工學碩士 學位請求論文

1GBps급 유럽형 CATV 하향 스트림 채널 부호화 알고리즘 FPGA 구현

An FPGA Implementation of Channel Coding Algorithm for European CATV Down Stream of 1[Gbps]

指導敎授 鄭 智 元

2008年 **2**月

韓國海洋大學校 大學院

電 波 工 學 科

金 憼 赫

本 論文을 金憨赫의 工學碩士 學位論文으로 認准함.

委員長 : 工學博士 金 基 萬 (인)

委 員 : 代表理事 韓 龍 俊 (인)

委 員 : 工學博士 鄭 智 元 (인)

2008年 **2**月

韓國海洋大學校 大學院

電 波 工 學 科

金 憨 赫

차 례

Nomenclature

c(x) : (127,122) RS 부호 최종 출력 생성 다항식

c_ : Extended 심볼 생성 다항식

$\hat{c}$: (128,122) extended RS 부호 최종 출력 생성 다항식

g(x) : RS 부호 원시 다항식

p(x) : RS 부호 parity 생성 원시 다항식

r(x) : RS 부호 parity 다항식

$s(x)$: 신드롬 계산 일반식

X_j, Y_j : j 시점에서의 Differential encoder의 출력값

W_j, Z_j : j 시점에서의 Differential encoder의 입력값

Abbreviation

BM : Branch Matric

FPGA : Field Programmable Gate Array

FSM : Finite State Machine

LUT : Look Up Table

QAM : Quadrature Amplitude modulation

PM : Path Matric

PS : Path Select signal

RS : Reed-Solomon

TCM : Trellis Coded Modulation

UW : Unique sync_Word

VHDL : VHSIC Hardware Description Language

Abstract

Originally, cable networks were established to provide TV signals to a community that otherwise could not receive reliable TV signals. With subsequent addition of more channels, cable TVs have gained enormous popularity in the general segment as well. With the advent of the Internet and other digital communications, however, cable networks and channels have become a focus for transmitting digitized information at high speed and bandwidth. This is because a cable network can provide a high speed digital communication channel in addition to well known traditional cable services.

In cable networks, cable modems provide high speed data transporting functions between a cable network and a connected user. Cable modems typically are implemented by a forward error correction(FEC) scheme. The ITU-T Recommendation J-38 Annex B specifies using 64- and 256- quadrature amplitude modulation (QAM). The cable transmission block includes an FEC encoder, an FEC decoder, and a cable channel. The FEC encoder encodes data using conventional FEC schemes for transmission to the FEC decoder through the cable channel.

One of candidate coding schemes is a concatenate coding scheme which is combined RS code and convolutional code. Concatenate coding schemes are considered as being the best solution for powerful protection of digital information against nonlinear and fading noise channel. However, the convolutional code is not appropriate to high-order modulation such as 64-QAM and 256-QAM. Therefore instead of using convolutional code, TCM(Trellis Coded Modulation) is best solution for cable modem. The FEC encoder includes a Reed Solomon (RS) encoder, a convolutional interleaver, a randomizer (e.g., scrambler), and a Trellis Coded Modulation (TCM) encoder.

This thesis analyzed the performance of cable modem by computer simulation and implemented the cable modem by FPGA chip.

In implementing the cable modem, there are some problems to fabricate and fitting on FPGA chip. First, many clocks are needed in implementing cable modem because of different code rate and different modulation types. To reduce the number of clocks, we use the two memories, which are different clock speed for reading and writing data. Second, this system lost the bit-synchronization and frame-synchronization in decoder, the system recognize that all data is error. This thesis solves the problems by using simple 5-stage registers and unique sync-word.

Based on solutions for about problems, the cable modem is fabricated on FPGA chip name as Vertex II pro xc2vp30-5 by Xilinx, and we confirmed the effectiveness of the results.

제 1 장 서 론

현대 사회가 정보화 사회로 발전함에 따라 통신망을 통한 데이터의 교환이 급격하게 증가하게 되었다. 방송 통신망에서 대표적으로 위성을 이용한 방송 통신망(DVB-S, DVB-S2), 지상파를 이용한 방송 통신망(DVB-T, DVB-H), 그리고 케이블을 이용한 방송 통신망(DVB-C)이 있다[1]-[5]. 본 논문에서는 케이블 망에서 채널 부호화 방식에 대해 연구하였다. 모든 방송망에서 데이터가 채널을 통과할 때 채널 상태에 따라 잡음이 첨가되며, 오류가 발생하게 된다. 이러한 오류를 찾아내고 정정하여 성능을 향상시키기 위해 통신 시스템에서는 채널 부호화 방식이 쓰이고 있다. 방송을 위한 통신시스템이나 데이터 전송을 위한 통신시스템에서는 주로 연접부호가 사용된다. 연접부호는 두 가지 이상의 부호를 연결해 사용하여 높은 부호이득을 얻어낼 수 있어서 각광을 받고 있다. 연접 부호화 방식은 주로 RS 부호와 convolutional 부호화 방식이 결합되는 방식인데, 이는 고차 변조 방식에서의 성능이 미비하여, 고차 변조 방식과 결합한 트렐리스 부호화 방식(TCM, Trellis Coded Modulation)이 주로 적용되고 있으며, 이 또한 케이블 방송에서 적용되고 있다[6]-[12].

현재 사용되고 있는 있는 케이블망의 표준인 ITU-T J38 은 유럽형 CATV 에 관한 표준이며, Series J 는 텔레비전과 음향 그리고 그 외의 멀티미디어 신호 전송에 관한 것으로 J38 은 그 중 디지털 CATV 시스템에 관한 것이다. 이 표준은 1Gbps 급을 요구하고 있어 고차 변조 방식 적용이 용이한 RS 부호 및 TCM 방식을 사용하고 있다[2].

본 논문은 케이블 망에서 적용되고 있고 채널 부호화 방식의 알고리즘을 분석 및 Visual C++ 언어를 이용하여 시뮬레이션을 하였고, 또한 구현상의 문제점을 제시하고 해결하여 FPGA 칩으로 설계하였다.

구현상의 문제점으로는 첫째로 케이블망의 오류 정정 방식의 부호화율에 따른 클럭이 많이 필요 하는데 본 논문에서는 메모리를 이용하여 필요한 클럭수로 줄였으며, 둘째로는 채널 복호화기에서 동기를 잃어버리면 복호 데이터가 모두 오류로 처리되는데, 본 논문에서는 몇 개의 레지스터를 이용하여 동기를 맞추었다. 셋째로, 복호단의 동기는 비트 단위의 동기인 반면 프레임간의 동기를 또한 맞추어야한다. 방송 데이터 통신인 경우 데이터 프레임의 맨 앞 또는 맨 뒤에 UW(Unique sync-Word)를 사용한다. 이는 수신단에서 데이터 프레임 동기를 맞추기 위해 사용된다. 동기를 맞추는 것은 매우 중요하다. 복호단에서 복호되어진 데이터들을 원래의 데이터 프레임으로 구성을 해야되는데, 이때 데이터 프레임의 동기를 맞추지

못한다면 원래의 데이터 프레임이 구성되지 않아 올바른 방송을 볼 수 없게 된다. 따라서 본 논문에서는 이러한 UW 를 맞추어 구현하였다.

이러한 알고리즘 분석 및 구현상의 문제점 해결을 토대로 본 논문에서는 ITU-T J38 annex B 의 하향 스트림 채널 부호화 시스템을 VHDL 언어를 사용하여 FPGA 칩에 직접 구현하였다. 본 논문의 구성은 제 1 장 서론에 이어 제 2 장에서는 채널 부호화 시스템의 각 블록에 대한 알고리즘 분석 및 시뮬레이션 결과를 보이고, 제 3 장에서는 위의 세가지 구현상의 문제점 해결 방안, 제 4 장에서는 구현 결과 그리고 제 5 장의 결론으로 본 논문의 끝을 맺는다.

제 2 장 1 Gbps 급 CATV 하향 스트림 채널 부호화 방식

다음 그림 2-1은 CATV 하향 스트림 채널 부호화 시스템의 모델이다.

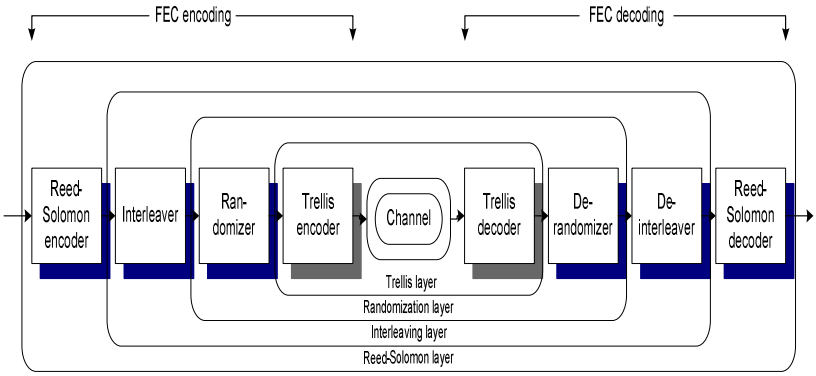


그림 2-1. CATV 하향 스트림 채널 부호화 시스템 모델

Fig 2-1. Model of channel coded system for CATV down stream channel.

CATV 하향 스트림 채널 부호화 시스템은 연접 부호로서 내부 부호(inner code)에는 RS(Reed-Solomon) 부호, 외부 부호(outer code)에는 TCM을 사용하고, 두 채널 부호 사이에는 interleaver와 randomizer를 사용하여 연집 오류에 강인하게 하여 성능을 향상 시킨다.

2.1 RS (Reed-Solomon) Code

RS 부호는 BCH 부호의 한 종류로서 I.S Reed와 G Solomon에 의해 1960년에 제안된 에러정정부호이다. RS부호는 복수 비트 단위로 정정하는 것이 특징이다. 심볼 단위로 데이터를 다룰 경우나 에러가 분산되어 발생할 경우 등에 적합한 에러정정 부호이다.

