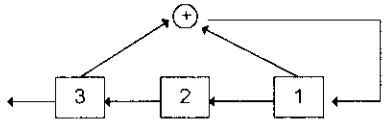


그림 5. 특성다항식 x^3+x+1 의 SRG 구성예

상태행렬 $\mathbf{A}$ 는 다음과 같고, 레지스터 초기값 r_0 은 $[1\ \ 1\ \ 1]^T$ 이다.

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

이 때 t = 0 일 때

$$\begin{bmatrix}1\\1\\1\end{bmatrix} = r_0 = \mathbf{A}^0 \bullet r_0$$

t = 1 일 때

$$\begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}0&1&0\\1&0&1\\0&0&1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \mathrm{A}\bullet r_0$$

t = 2 일 때

$$\begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}0&1&0\\0&0&1\\1&0&1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}0&1&0\\0&0&1\\1&0&1\end{bmatrix}^2\begin{bmatrix}1\\1\\1\end{bmatrix} = \mathbf{A}^2\bullet r_0$$

t = 3 일 때

$$\begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}0&1&0\\0&0&1\\1&0&1\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}0&1&0\\0&0&1\\1&0&1\end{bmatrix}^3\begin{bmatrix}1\\1\\1\end{bmatrix} = \mathbf{A}^3\bullet r_0$$

$$\vdots$$

t = 8 일 때

$$\begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}0&1&0\\0&0&1\\1&0&1\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}0&1&0\\0&0&1\\1&0&1\end{bmatrix}^8\begin{bmatrix}1\\1\\1\end{bmatrix} = \mathbf{A}^8\bullet r_0$$

다시 정리하면,

$t=0: \quad r_0$

$t=1: \mathbf{A}\bullet r_0 = r_1$ --- 식(1)

$t=2: \mathbf{A}\bullet r_1 = \mathbf{A}[\mathbf{A}\bullet r_0] = \mathbf{A}^2\bullet r_0 = r_2$ --- 식(2)

$t=3: \mathbf{A}\bullet r_2 = \mathbf{A}[\mathbf{A}^2\bullet r_0] = \mathbf{A}^3\bullet r_0 = r_3$ --- 식(3)

$t=4: \mathbf{A}\bullet r_3 = \mathbf{A}[\mathbf{A}^3\bullet r_0] = \mathbf{A}^4\bullet r_0 = r_4$ --- 식(4)

$t=5: \mathbf{A}\bullet r_4 = \mathbf{A}[\mathbf{A}^4\bullet r_0] = \mathbf{A}^5\bullet r_0 = r_5$ --- 식(5)

$t=6: \mathbf{A}\bullet r_5 = \mathbf{A}[\mathbf{A}^5\bullet r_0] = \mathbf{A}^6\bullet r_0 = r_6$ --- 식(6)

$t=7: \mathbf{A}\bullet r_6 = \mathbf{A}[\mathbf{A}^6\bullet r_0] = \mathbf{A}^7\bullet r_0 = r_7$ --- 식(7)

$t=8: \mathbf{A}\bullet r_7 = \mathbf{A}[\mathbf{A}^7\bullet r_0] = \mathbf{A}^8\bullet r_0 = r_8$ --- 식(8)

결론적으로 $\mathbf{A}^k$ 은 k 클럭 후의 레지스터 상태이고, r_8 은 8 클럭 후의 레지스터 값이다.

이번엔 t = 1 일 때 레지스터값 r_1 을 알면 1 클럭 전의 레지스터값 r_0 를 알 수 있다. 식(1)의 $\mathbf{A}\bullet r_0 = r_1$ 의 양변에 $\mathbf{A}^{-1}$ 를 곱하면

$$\mathbf{A}^{-1}[\mathbf{A}\bullet r_0] = \mathbf{A}^{-1}\bullet r_1 \quad \text{--- 식(9)}$$

그러면, $r_0 = \mathbf{A}^{-1}\bullet r_1$ 이고, $\mathbf{A}^{-1}=\mathbf{B}$ 라 하면

$$r_0 = \mathbf{B}\bullet r_1 \quad \text{--- 식(10)}$$

이번엔 t = 8 일 때의 레지스터값 r_8 를 알면 8 클럭 전의 레지스터값 r_0 를 알 수 있다. 식(8)의 $\mathbf{A}^8\bullet r_0 = r_8$ 양변에 $[\mathbf{A}^8]^{-1}$ 를 곱하면

$$[\mathbf{A}^8]^{-1}\bullet[\mathbf{A}^8\bullet r_0] = [\mathbf{A}^8]^{-1}\bullet r_8 \quad \text{--- 식(11)}$$

