고속 이더넷 응용을 위한 10b/8b 디코더의 설계

차 근호*, 손 승일*, 최 익성**

한신대학교*, ETRI**

A Design of 10b/8b Decoder for High-Speed Ethernet Applications.

Keun Ho Cha*, Seung Il Sonh*, Ick sung Choi**

*HanShin University, **Electronnics and Telecommunications Research Institute

E-mail : chakeunho@hotmail.com

요 약

본 논문에서는 고속 이더넷의 고속의 이더넷의 물리계층에서 수신된 비트열로 부터 클록을 복원하고, 이 클록으로부터 동기된 비트열을 10b/8b 디코딩한 다음, 바이트열로 복원하여 데이터 링크계층의 MAC(Media Access controller)로 전송한다. PCS의 디코더는 8비트의 데이터와 제어신호를 추출하여 MAC으로 전달하는 기능을 수행한다. 즉 본 논문에서는 PCS기능 중 가장 중요한 요소인 10b/8b 디코더를 VHDL언어를 사용하여 기술하고 *Xilinx ISE5.1*를 이용하여 구현하였고, 입력 부분에 DDR인터페이스를 사용하였다. 구현한 결과 1056개의 게이트 사용하였으며, 10Gbps를 지원하기 위해서는 한 블록 당 2.5Gbps의 처리속도가 필요하다. 설계 모듈은 5.1Gbps의 처리 속도를 지원하여 관련 응용분야에서 사용이 가능할 것으로 사료된다.

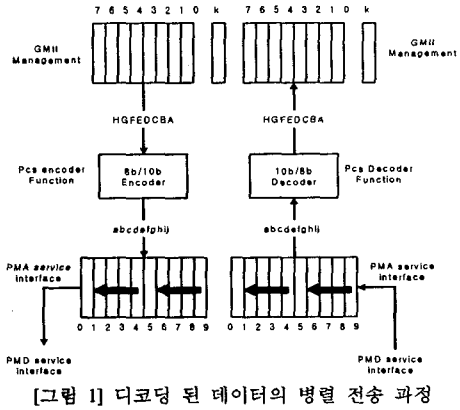
1. 서 론

10b/8b 코드는 Fiber Channel 시스템을 위하여, IBM에서 1980년대 중반에 설계되었다[1]. 이것은 8비트의 데이터에 대하여, 10비트로 인코딩 된 코드를 다시 8비트의 데이터로 복원하는 기능을 수행한다. 10b/8b 기능을 수행하는 이유는 인코더에서 수신클럭을 추출하기 위해서 수신비트열에 있는 1과 0이 충분히 변화되어 수신클럭을 복원하기 쉬워지게 하였고 코드 그룹간에 경계를 용이 하게 추출할 수 있게 하였다. 이와 같은 코딩 방법은 하나 또는 여러 개의 비트오류를 감지할 수 도 있다. 이러한 장점들로 인하여 10b/8b 디코더를 이용하여 인코딩 된 데이터 8비트의 데이터로 복원하는 기능이 필요하게 되었고, 기가비트 이더넷에 적용하게 되었으며, 고속의 10Gbps 이더넷에 적용하여 물리계층에서 수신된 신호를 증폭한 후 신호의 세기를 일정하게 변환하여 이 신호로부터 클럭을 복원하고, 이 클럭을 이용하여 비트들을 추출하고, 추출된 비트열을 병렬화시켜, PCS(Physical Coding Sublayer)의 10b/8b 디코더로 전달한다. PCS의 디코더는 8비트의 데이터와 제어신호를 추출하여 MAC(Media Access controller)으로 전달하는 기능이 필요하다. 10b/8b는 PCS계층에서 병렬화 기능을 수행한다.

본 논문에서는 고속의 10Gbps 이더넷의 물리계층과 데이터링크 계층 사이의 PCS에 적용 가능한 10b/8b 디코더를 VHDL언어를 사용하여 기술하고 *Xilinx ISE5.1*를 이용하여 구현하였다[1].

2. 10b/8b 기능

10b/8b의 기능은 PMA(Physical Medium Attachment)에서 10비트의 직렬 데이터를 병렬로 10비트 전송한다.