2.1.1 부호기

일반적인 RS 부호의 경우 한 심볼이 7비트의 데이터로 구성될 경우 한 블록당 출력 심볼 개수는 127개이다. 이는 비트 수를 n이라고 했을 때 $2^n - 1$개의 출력을 가지기 때문이다. 그림 2-2는 CATV 하향 스트림 채널 부호화 시스템에 사용된 RS encoder의 구조이다.

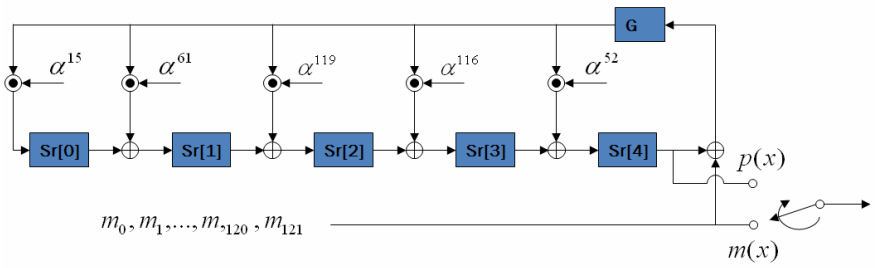


그림 2-2. RS Encoder의 구조

Fig 2-2. The structure of RS encoder.

데이터 심볼 m은 0~121까지 122개가 입력 되어진다. 122개의 입력심볼은 바로 출력이 됨과 동시에 shift register Sr(4)와 더해져서

부호화기에 feedback되어 지고, 122개의 입력 심볼이 모두 출력이 되면 스위치가 올라가 shift register Sr(4)의 값이 출력이 되어지게 된다. 따라서 5개의 shift register로 인해 127개의 출력 심볼이 생성되어진다. 즉, 122/127의 부호화율을 가지는 (127,122) RS 부호이다.

(n,k,t) RS 부호에서 n은 출력 심볼의 개수, k는 입력 심볼의 개수, t는 RS 부호 한 블록의 정정 능력을 나타낸다. 이들은 n-k=2t의 관계를 가진다. 그림 2-2의 구조를 가지는 RS 부호는 (127,122,2)가 되고, 한 블록당 2심볼의 에러까지 정정 할 수 있는 능력을 가진다.

CATV 시스템에서는 127개의 출력 심볼외에 extended 심볼 하나를 더 만들어 내어 에러 정정능력을 향상시킨다. 이를 extended RS 부호라고 한다. Extended 심볼은 다음 식에 의해 만들어 진다. 식 (2-1)의 원시 다항식을 이용하여 식(2-2)의 원시 근으로 생성다항식을 만든다.

$$p(x) = x_0^7 + x^3 + 1 \qquad \cdots\cdots\cdots\cdots\cdots\cdots (2-1)$$

$$g(x) = (x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4)(x+\alpha^5) = x^5 + \alpha^{52}x^4 + \alpha^{116}x^3 + \alpha^{119}x^2 + \alpha^{61}x + \alpha^{15}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots (2-2)$$

원신호를 $m(x) = m_{121}x^{121} + m_{120}x^{120} + \cdots + m_1x + m_0$ 이라 하고, 잉여 데이터를 $r(x) = r_4x^4 + r_3x^3 + r_2x^2 + r_1x + r_0$ 라 하면 부호화된

신호는

$$c(x) = m_{121}x^{126} + m_{120}x^{125} + m_{119}x^{124} + \cdots + r_4x^4 + r_3x^3 + r_2x^2 + r_1x + r_0$$

이 된다. 원래의 c(x)는 (127,122,2)의 부호화 비트에 대한 다항식이며, extended RS 부호 (128,122,3)의 생성다항식은 식 (2-3)에 의해 생성된다.

$$c\_ = c(\alpha^6)$$

$$\hat{c} = xc(x) + c\_ = m_{121}x^{127} + m_{120}x^{126} + m_{119}x^{125} + \cdots + r_4x^5 + r_3x^4 + r_2x^3 + r_1x^2 + r_0x + c\_$$

..... (2-3)

송신되는 데이터의 순서는 $m_{121}m_{120}m_{119}\cdots m_1m_0r_4r_3r_2r_1r_0c\_$ 로 왼쪽에서 오른쪽으로의 순서로 전송하게 된다.

(127,122,2) RS 부호의 최종 출력은 C(x)에서이다. 여기에 x 대신 α^6을 대입하면 식 (2-3)에서 처럼 하나의 심볼 C_가 생성되어 진다. 이에 식 (2-3)에 의해 extended RS 부호의 최종 출력 $\hat{c}$가 생성되어 진다. 이에 따라 (127,122,2) RS 부호는 출력 심볼이 하나 더 생성되므로 (128,122,3) RS 부호가 되어 한 블록당 에러 정정 능력이 높아 지게 된다.

2.1.2 복호기

그림 2-3은 RS decoder의 구조이다. RS decoder는 우선 encoding된 블록에 대해 신드롬을 계산한다. 신드롬 계산 식은 식 (2-4)와 같다.

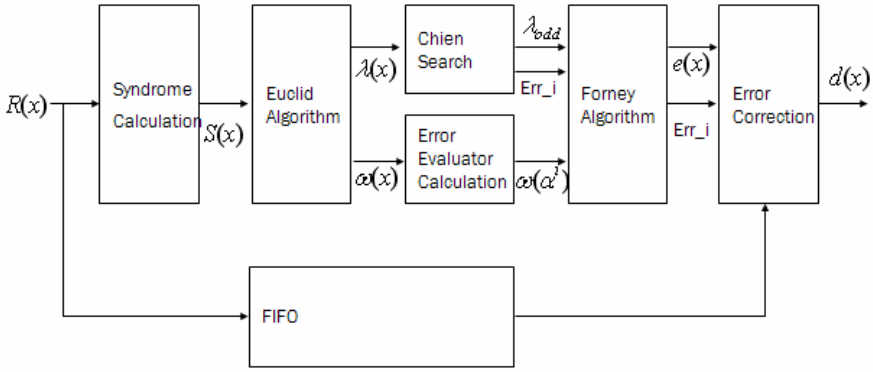


그림 2-3. RS Decoder의 구조

Fig 2-3. The structure of RS decoder.

$$s(x) = \sum_{k=1}^{2t} s_k x^{k-1} = s_1 + s_2 x^1 + \ldots + s_{2t} x^{2t-1} \qquad (2-4)$$

처음 신드롬 계산 과정에서 에러가 없다고 판별되면 decoding과정이 필요치 않다. 하지만 신드롬 계산 과정에서 에러가 판별되면 chien search를 사용한 euclid 알고리즘과 forney 알고리즘을 이용하여 블록 내에서의 에러 위치를 찾고, 정정을 한다. 만일 심볼 에러의 개수가 정정 능력을 벗어난다면 euclid와 forney를 거친 후 다시 에러의 유무 판단을 위한 신드롬 계산에서 에러가 판별된다. 이때는 정정 능력을 벗어났다는 판단하에 decoding과정을 거치지 않고, 에러를 가진

데이터를 그대로 가지고 있는다. 그 이유는 (128,122)의 경우 t=3의 정정 능력을 갖는데, 만약 4개의 심볼에서 에러가 발생했을 때 decoding과정에서 더 많은 에러가 발생할 수 있기 때문에 에러가 난 4개의 심볼을 그대로 가지고 있는게 더 심한 성능 열화를 방지해주기 때문이다.

2.2 Interleaver

Inner code인 RS 부호는 burst 에러에 취약함을 보인다. 에러가 연속적으로 발생할 경우 블록내의 정정 능력을 쉽게 벗어나기 때문이다. 따라서 burst 에러를 방지하기 위해 inner Code와 outer Code 사이에 interleaver를 넣어준다. CATV 하향 스트림 채널 부호화 시스템에서는 convolutional interleaver를 사용한다. 그림 2-4는 convolutional interleaver의 구조이다.

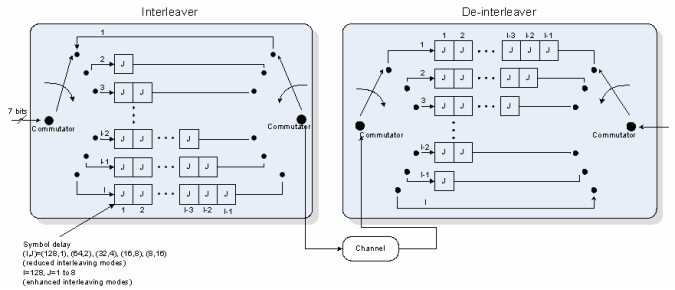


그림 2-4. Convolutional Interleaver의 구조

Fig 2-4. The structure of convolutional interleaver.

그림 2-4에서 보는 바와 같이 convolutional interleaver는 구현시 상당히 많은 수의 레지스터를 필요로 한다. 하지만 이렇게 사용되는 레지스터가 많을수록 하드웨어의 처리 속도가 느려지고 구현이 복잡해진다. 이런 문제점을 해결하기 위해 본 논문에서는 레지스터 대신 메모리를 사용하였다. 메모리 사용에 대한 구조는 그림 2-5와 같다.

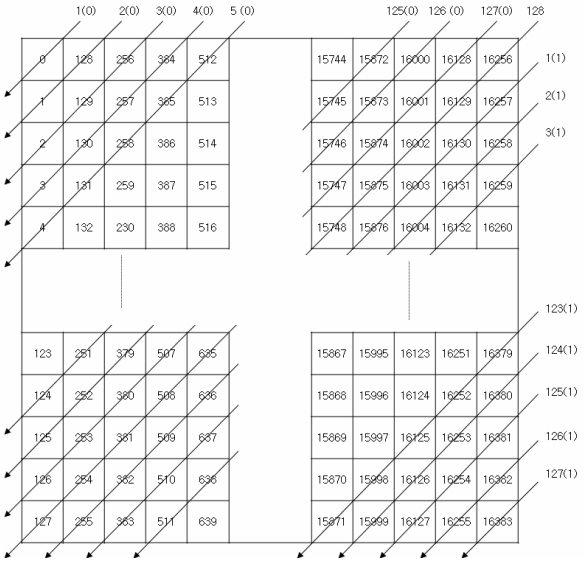


그림 2-5. 메모리를 사용하여 구성한 Convolutional Interleaver

Fig 2-5. Convolutinal interleaver using memory.