그러면, $r_0 = [\mathbf{A}^8]^{-1}\bullet r_8$ 이고, $\mathbf{A}^{-1}=\mathbf{B}$ 라 하면

$$r_0 = \mathbf{B}^8\bullet r_8 \quad \text{--- 식(12)}$$

결론적으로 $\mathbf{B}^k$는 k 클럭 전의 레지스터상태이고, r_s를 현재 레지스터 초기값이라면 r_0는 8 클럭 전의 레지스터 값이다. 이와 같이 SRG 의 전, 후의 값을 추출할 수 있다는 점을 이용하여 역혼확기의 정정시간을 변경하고자 한다.

그림 6 에서와 같이 직렬 DSS 에서는 1 셀 마다 2 개의 샘플값이 전달되어 212 비트마다 비교되고 정정동작이 발생한다. 그러므로 이 때 정정이 일어날 수 있는 경우는 3 가지 경우가 있다. 첫번째는 셀의 시작점으로부터 33 번째에서만 정정이 일어나는 경우, 두번째는 245 번째에서만 일어나는 경우, 세번째는 33 번째와 245 번째에서 2 번 일어나는 경우이다.
본 논문에서는 위 3 가지 경우 모두 정정시간을 셀의 시작점으로 옮김으로써 병렬처리가 가능하도록 했다.

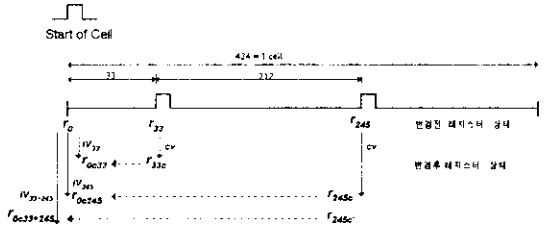


그림 6. 한 셀에서의 변경된 정정시간

i. 33 에서만 정정이 일어날 경우

송신측에서 전달된 샘플값(U_{t-211})과 수신측 SRG 의 샘플값(V_{t-211})이 서로 다른 경우로서, 직렬 DSS 에서는 Start of Cell (SOC)를 현재 기준시점으로 했을 때 33 클럭 후에 정정동작이 일어난다. $x^{31}+x^{28}+1$ 의 특성다항식에 의한 상태 행렬을 **A** 라 하자. 이 때 SRG 초기값(모두 "1")은 r_0이다. 33 번 이동된 후의 레지스터의 상태는 위에서 증명했듯이 $\mathbf{A}^{33}$ 이고, 레지스터값은 다음과 같다.

$$\mathbf{A}^{33} \bullet r_0 = r_{33} \quad \text{---- 식(13)}$$

이 때 정정이 일어날 것이므로

$$\mathbf{A}^{33} \bullet r_0 + CV = r_{33c} \quad \text{---- 식(14)}$$

여기서 CV 는 ITU-T. I.431 규격에 있는 정정벡터이고, r_{33c} 는 정정된 후의 레지스터 값이다.

그러면, 처음 SOC 시점에서 어떠한 레지스터값을 갖고 있어야만 현재 r_{33c} 값을 갖을 수 있을까? 이 값을 구하기 위해서는 r_{33c}를 초기값으로 하여 33 번 뒤로 이동된 값을 유추하면된다.

A^{33}와 역관계인 $\mathbf{B}^{33}$은 33 클럭 전의 레지스터이고, 식(14)의 r_{33c} 를 초기값으로 이용하면

$$\mathbf{B}^{33} \bullet r_{33c} = \mathbf{B}^{33}[\mathbf{A}^{33} \bullet r_0 + CV] = r_0 + \mathbf{B}^{33} \bullet CV = r_{0c33} \quad \text{---- 식(15)}$$

여기서 r_{0c33} 는 33 번째의 레지스터값이 r_{33c}로 정정되었을 때 SOC 에서 이 값이 발생되기 위한 초기값이다.

또한 식(15)에서 $\mathbf{B}^{33} \bullet CV = IV$ 라 하면

$$r_0 + IV = r_{0c33} \quad \text{---- 식(16)}$$

이고, r_0 와 r_{0c33}의 관계는 초기값 r_0에 IV_{33} *(Inverse Vector)* 벡터를 더해줌으로써 벡터값 r_{0c33}을 구할 수가 있다.