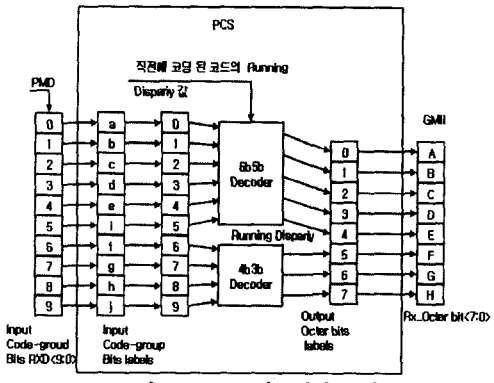


[그림 1] 디코딩 된 데이터의 병렬 전송 과정

데이터의 디코딩 되지 않은 바이트를 a,b,c, d,e,i,f,g,h,j 라고 하고, 입력된 데이터의 값이 제어 변수인 경우 제어신호를 활성화 하여 K 표현한다. 만약 10비트가 데이터 문자인 경우 제어 신호를 비활성화 하여 표현하고, 문자로는 D로 표현한다. 10 비트의 데이터는 디코딩 과정을 통하여 8비트 문자 A,B,C,D,E,H,G,F의 결과를 출력한다. 그림 1은 디코딩 된 데이터의 병렬 전송 과정을 나타낸 그림이다[1].

10b/8b 코딩 과정은 2단계로 정의된다. 그림 2에서 보여주는 것처럼 첫 번째 단계에서는 6b/5b 디코더를 이용해서 변환시킨다. 두 번째 단계에서는 4b/3b 디코더로 데이터 바이트의 나머지 4비트를 디코더 한다.

디코딩 방법은 10비트의 데이터를 바이트로 디코딩 된 문자로 만드는 것은 RD(Running Disparity)를 이용한다. RD는 2개의 코드로 구성되어 있다. 이 중 6b/5b 디코더는 RD을 이전 값이 (+) 혹은 (-)인가에 따라 6비트의 데이터를 5비트의 일련의 값으로 디코딩 하며, 디코더에 초기 RD값은 (-) 혹은 (+)이든 상관없다. 디코딩 값에 따라 변경된 RD값은 다음 4b/3b 디코더 과정에서 사용하며, 4비트를 3비트의 값으로 디코딩 하는 과정에서 발생한 RD값은 다음에 들어오는 10비트 데이터의 6b/5b 디코딩 값에 영향을 준다. 여기서 RD의 개념은 (-) 또는 (+)값을 지닌 파라미터로 정의된다. 다시 말해, 그 과정은 RD의 조건에 따라 현재 상태 값으로 다음 상태 값을 얻는다[1].



[그림 2] 10b/8b 디코더의 구성

표1 은 6비트의 데이터가 입력이 들어오면 RD값의 해 계산되어 5비트의 코드를 선택하여 출력하게 된다. 이렇게 하는 것은 인코더에서 10비트 코드그룹 당 비트 변환 율이 3회에서 8회까지 되기 때문에 동기 추출이 유리하고, 자연스럽게 평균전압이 0이 되게 한 것을 제어코드인지 데이터 코드인지를 확인하여 복원하기 위해 한 것이다.[2].

[표 1] 6b/5b 코딩 테이블

Input		output	Dx.y	Rd	Input		output	Dx.y	Rd
abcdei	abcdei	EDCBA			abcdei	abcdei	EDCBA		
011000	100111	00000	D0.y		100100	011011	10000	D16.y	-Rd
100010	011101	00001	D1.y	-Rd	100011		10001	D17.y	
010010	101101	00010	D2.y		010011		10010	D18.y	
110001		00011	D3.y	Rd	110010		10011	D19.y	Rd
001010	110101	00100	D4.y	-Rd	001011		10100	D20.y	
101001		00101	D5.y		101010		10101	D21.y	
011001		00110	D6.y	Rd	011010		10110	D22.y	
000111	111000	00111	D7.y		000101	111010	10111	D23.y	-Rd
000110	111001	01000	D8.y	-Rd	001100	110011	11000	D24.y	
100101		01001	D9.y		100110		11001	D25.y	Rd
010101		01010	D10.y		010110		11010	D26.y	
110100		01011	D11.y	Rd	001001	110110	11011	D27.y	-Rd
001101		01100	D12.y		001110		11100	D28.y	Rd
101100		01101	D13.y		010001	101110	11101	D29.y	
011100		01110	D14.y		100001	011110	11110	D30.y	-Rd
101000	010111	01111	D15.y	-Rd	010100	101011	11111	D31.y	

표2는 4b/3b는 입력된 데이터를 제어코드 인지 데이터 코드인지를 확인하여 인코더 된 데이터를 디코딩하기 위한 것이다.[2].