데이터 심볼은 메모리내의 주소값 순서대로 저장을 하면서 읽는 순서는 화살표 방향의 순서대로 한다. 이런 방법으로 구현을 할 경우 레지스터의 사용에 따른 문제가 해결된다.

Deinterleaver의 과정은 interleaver의 역순으로 우선 128x128x7의 크기를 갖고 있는 메모리에 화살표 방향으로 저장한 후 메모리가 다 채워지는 시점에서 메모리 내의 주소값에 따라 데이터를 읽어오기 시작한다.

2.3 Randomizer

그림 2-6은 randomizer의 구조이다.

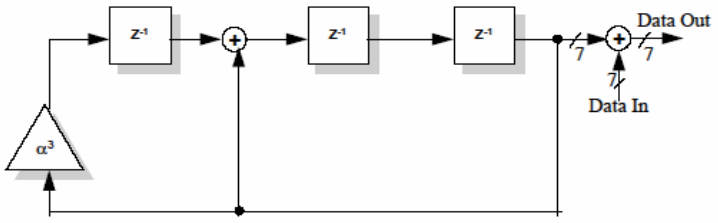


그림 2-6. Randomizer의 구조

Fig 2-6. The structure of randomizer.

Randomizer는 복조시 전력의 안정을 위해 사용되어진다. 예를 들어 '1'의 값이 연속해서 나오는 경우 이 신호에서 DC 성분이 발생해 과전력이 발생하게 된다. 이러한 현상을 방지하기 위해 randomizer가 사용된다. 그림 2-6에서 보듯이 CATV 하향 스트림 시스템에서는 세 개의 shift register를 사용하여 데이터를 심볼로 처리한다. 각 레지스터의 초기값은 모두 '1'이고, 데이터 프레임이 바뀔 때 마다 초기화 된다. Derandomizer는 randomizer와 구조와 동작 모두 같다.

2.4 TCM(Trellis Coded Modulation)

CATV 하향 스트림 시스템에서는 두가지 모드의 변조방식이 사용된다. 이 두가지 방식은 64-QAM과 256-QAM이다. RS 부호, interleaver, randomizer는 변조 방식에 관계없이 동작하지만 TCM의 경우 변조 방식에 따라 coded data와 uncoded data가 틀려진다. 그림 2-7는 64-QAM 모드의 TCM 구조이다.

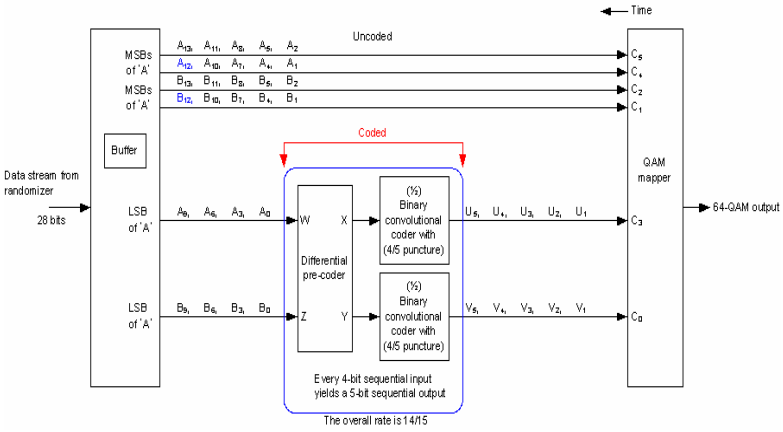


그림 2-7. TCM의 구조(64-QAM)

Fig 2-7. The structure of TCM(64-QAM).

부호화율 14/15인 64-QAM TCM encoder는 총 6비트로 구성된 심볼을 데이터로 사용한다. Phase ambiguity를 방지하기 위해 90° invariance differential code와 부호화율 4/5의 convolutional code로 구성되어진다.

그림 2-7에서 보듯이 8개의 bit가 coding 되어 10개가되고, 20개의 uncded_bit에 의해 28 data_bit가 30 data_bit, 총 5개의 TCM 심볼이 생성된다.

256-QAM의 경우 uncoded_bit가 30 bits이므로 부호화율은 19/20이 되고, 한 심볼은 8 bit로 구성된다.

2.4.1 Differential Code

Differential code는 phase ambiguity를 방지하기 위해 사용된다. CATV 하향 스트림 시스템에서는 90° 의 위상 틀어짐을 방지한다. 식 2-5는 그림 2-7의 differential encoder를 나타낸다. J는 시점을 나타낸다.

$$X_j = W_j + X_{j-1} + Z_j(X_{j-1} + Y_{j-1}),\quad Y_j = Z_j + W_j + Y_{j-1} + Z_j(X_{j-1} + Y_{j-1}) \quad (2\text{-}5)$$

식 (2-5)에 의해 coding 된 데이터는 식 (2-6)에 의해 decoding 된다.

$$W_j = X_j + X_{j-1} + (X_{j-1} + Y_{j-1})(X_j + Y_j), Z_j = X_j + Y_j + X_{j-1} + Y_{j-1} \quad (2\text{-}6)$$

2.4.2 Convolutional Encoder

그림 2-8은 convolutional encoder의 구조이다.

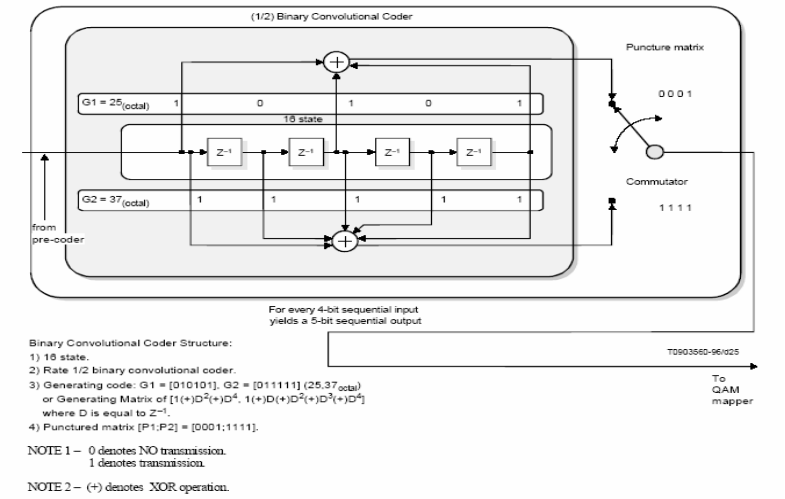


그림 2-8. Convolutional Encoder의 구조

Fig 2-8. The structure of convolutional encoder.

(2,1,5) convolutional encoder의 부호화율은 4/5이다. 이는 그림 2-8에서 puncture matrix에 의해 만들어진다.

2.4.3 Mapping and Viterbi Decoder

그림 2-9는 64-QAM에서의 mapping 포인트를 나타낸다.

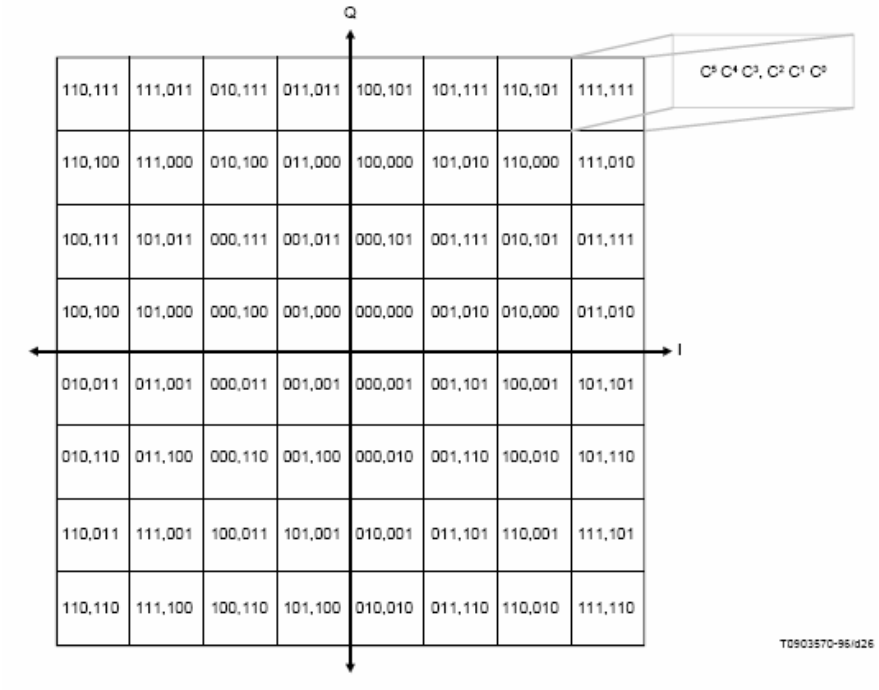


그림 2-9. 64-QAM에서의 Mapping 포인트

Fig 2-9. The mapping point in 64-QAM.

그림 2-9에서 C0, C3가 coded_bit이므로 나머지 uncoded_bit는 90° 로 같은 것을 볼 수 있다.

그림 2-10은 viterbi decoder의 구조를 나타낸 것이다.

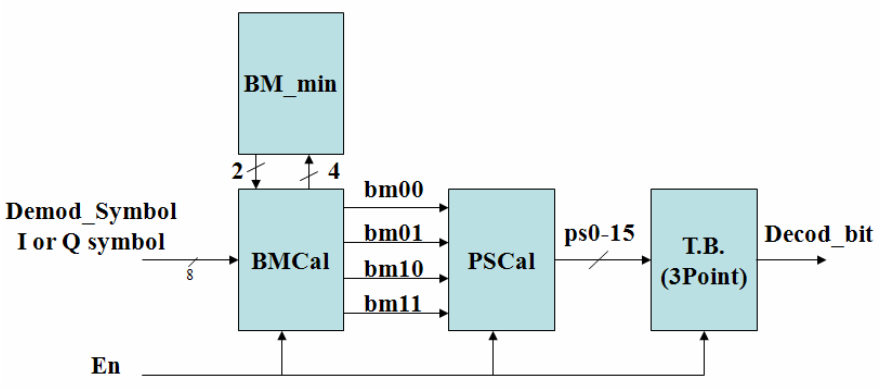


그림 2-10. Viterbi Decoder의 구조

Fig 2-10. The structure of viterbi decoder.