이와 같이 SOC 로부터 33 번째에서 정정이 일어나는 경우에는 단지 SOC 시점에서 초기 레지스터값을 IV_{33}에 의해 바꿔주면 된다. 현재 구현된 SRG 의 초기값은 모두 1 로 셋팅되었으므로 IV_{33}은 다음과 같다.

$$IV_{33} = [011101011111001110111011100111 1]^T$$

ii. 245 번째에서 정정이 일어 날 경우

원리는 위에서 기술했던 바와 같으며, 처음 SOC 의 초기상태에서 앞으로 245 번 이동시킨 레지스터 상태행렬은 $\mathbf{A}^{245}$ 이고 이 때 레지스터값은

$$\mathbf{A}^{245} \bullet r_0 = r_{245} \quad \text{---- 식(17)}$$

이 때 정정이 일어날 것이므로

$$\mathbf{A}^{245} \bullet r_0 + CV = r_{245c} \quad \text{---- 식(18)}$$

여기서 r_{245c} 는 정정된 후의 레지스터값이다.

그러면 SOC 시점에서 245 클럭 후에 r_{245c}를 갖기 위해서는 어떤 초기값을 가져야 하는가? $\mathbf{A}^{245}$와 역관계인 $\mathbf{B}^{245}$인 레지스터상태와 식(18)의 초기값을 이용하면

$$\mathbf{B}^{245} \bullet r_{245c} = r_{0c245} \quad \text{--- 식(19)}$$

식(18)를 식(19)에 대입하면

$$\mathbf{B}^{245}[\mathbf{A}^{245} \bullet r_0 + CV] = r_{0c245} \quad \text{--- 식(20)}$$

$$r_0 + \mathbf{B}^{245} \bullet CV = r_{0c245} \quad \text{--- 식(21)}$$

여기서 $\mathbf{B}^{245} \bullet CV$를 IV_{245} 라 하면 다음과 같이 쓸 수 있다

$$r_0 + IV_{245} = r_{0c245} \quad \text{--- 식(22)}$$

결과값은 다음과 같다.

$$IV_{245} = [101111010110000000010111011110 00]^T$$

iii. 33 와 245 에서 정정이 일어났을 경우

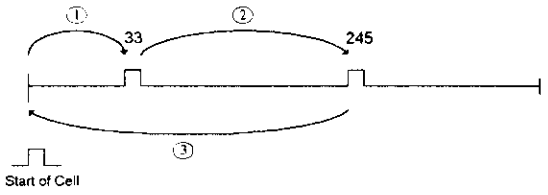


그림 7. SRG 의 변환시점

33 번에서 정정이 일어나면 레지스터는 다음과 같다.

$$\mathbf{A}^{33} \bullet r_0 + CV = r_{33c} \quad \text{---- 식(23)}$$

이 때 정정된 레지스터값 r_{33c}을 가지고 212번 앞으로 이동시켜 정정된 레지스터값은 다음과 같다.

$$\mathbf{A}^{212} \bullet r_{33c} = \mathbf{A}^{212}[\mathbf{A}^{33} \bullet r_0 + CV]$$

$$= \mathbf{A}^{245} \bullet r_0 + \mathbf{A}^{212} \bullet CV = r_{245c'} \quad \text{---- 식(24)}$$

$r_{245c'}$를 초기값으로 하여 245번 뒤로 이동시키면

$$\mathbf{B}^{245} \bullet r_{245c'} = \mathbf{B}^{245}[[\mathbf{A}^{245} \bullet r_0 + \mathbf{A}^{212} \bullet CV] + CV]$$
$$= r_0 + \mathbf{B}^{245} \bullet \mathbf{A}^{212} \bullet CV + \mathbf{B}^{245} \bullet CV$$
$$= r_0 + \mathbf{B}^{33} \bullet CV + \mathbf{B}^{245} \bullet CV$$
$$= r_0 + [\mathbf{B}^{33} + \mathbf{B}^{245}]CV = r_{0c33+245}$$
$$\text{---- 식(25)}$$

여기서 $[\mathbf{B}^{33} + \mathbf{B}^{245}]CV = IV_{33+245}$ 라 하면

$$r_0 + IV_{33+245} = r_{0c33+245} \quad \text{---- 식(26)}$$

IV_{33+245}결과는 다음과 같다

$$IV_{33+245} = [11001000100100111001010101101111]^T$$

위 3가지 경우에서 초기 레지스터값 r_0를 각각의 IV에 의해 SOC 시점에서 변환시키고, 샘플값 비교결과에 따라 정정이 없을 경우와 더불어 MUX 회로를 통해 선택한다.