[표 2] 4b/3b 코딩 테이블

Inputs		Output	Dx.y	Rd
f g h j	f g h j	HGF		
0100	1011	000	Dx.0	-Rd
1001		001	Dx.1	
0101		010	Dx.2	Rd
0011	1100	011	Dx.3	
0010	1101	100	Dx.4	-Rd
1010		101	Dx.5	
0110		110	Dx.6	Rd
0001	1110	111	Dx.7	-Rd

3. 제어 코드

데이터 코드 외에 표3과 같이, 12개의 특수 코드 그룹이 추가적으로 정의되는데, 이 코드 그룹은 Kx.y로 표기 한다. 12개의 특수 코드그룹 중에서, 기가비트 이더넷에서는 5개의 코드그룹만 사용된다. K27.7 코드는 패킷에 시작을 알려주고, K29.7 코드는 패킷에 끝을 알려준다. K30.7 코드는 전송 오류를 K28.5 코드는 Idle 신호이며 연속적으로 반복해서 생성된다. 이것은 상대방 시스템과의 클록을 동기화하고, 유지하기 위해서 연속적인 패턴을 제공한다. K23.7 코드는 캐리어 이벤트의 길이를 연장하기 위해 사용하거나 버스트 프레임 사이에 IFG 기간 동안 각 프레임의 경계를 구분하기 위해 사용된다[3].

특수 코드그룹 중, K28.1, K28.5, K28.7 코드그룹은 "00111", "11000"의 비트열이 공통적으로 들어있다. 이 비트열들이 들어있는 코드그룹을 콤마열(Comma)이라 부른다. 기가비트 이더넷에서는 K28.5만 사용된다. 이 코드는 최대한 비트가 변하는 값이므로 동기추출에 유리한 특징을 갖고 PMA가 코드그룹의 경계를 결정하는데 매우 유용하다[4].

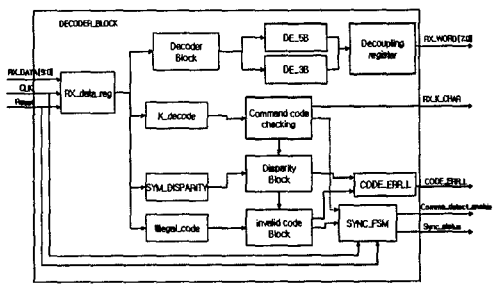
[표 3] 특수 코드그룹

Code out roup value Name	Octet Bit HGF EDCBA	Current RD- Abcdei fghj	Current RD+ abcdei fghj	Notes
K28.0 1C	000 11100	001111 0100	110000 1011	1
K28.1 3C	001 11100	001111 1001	110000 0110	1,2
K28.2 5C	010 11100	001111 0101	110000 1010	1
K28.3 7C	011 11100	001111 0011	110000 1100	1
K28.4 9C	100 11100	001111 0010	110000 1101	1
K28.5 BC	101 11100	001111 1010	110000 0101	2
K28.6 DC	110 11100	001111 0110	110000 1001	1
K28.7 FC	111 11100	001111 1000	110000 0111	1,2
K23.7 F7	111 10111	111000 1000	000101 0111	
K27.7 FB	111 11011	110110 1000	001001 0111	
K29.7 FD	111 11101	101110 1000	010001 0111	
K30.7 FF	111 11110	011110 1000	100001 0111	

NOTE 1- 예약됨
NOTE 2- Comma가 있는 것임.

4. 10b/8b 디코더 설계

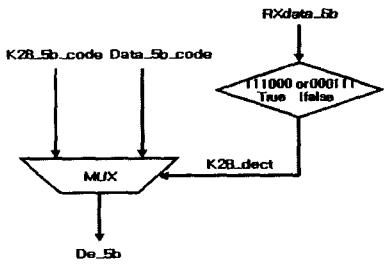
10b/8b 디코더는 PMD(Physical Medium Dependent) 10비트가 직렬로 입력되고, 이는 PMA (Physical Medium Attachment)에서 병렬로 PCS (Physical Coding Sublayer)로 전송하여 디코딩 과정을 거쳐 바이트의 데이터와 제어 코드 , 에러 코드, 동기 상태, 콤마코드 인지의 유무 신호를 MAC (Media Access controller)로 전송한다. 다음 그림 3은 10b/8b 전체 블록도를 보여준다.