BMCal은 입력 데이터 심볼을 받아 BM을 계산하는 블록으로 하위 블로으로 BM_min이 있다. BM_min은 입력 데이터 심볼의 부호화 비트가 '0'과 '1'일 경우 가장 거리가 짧은 mapping 구간을 계산한다. BMCal에서 계산된 BM값은 PSCal로 입력된다. PSCal은 입력된 BM값들을 이용해 각 상태의 PM값을 계산하고, 그에 따른 PS값을 출력한다. T.B.에서는 PS값을 입력 받아 저장한 후 trace_back 과정을 거쳐 복호 비트를 출력하게 된다.

Viterbi decoding을 위해 우선 LUT(Look Up Table)을 만들었다. 다음 그림 2-11은 mapping 포인트를 이용한 LUT이다.

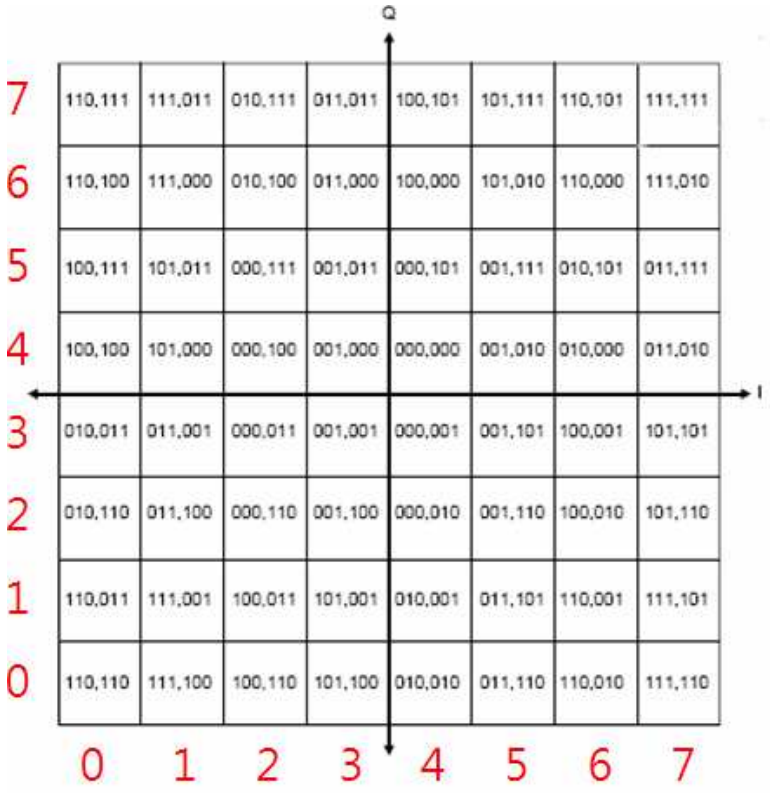


그림 2-11. Mapping 포인트를 이용한 LUT

Fig 2-11. The LUT using mapping point.

그림 11에서 I 채널 좌표 중 0,2,4,6의 짝수 번째 줄은 coded_bit C3가 ‘0’이고, 홀수 번째 줄은 ‘1’임을 볼 수 있다. 이를 이용하여 다음과 같은 7가지 절차로 decoding을 할 수 있다.

[1] BMCal에 입력된 신호는 coded_bit가 ‘0’ 일때 가장 가까운

좌표와 ‘1’일 때 가장 가까운 좌표를 찾는다.

[2] I 채널, Q 채널 각각 2개의 좌표를 저장한다.

[3] 입력된 신호로 BM00, BM01, BM10, BM11을 구한다.

[4] PS0~PS15를 구하여 3-point 알고리즘을 이용하여 decoding한다.

[5] decoding된 신호를 다시 re-encoding하여 그 값에 따라 절차 [2]에서 저장한 2개의 좌표 중 하나를 선택한다.

[6] 각 채널에서 선택된 좌표를 이용하여 그림 2-11을 이용하여 만든 LUT에서 uncoded_bit를 찾아낸다.

[7] Decoding된 신호는 differential decoding하여 원래의 데이터를 찾아낸다.

2.5 시뮬레이션 결과

그림 2-12는 64-QAM과 256-QAM 방식의 TCM 성능곡선이다.

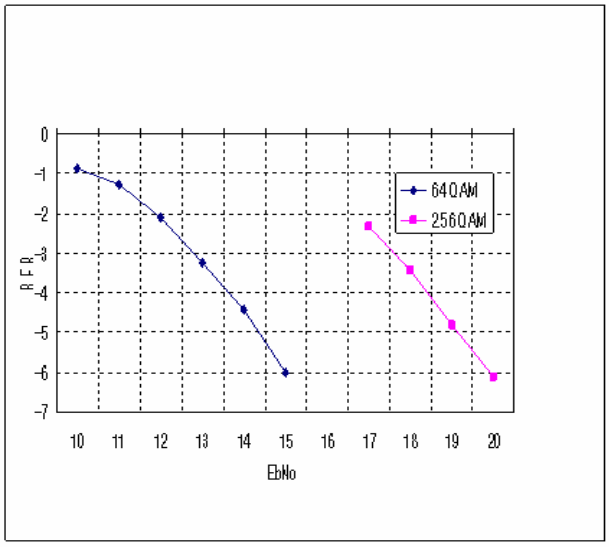


그림 2-12. 64-QAM과 256-QAM의 TCM 성능 곡선

Fig 2-12. The performance of TCM about 64-QAM and 256-QAM.

시뮬레이션에서 사용한 데이터는 64-QAM과 256-QAM 모두 10^6비트를 사용하였고, 채널 환경은 AWGN을 첨가하였다. 10^{-4}에서 약 5dB의 성능 차가 있음을 알 수 있다.

다음 그림 2-13은 64-QAM 일 때 CATV 전체 시스템의 성능을 시뮬레이션 한 것이다. 그림에서 알 수 있듯이 outer code인 TCM에 연접 부호로서 inner code에 RS 부호를 사용할 때 성능이 크게 향상됨을 알 수 있다.

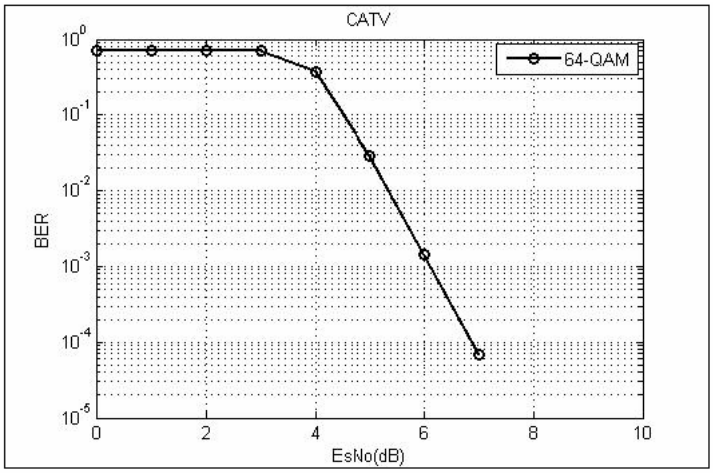


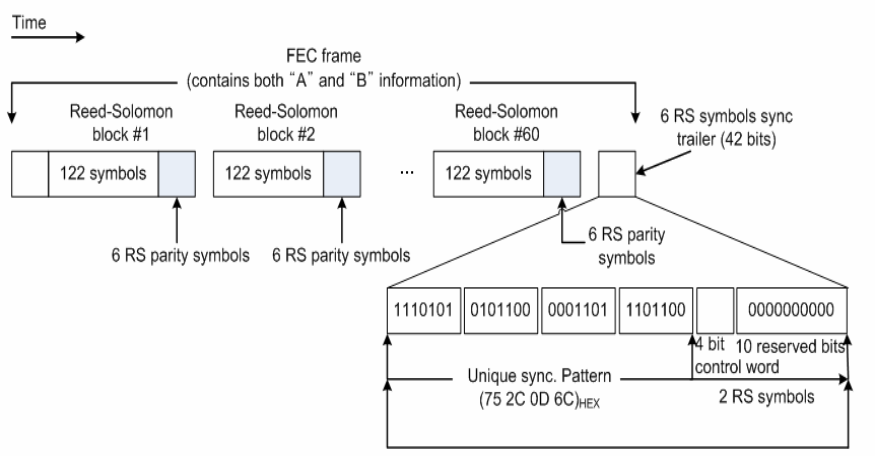
그림 2-13. 64-QAM모드에서의 전체 시스템 성능 곡선

Fig 2-13. The performance of total system about 64-QAM.

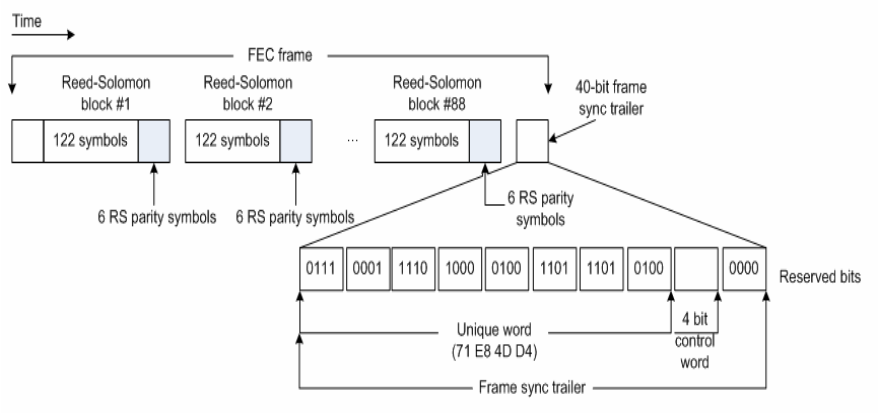
제 3 장 구현상의 문제점 및 해결 방법

3.1 프레임 동기 알고리즘

그림 3-1는 CATV 하향 스트림의 프레임 구조이며, 각 변조 방식 별 프레임 구조는 틀리고 그림 3-1(a)는 64-QAM 모드 일 때, 그림 3-1(b)는 256-QAM 모드 일 때의 프레임 구조이다.