4. 8 비트단위의 데이터처리 방법

31 개의 시프트 레지스터에서 나오는 출력을 현 시점에서 8 비트 단위로 동작시키고 그 데이터출력을 병렬처리를 구현할 수 있다. 즉, 그림 4 에서와 같은 특성다항식 $x^{31}+x^{28}+1$ 의 SRG 상태방정식 $\mathbf{A}$ 는 31 × 31 행렬로 표현된다. 이 때 8 번 이동 의 $\mathbf{A}^8$ 의 SRG 상태는 다음과 같다. 이것을 회로로 구현하면 그림 9 와 같다.

$\mathbf{A}^8 =$

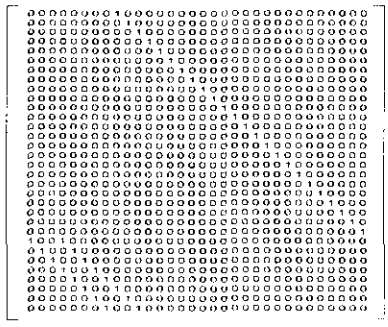


그림 8. $\mathbf{A}^8$의 행렬

5. 구 현

그림 9 는 전체 기능블록도이다. SOC 를 기준시점으로 하여 수신측 SRG 의 랜덤수열(Psudo Random Binery Sequence, PRBS)에서 추출한 2 개 샘플값과 송신측에서 전달된 2 개의 샘플값을 비교 했을 때 샘플값이 서로 같아 정정이 일어나지 않는 경우는 NO_COR 블록의 출력이, 샘플 U_{t-211}과 V_{t-211}만이 서로 다를 경우는 33C 블록의 출력이, 샘플 U_{t+1}과 V_{t+1}만이 서로 다를 경우는 245C 블록의 출력이, 두개의 샘플이 다 다를 경우는 33C245C 블록의 출력

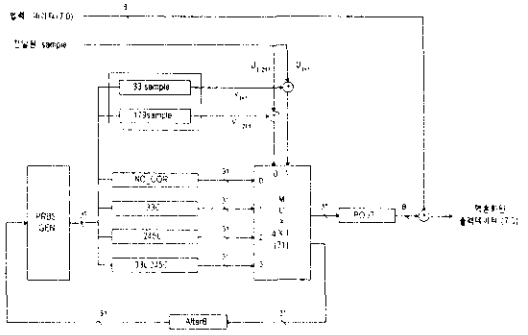


그림 9. 병렬 DSS 의 기능블록도

이 4 대 1 MUX 에서 선택되어 출력된다.

POUT 블록은 31 비트의 출력을 8 비트단위로 출력하는 역할을 하여 입력데이터와 2 진 연산 되어 역혼화된 데이터를 출력한다. 또한 AFTER8 블록은 MUX 의 출력을 SRG 로 구성된 PRBS_GEN 블록으로 8 비트단위로 업데이트시켜 주는 역할을 한다. 이것은 $\mathbf{A}^8$을 회로로 구현한 것으로 31 비트의 데이터가 입력되고 병렬 클럭 1 클럭에 의해 8 비트단위씩 데이터가 처리되고 31 비트의 출력(OUT[30:0])은 PRBS_GEN 블록으로 피드백된다. 이중 출력 0 번 비트부터 7 번 비트(OUT[7:0])까지는 8 비트단위 출력으로 8 비트 병렬된 랜덤 샘플이다.

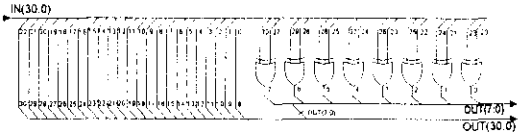


그림 10. 8 비트단위의 데이터처리 회로

5. 결론

본 논문에서 ATM 전송방식의 물리계층에 관한 ITU-T I.432 표준안의 셀 기반 전송에 사용되는 직렬 분산 표본혼화방식을 병렬로 처리하는 방법을 구현한 것이다. 여기서 제시된 알고리즘은 ASIC 으로 구현되어 20Mbps 급 동기식 전송시스템에 실장되었고, ITU_T I.432 규격을 만족하는 안정된 동작이 확인되었다.

참고문헌

1. DooWhan Choi, "Parallel scrambling techniques for digital multipexer," AT&T Technical Journal, pp.123 – 136, Sept/Oct. 1986.
2. 이병기, "광대역 정보통신", 교학사, pp.290 – pp.298, 1994
3. 이병기, 김석창, "Recent Advances in Theory and Applications of Scrambling Techniques for Lighwave Transmission " PROCEEDINGS OF THE IEEE, VOL. 83, NO. 10, OCTOBER 1995
4. ITU-T Recommendation I.432, B-ISDN User-Network Interface Physical Layer Specification. June 1992.