[그림 3] 10b/8b 디코더 블록도

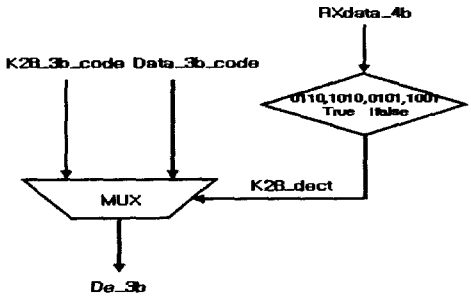
4.1 Decoder block 처리부

디코더 블록은 6b/5b, 4b/3b 처리부로 나눠진다. 6b/5b블록은 입력된 데이터 중 6비트를 받아드려 5비트로 출력하는데 입력된 데이터 중 K28_dect신호가 활성화가 되면, 데이터 값이 달리 입력되어야 한다. 다음 그림 4는 6b/5b의 데이터 처리과정을 보여주고 있다.



[그림 4] 6b/5b 데이터 처리 블록도

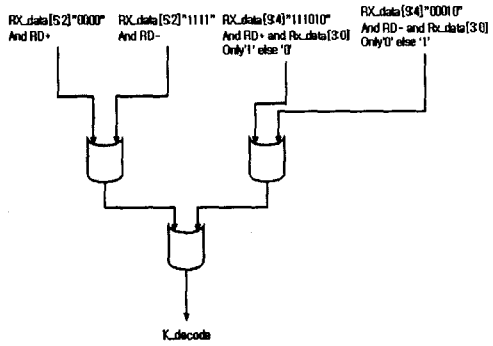
4b/3b블록은 6b/5b와 같은 기능을 수행한다. 입력된 데이터 중 4비트를 받아드려 3비트로 출력하는데 K28_dect 신호가 활성화되면 데이터의 값이 달리 입력되어야 한다. 그림 5는 4b/3b 데이터 처리 과정을 보여주고 있다.



[그림 5] 4b/3b 데이터 처리 블록도

4.2 Command code checking 처리부

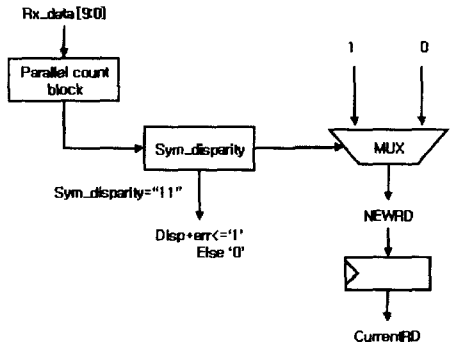
Command code checking 처리부는 10비트의 입력되는 데이터 값 중 제어코드 인지를 확인하여 제어코드 일 경우 k_decode 신호를 활성화 하여 입력된 데이터의 값이 제어코드 임을 알려준다. 다음 그림 6은 Command code 처리부의 블록도를 보여주고 있다.



[그림 6] Command code checking 블록도

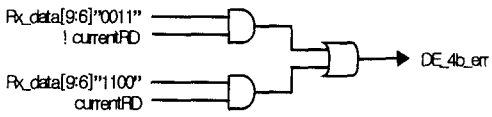
4.3 Disparity 블록

Disparity 블록은 10비트의 입력 값의 '1'의 개수를 parallel counter 블록에서 개산 후 currentRd값을 결정한다. '1'의 수가 '0'보다 많거나 '0'의 개수가 '1'보다 많으면 currentRd값은 반전하게 된다. 만약 '1'개수와 '0'의 개수가 같다면 currentRd값은 전 상태를 유지하고 '1'과 '0'의 개수의 비율이 4개, 5개 ,6개 이외의 값이 나오면 Disp_err신호를 활성화 시킨다. 다음 그림 7은 Disparity블록을 보여주고 있다.

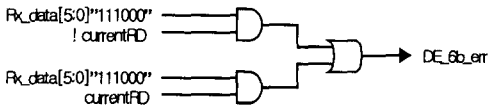


[그림 7] Disparity 블록도

다음 그림 8, 9는 Disparity가 발생하지 않지만 반전 돼야 하는 경우를 처리하기 위해 나타낸 블록도이다.