(a) 64-QAM모드의 데이터 구조



(b) 256-QAM모드의 데이터 구조

그림 3-1. CATV 하향 스트림의 데이터 구조

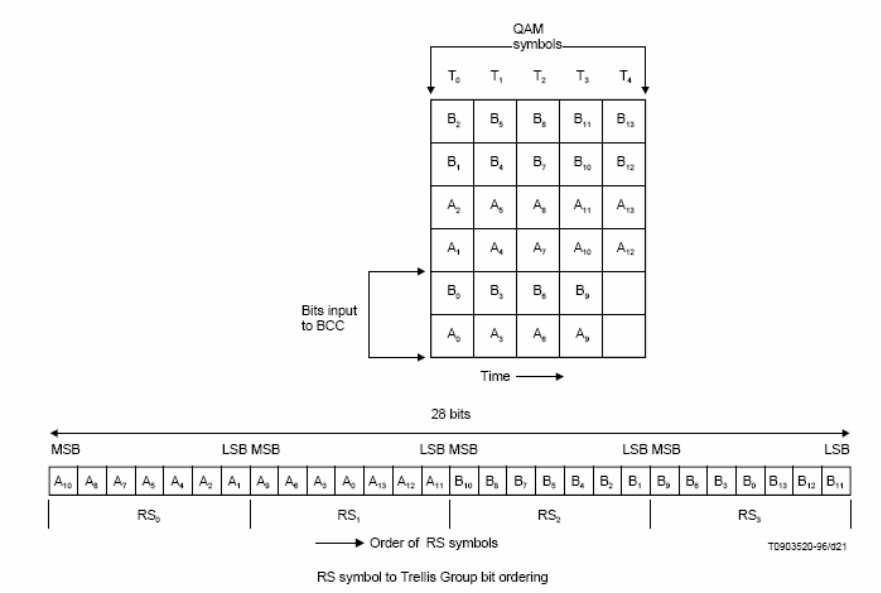
Fig 3-1. The structure of CATV down stream data.

64-QAM의 경우 한 프레임의 데이터는 60개의 RS 블록과 42 비트의 sync trailer로 구성되어있다. 42 비트의 sync trailer는 4개의 RS 심볼과 4비트의 interleaver control word, 10 reserved bit로 구성된다. 여기서 4개의 RS 심볼은 UW로 항상 일정하며, 이를 이용해 수신단에서 프레임 동기를 맞추게 된다. TCM 심볼과 RS 심볼을 구성하는 비트 수가 틀리므로 프레임 동기를 맞추지 못하면 수신단에서는 RS 심볼을 제대로 구성할 수 없다. 때문에 프레임 동기를 맞추는 것은 매우 중요하다.

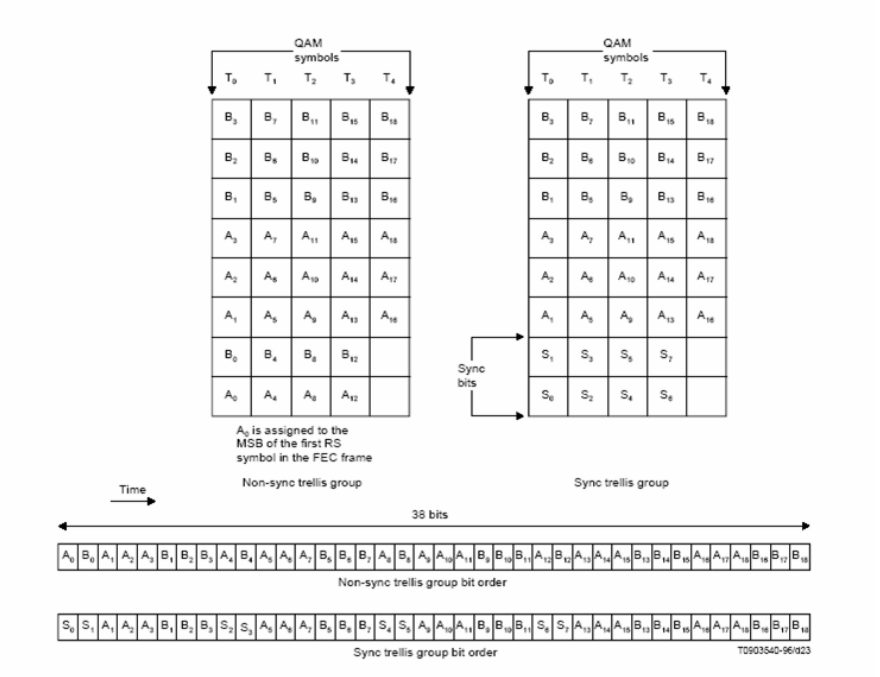
일반적으로 동기를 맞추기 위해 심볼을 비트 단위로 풀어내어 한

비트씩 UW와 비교하면서 동기를 맞추는 방법을 사용한다. 하지만 이 경우 64-QAM같은 경우 한 TCM 심볼당 6비트이므로 6배 빠른 클락이 필요하거나 굉장히 큰 메모리 용량이 필요하다. 또한 256-QAM의 경우 8배 빠른 클락 또는 그에 맞는 메모리가 필요하다.

그림 3-2는 RS 심볼을 TCM 심볼로 만들기 위한 데이터 오더링을 나타낸다.



(a) 64-QAM 데이터 오더링 패턴



(b) 256-QAM 데이터 오더링 패턴

그림 3-2. 데이터 오더링 패턴

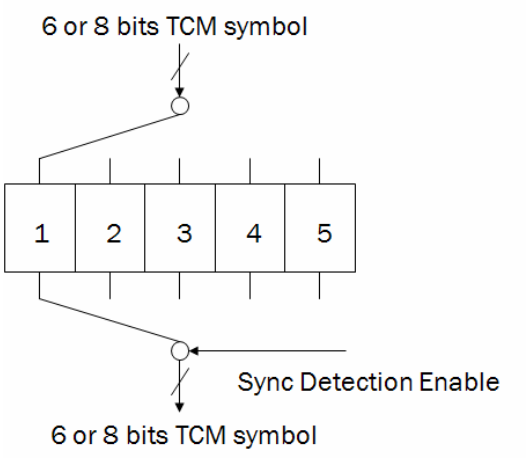
Fig 3-2. The pattern of data ordering.

64-QAM의 경우 4개의 RS 심볼, 즉 28bit의 데이터를 저장 후 오더링 패턴과 같이 5개의 TCM 심볼로 변화시킨다. 256-QAM도 마찬가지로 38bit의 데이터를 저장 후 5개의 TCM 심볼로 변화시킨다.

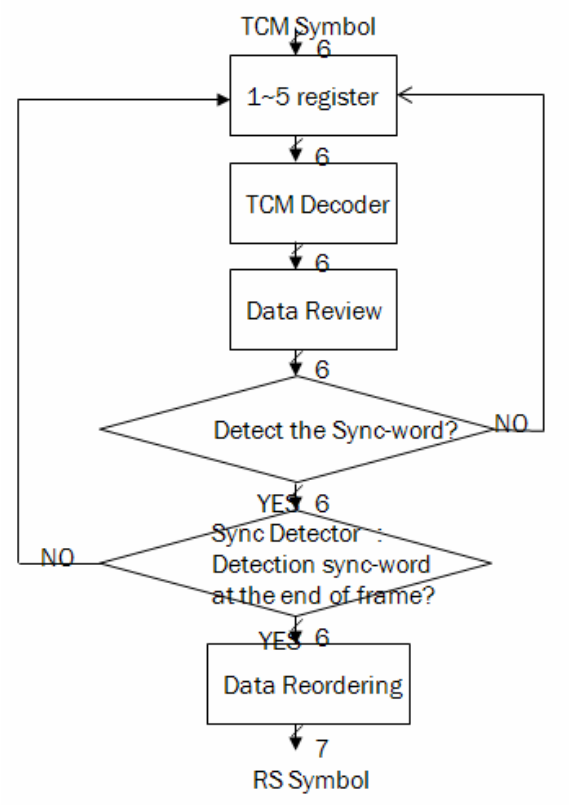
위에서 알 수 있듯이 심볼에서 심볼로 데이터가 변하기 때문에 채널 상에서 에러가 나더라도 비트천이 현상은 발생하지 않는다. 즉,

심볼내의 비트는 유지되어 진다. 이를 이용하여 데이터 프레임의 동기를 맞추면 심볼을 비트 단위로 풀어내지 않아도 충분히 구현이 가능하다. 그렇게 되면 비트 동작을 위한 클락은 필요 없어진다.

본 논문에서는 구현시 TCM decoder의 이전 단계에서 5개의 레지스터를 사용하였다. 그림 3-3은 사용한 레지스터와 UW detection의 flow-chart를 나타낸 것 이다.



(a) Unique Sync-word Detection을 위한 5개의 레지스터



(b) Sync Detection Flow-chart

그림 3-3. Sync Detection을 위한 레지스터와 Flow-chart

Fig 3-3. The registers and flow-chart for sync detection.