[그림 8] 4b/3b invalid 데이터 처리부



[그림 9] 6b/5b invalid 데이터 처리부

4.4 invalid code checking 처리부

유효하지 않은 코드가 들어오면 잘못된 코드가 입력되었음을 알리는 처리부이다. 유효하지 않는 코드값은 아래에 보는 것과 같고 유효하지 않는 코드가 들어오면 illegal code를 활성화시킨다.

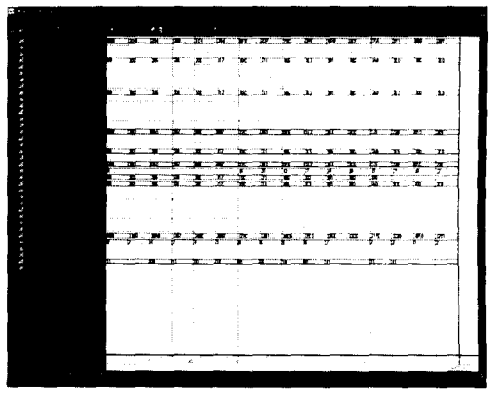
- a = b= c= d
- e = i = f = h = j
- p13 & !e & !i
- !i = e = g = h = j
- p31 & e & I
- (e=!i=g=h=j)&!(c=d=e)
- f = g = h = j
- !p31&e&!i&!g&!h&!j
- !p13&!e&i&g&h&j
- !a&!b&c&d&e&i&!f&!g&!h&i

p13은 입력된 4b 값 중 하나의 비트만 '1'이고 나머지 비트가 모두 '0' 인경우이다.

p31은 입력된 4b 값 중 하나의 비트만 '0'이고 나머지 비트가 모두 '1'인경우이다.

5.시뮬레이션 파형

그림 10은 10b/8b의 시뮬레이션 한 파형을 보여준다. 리셋이 되고 난 후 10비트의 데이터가 입력된다. 입력된 데이터는 6b/5b, 4b/3b블록에서 CurrentRd값에 따라 8비트의 데이터로 변경된 후 k28_dect 신호가 활성화가 되지 않고 illegal_code가 활성화 되지 않으면 정상적인 데이터가 축력된 것을 볼 수 있다. 그리고 k28_dect 신호가 활성화가 되면 제어코드이므로 다른 코드로 변경되어 출력된다. illegal_code 활성화되면 잘못된 데이터가 출력되는 것 을 볼 수 있다.



[그림 10] 10b/8b 시뮬레이션 파형

6. 결론

10b/8b 디코터는 물리계층에서 수신된 신호를 증폭한 후 신호의 세기를 일정하게 변환하여 이 신호로부터 클럭을 복원하고, 이클럭을 이용하여 비트들을 추출하고, 추출된 비트열을 병렬화시켜, PCS(Physical Coding Sublayer)의 10b/8b 디코더로 전달한다. PCS의 디코더는 8비트의 데이터와 제어신호를 추출하여 MAC으로 전달하는 기능을 수행한다.

본 논문에서는 10b/8b 디코더를 10Gbps 이더넷에 적용하기 위해 속도를 향상 시켜 VHDL언어를 사용하여 기술하고 Xilinx ISE5.1을 이용하여 구현하였다. 이를 위해 최대한의 병렬 처리를 하였으며, 입력 부분에 DDR인터페이스를 사용하였다. Xilinx툴에서 구현한 결과 1056개의 게이트 사용하였으며, 10Gbps를 지원하기 위해서는 한 블록 당 2.5Gbps의 처리속도가 필요하다. 설계 모듈은 5.1Gbps의 처리 속도를 지원하여 관련 응용분야에서 사용이 가능할 것으로 사료된다.

참고문헌

[1] A.X. Widmer, P.A. Franaszek,"ADC-DALANCED PARTITIONED-BLOCK, 8B/10B TRANSMISSION CODE", IBM journal of Research and Development, September, 1983

[2] IEEE Std 802.3, "CSMA/CD Access method and physical layer specifications", 2000
[3] 윤종호 저, "네트워크 엔지니어를 위한 최신 이더넷", 교학사, 2002. pp.229-391
[4] 허정화, 박노경, 박상봉, "직렬 ATA 용 8b/10b 인코더와 디코더 설계 및 구현" 한국통신학회논문지 '04-1 Vol.29 No.1A, pp.93-97, january, 2004
[5] "8b/10b Encoder v4.0", Xilinx, March, 2003