그림 3-3 (a)의 5개의 레지스터에 1번부터 순서대로 데이터가 저장된다. 저장된 데이터는 순서대로 TCM decoder로 출력된다. 그림 3-3 (b)의 flow-chart에서 보면 레지스터에서 출력된 데이터는 TCM decoder에 의해 decoding 되어진다. decoding된 데이터는 sync-word detector에 의해

판별되어지는데, 맨 처음의 데이터부터 시작해서 한 프레임의 데이터가 decoding 될 때까지 sync-word를 찾지 못하면 TCM decoder 전의 레지스터로 신호를 보내게 된다. 레지스터에 sync-word detector에서 신호가 들어오는 순간 레지스터의 출력 순서를 하나 건너뛰게 한다. 예를 들어 sync-word detector에서 신호를 보내는 순간 레지스터의 출력 순서가 3번 레지스터 였다면, 다음 클락에서는 4번 레지스터에서 출력이 되지 않고, 4번을 건너뛰고 5번 레지스터에서 출력 후 다시 순서대로 계속 출력을 하는 것이다. 이 방법은 그림 25의 데이터 오더링을 이용한 것으로 오더링 패턴에서 첫번째 TCM 심볼을 찾아 낼 수 있으면 sync-word detection을 할 수 있다는 것에서 감안한 것이다. 오더링 패턴에서 생성되는 TCM 심볼이 5개 이므로 5개의 레지스터를 사용하는 것이다. 이 방법을 사용할 경우 sync-word는 최대 5프레임의 데이터가 첫 detection에 필요하지만 5 프레임이 실제로 처리되는 시간은 극히 짧은 시간이므로 많은 클락과 계산량 없이 detection 할 수 있다.

처음 detection이 성공한 후 detection에 신뢰를 가지기 위해 최소 3~5 프레임 동안 정확히 프레임이 끝나는 지점에서 sync-word가 계속 detection되는 것을 확인하고, detection되지 않는다면 detecion과정을 처음부터 다시 수행한다. Detection이 성공하면 다시 데이터 오더링을

통해 TCM 심볼을 RS 심볼로 변환시킨 후 출력한다.

3.2 제한된 클락을 사용한 시스템 구현 기법

연접 부호의 구현일 경우 사용된 부호화 기법에 따라 각각의 클락이 필요하다. 그림 3-4은 CATV 하향 스트림 채널 코딩 시스템에서 각 블록마다 필요한 클락을 나타낸 것이다.

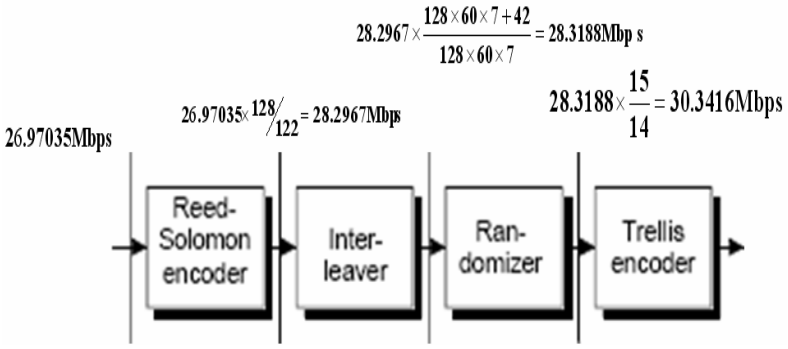


그림 3-4. 각 블록의 필요한 클락

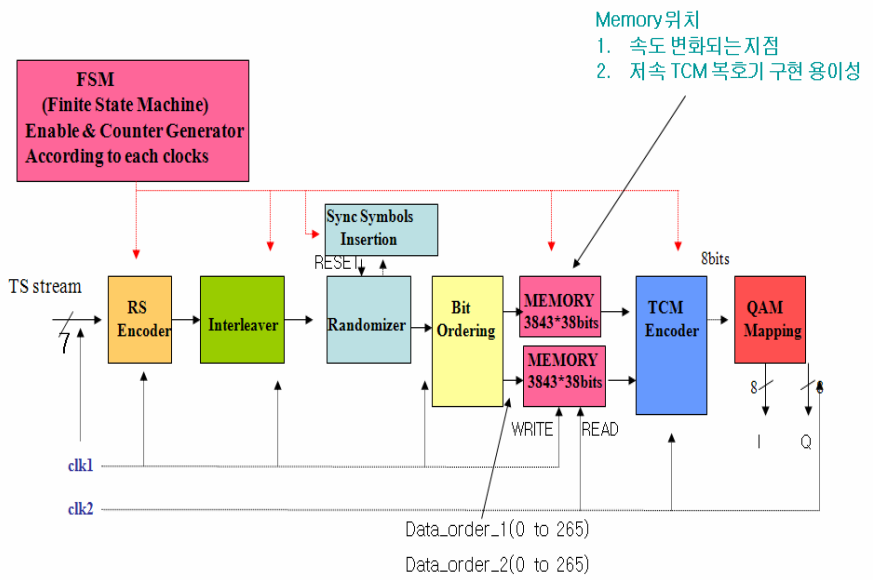
Fig 3-4. The necessary clock of each block.

RS 부호와 TCM의 경우 부호화율이 적용이 되기 때문에 입력과 출력에 필요한 클락이 틀려야 된다. 송신단에서는 입력보다 출력의 클락이 빨라야 입출력에 대한 타이밍을 맞출 수 있다. Randomizer의

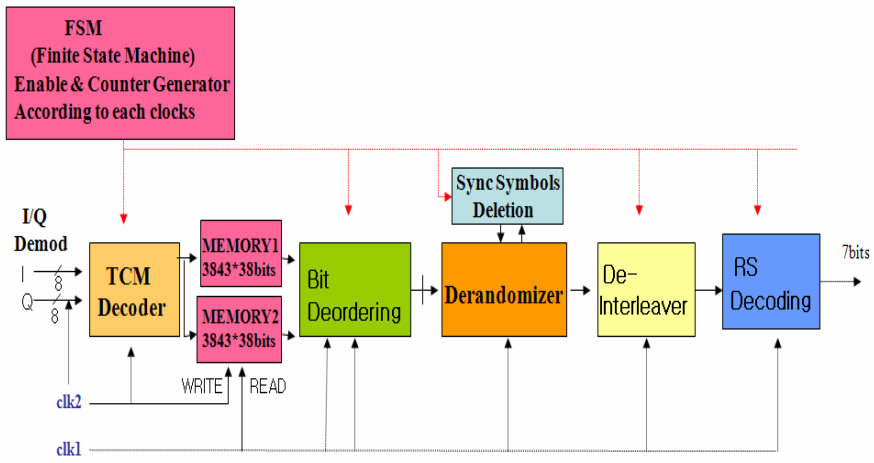
경우 출력단에서 데이터 프레임이 끝날 때 sync trailer를 첨가 하기 때문에 그에 대해 클락이 틀려진다. 그림 3-4에는 각 블록에서 출력시 필요한 클락 계산을 볼 수 있다. 수신단의 경우는 송신단과 반대로 입력의 클락이 출력의 클락보다 빨라야 함을 알 수 있다.

실제 시스템 구현시 각 블록에 대한 클락들을 맞춰서 적용시킬 수 없다. 그 이유는 각 클락의 동기가 맞아야 하기 때문이다. 각 클락간의 동기가 맞지 않으면 시스템은 오동작을 일으키게 된다. 기본적으로 제공되는 클락과의 동기를 맞추는 가장 기본적인 방법은 분주를 이용하는 것이다. 기본 클락을 분주를 시킬 수 있다면 클락의 동기는 문제가 되지 않는다. 하지만 그림 3-4에서 알 수 있듯이 각 블록에서 필요한 클락은 분주가 불가능하다. 그렇기 때문에 클락이 제한되어 있으면 시스템을 구현하기 힘들다.

이러한 문제를 해결하기 위해 그림 3-5과 같이 두 개의 메모리를 사용하였다.



(a) 송신단 시스템구조



(b) 수신단 시스템 구조

그림 3-5. 구현시 송수신 시스템 구조

Fig 3-5. The structure of TX and RX system for implementation.

‘clk1’을 시스템의 기본 클락이라 하면 ‘clk2’는 기본 클락의 4분주시킨 클락이다. 메모리의 위치는 RS 심볼과 TCM 심볼이 서로 변환되는 위치이다. RS 부호, interleaver, randomizer는 RS 심볼을 사용하고, TCM은 TCM 심볼을 사용한다.

메모리 사이즈가 3843x38인 이유는 64-QAM으로 동작 할 때 한 프레임이 데이터 오더링 패턴에 정확히 떨어지지 않고, 두 프레임이 될 때 오더링 패턴과 일치되기 때문에 메모리 하나 당 64-QAM 데이터 두 프레임을 저장할 수 있도록 하기 위해서이다.

RS 부호, interleaver, randomizer와 같이 ‘clk1’에 의해 동작하는 부분을 part1이라하고, TCM과 같이 ‘clk2’에 의해 동작하는 부분을 part2라고 하자.

그림 3-5에서 볼 수 있듯이 우선 part1은 ‘clk1’에의해 동작을 하고, 첫번째 메모리에 저장이 된다. 첫번째 메모리에 모두 저장이 된 후 두번째 메모리에 저장을 시작할 때 ‘clk2’에 의해 첫번째 메모리에서 읽기 동작이 시작된다. 첫번째 메모리에서 데이터를 읽어내면 part2가 동작한다. ‘clk1’이 두번째 메모리에 모두 저장을 끝마쳐도 ‘clk2’의 첫 번째 메모리 동작은 끝나지 않는다. 이 때 ‘clk2’의 첫 번째 메모리 동작이 끝날 때까지 part1은 동작을 하지 않고 part2의 동작이 끝날

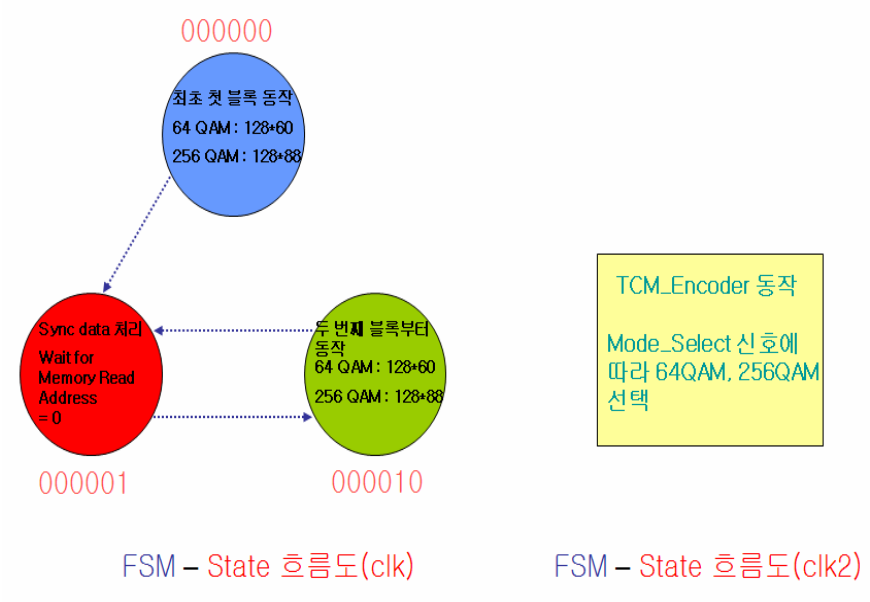
때까지 기다린다. ‘clk2’에 의해서 part2가 첫번째 메모리의 동작이 끝나면 두번째 메모리 동작을 시작하고, ‘clk1’에 의해 part1은 다시 첫번째 메모리 동작을 시작한다. 이러한 절차로 part1은 송신단에서는 메모리에 번갈아가며 데이터를 저장하고, 수신단에서는 데이터를 읽어낸다. 마찬가지로 part2는 송신단에서는 읽기 동작을 수신단에서는 데이터를 저장한다. 이러한 방법에 의해 기본 클락 하나를 이용하여 시스템을 구현할 수 있다.

제 4 장 FPGA 구현 결과

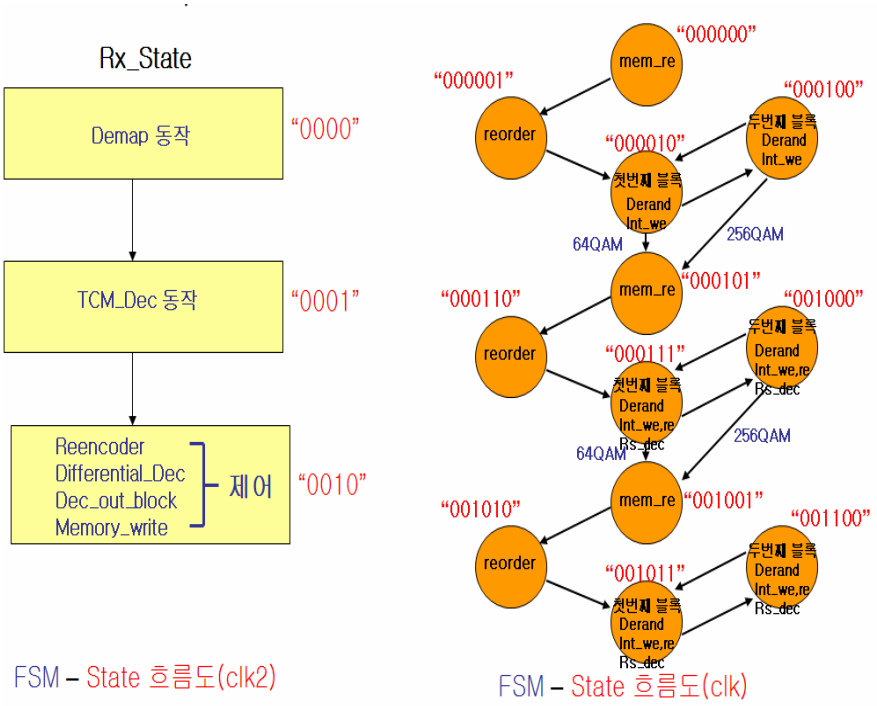
제 2장과 제 3장에서 구현시 필요한 알고리즘과 시스템 구현시 문제점, 그리고 그 해결 방법에 대하여 알아보았다. 제 4장에서는 이것을 토대로 실제 구현하기 위해 timing simulation을 하였고 이를 FPGA 칩에 구현하였다.

4.1 채널 부호화 방식의 FSM(Finite State Machine)

본 논문에서 구현에 사용한 전체적인 컨트롤은 FSM(Finite State Machine)을 사용하였고, 그림 4-1는 그 흐름도를 나타낸 것이다.



(a) 송신단에서의 FSM State 흐름도



(b) 수신단에서의 FSM State 흐름도

그림 4-1. 송수신단의 FSM State 흐름도

Fig 4-1. The FSM state flow-chart of TX and RX.

구현시 FSM을 사용한 이유는 전체적인 컨트롤이 용이하기 때문이며, 이는 모든 VHDL의 component의 top 모델에서 제어를 한다.

그림 4-1 (a)는 송신단에서의 FSM state에 대한 것이다. ‘clk’사용에서 상태 값이 “000000” 일 때 최초 첫 데이터 프레임을 처리하고, 첫 프레임이 처리된 후 sync trailer 삽입을 위해 “000001”로 바뀐다. Sync

trailer를 삽입하면 “000010”으로 상태 값이 바뀌면서 두 번째 프레임이 처리되며, 이 상태에서 메모리의 데이터를 TCM으로 출력하게 된다. 두 번째 프레임 처리 후 다시 상태 값이 “000001”로 되며 sync trailer를 삽입한다. 이후 “000010”과 “000001”의 상태 값이 계속 반복된다. ‘clk2’사용에서는 메모리에서 연속적으로 데이터가 출력되기 때문에 상태 값의 변화가 없다.

그림 4-1(b)는 수신단의 FSM state에 대한 것으로, ‘clk2’사용시에는 상태 값 “0000”에서 TCM 복호기 이전의 sync detection을 위한 레지스터 사용 컨트롤 신호가 처리된다. “0001”에서 TCM 복호기의 컨트롤 신호가 처리되며, TCM 복호기의 출력이 나오면 “0010” 으로 상태 값이 변하여 ‘clk2’를 사용하는 나머지 component에 대한 컨트롤 신호를 처리하고, 이 때 데이터는 연속적으로 처리되므로 상태값 반복은 필요하지 않다. ‘clk’에서는 메모리 출력과 데이터 오더링 후 상태 값 “000010”과 “000100”을 이용하여 derandomizer를 동작 시키고, deinterleaver에서 메모리의 데이터 저장을 실행한다. 그 후 다시 메모리 출력과 데이터 오더링 후 상태 값 “000111”과 “001000”을 이용하여 derandomizer와 deinterleaver 메모리 데이터 저장을 실행한다. 이 때, 메모리가 다 채워지는 시점이 있기 때문에 그 시점부터는 deinterleaver 출력을 같이 하게 된다. 그 후 “001001”에서부터 “001100”까지의 상태

값이 반복되면서 ‘clk’의 복호단이 실행된다.

그림 4-1에서 송신단에 비해 수신단의 상태 변화가 복잡한 이유는 UW detection과 deinterleaver에 의해 상태의 가지 수가 많아 졌기 때문이다.

4.2 각 알고리즘의 구현 결과

제 2장에서 알아본 알고리즘 각각을 VHDL을 이용하여 타이밍 시뮬레이션을 했고, 제 3장에서의 구현상의 문제점에 대한 해결책까지 VHDL을 이용하여 시뮬레이션 하였다. 이렇게 CATV 전체 시스템에 대하여 시뮬레이션 한 후 FPGA칩을 이용하여 구현하였다. 시스템 구현시 사용한 시뮬레이션 툴은 Xilinx 사의 modelsim을 이용하였다.

그림 4-2은 기존의 (127,122) RS encoder를 VHDL로 타이밍 시뮬레이션 한 것이다.

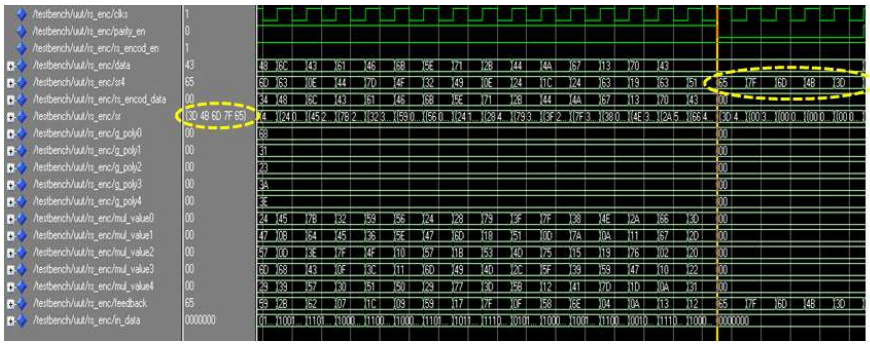


그림 4-2. RS Encoder 타이밍 시뮬레이션

Fig 4-2. The timing Simulation of RS E\encoder.

입력 심볼이 모두 rs_encod_data로 출력된 후 sr4의 데이터가 5 클럭에 거쳐 순서대로 출력됨을 볼 수 있다.

그림 4-3은 extended RS encoder를 VHDL로 타이밍 시뮬레이션 한 것이다. 카운터가 127이 될 때 sum의 값이 출력되어진다.

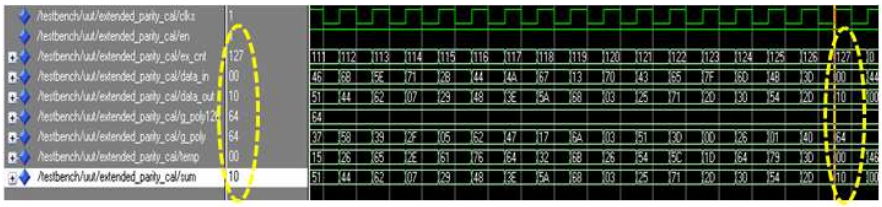


그림 4-3. Extended 심볼 생성 타이밍 시뮬레이션

Fig 4-3. The timing simulation of extended symbol generating.

그림 4-4은 RS decoder를 VHDL로 구현하여 timing simulation 한 것이다. 복호되어 출력된 데이터가 부호화 되기 전의 데이터와 일치되는 것을 알 수 있다.

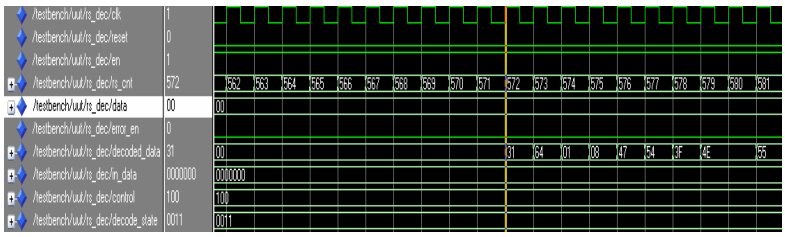


그림 4-4. RS Decoder 타이밍 시뮬레이션

Fig 4-4. The timing simulation of RS decoder.

그림 4-5는 interleaver의 timing simulation이다. 입력 데이터는 RS encoder에서 encoding된 데이터를 사용하였다.

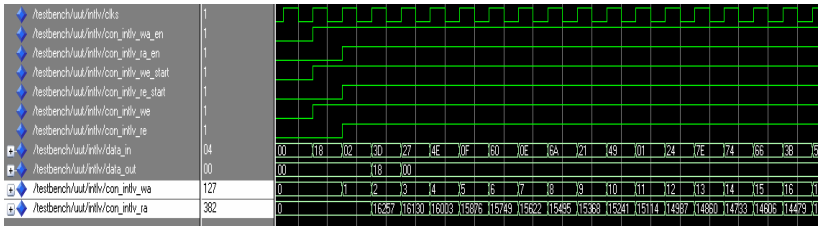


그림 4-5. Convolutional Interleaver 타이밍 시뮬레이션

Fig 4-5. The timing simulation of convolutional interleaver.

그림 4-6은 randomizer를 VHDL로 구현한 것이다. 입력 데이터는 interleaver의 출력 데이터를 사용하였다.

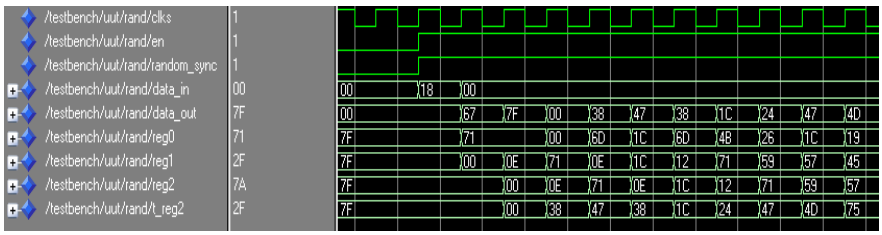


그림 4-6. Randomizer 타이밍 시뮬레이션

Fig 4-6. The timing simulation of randomizer.

그림 4-7와 그림 4-8는 differential encoder와 decoder를 VHDL로 구현하여 시뮬레이션 한 것이다. 입력 데이터는 randomizer의 출력 데이터에서 데이터 오더링 한 후 coded_bit를 입력으로 사용하였다.

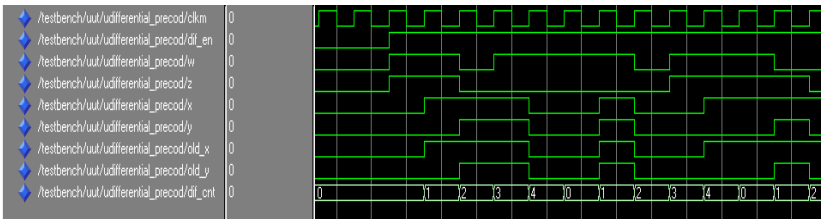


그림 4-7. Differential Encoder 타이밍 시뮬레이션

Fig 4-7. The timing simulation of differential encoder.

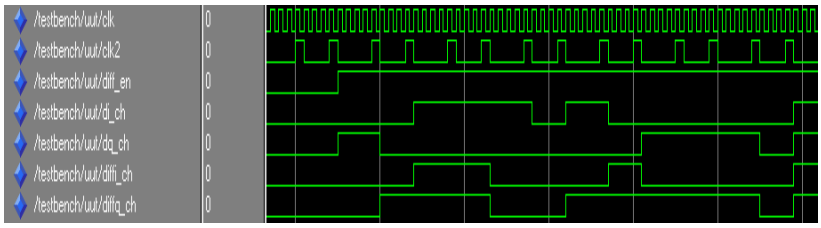


그림 4-8. Differential Decoder 타이밍 시뮬레이션

Fig 4-8. The timing simulation of differential decoder.

그림 4-9은 convolutional encoder의 timing simulation이다. 입력 데이터는 differential encoder의 출력 데이터를 사용하였다.

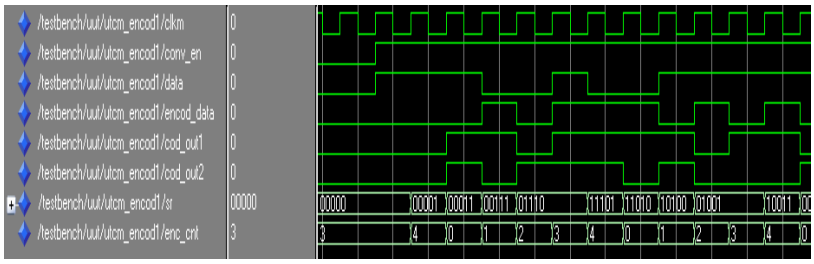


그림 4-9. Convolutional Encoder 타이밍 시뮬레이션

Fig 4-9. The timing simulation of convolutional encoder.

그림 4-10에서 convolutional encoder의 출력 데이터와 randomizer의

출력데이터 중 데이터 오더링에서의 uncoded_bit를 심볼로 만들어 64-QAM의 Mapping을 거친 신호를 볼 수 있다.

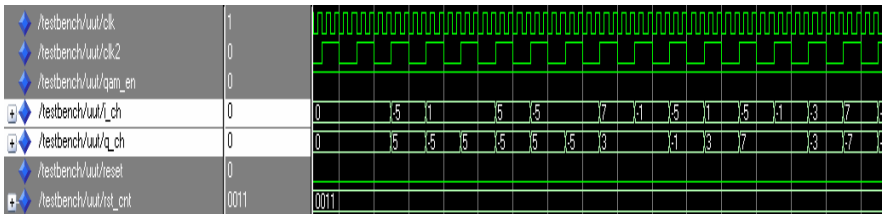


그림 4-10. 64-QAM Mapping 신호.

Fig 4-10. The mapping signal of 64-QAM.

그림 4-11와 그림 4-12, 그림 4-13은 Viterbi Decoder에서 BMCal, PSCal, Trace_back을 VHDL로 구현하여 timing simulation한 것이다. 입력 데이터는 그림 4-10에서 mapping된 데이터를 8bit soft decision하여 사용하였다.

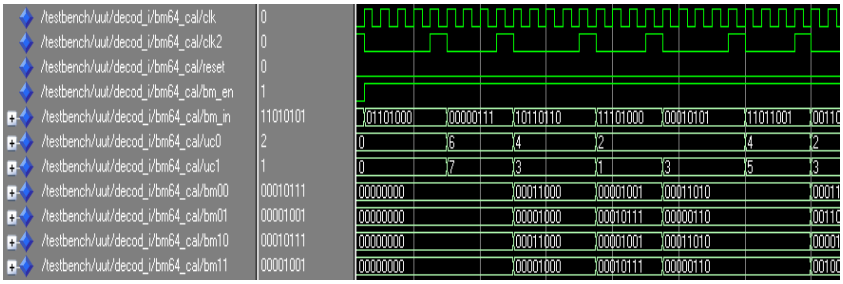


그림 4-11. Viterbi Decoder BMCal 타이밍 시뮬레이션

Fig 4-11. The timing simulation of BMCal in viterbi decoder.

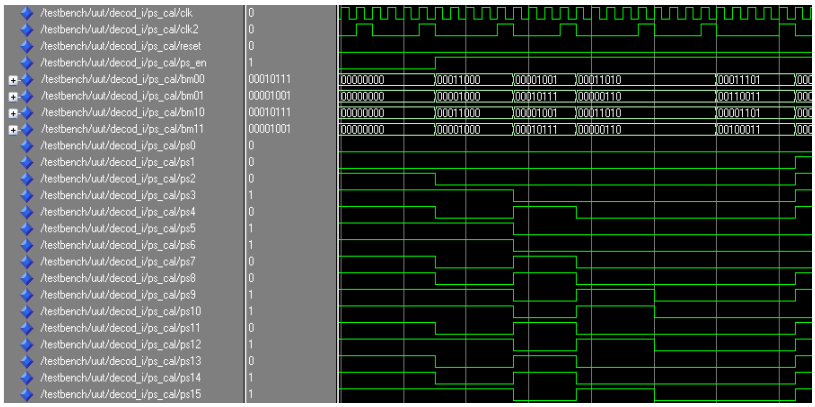


그림 4-12. Viterbi Decoder PSCal 타이밍 시뮬레이션

Fig 4-12. The timing simulation of PSCal in viterbi decoder.

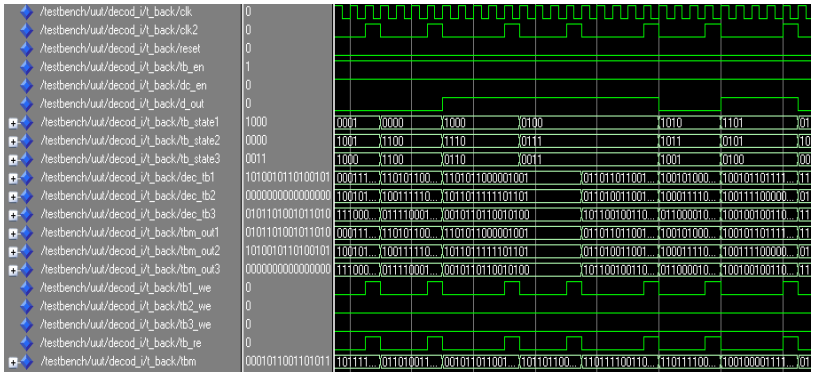


그림 4-13. Viterbi Decoder Trace_back 타이밍 시뮬레이션

Fig 4-13. The timing simulation of trace_back in viterbi decoder.

그림 4-14은 프레임 동기를 맞추는 것을 나타낸 것 이다.

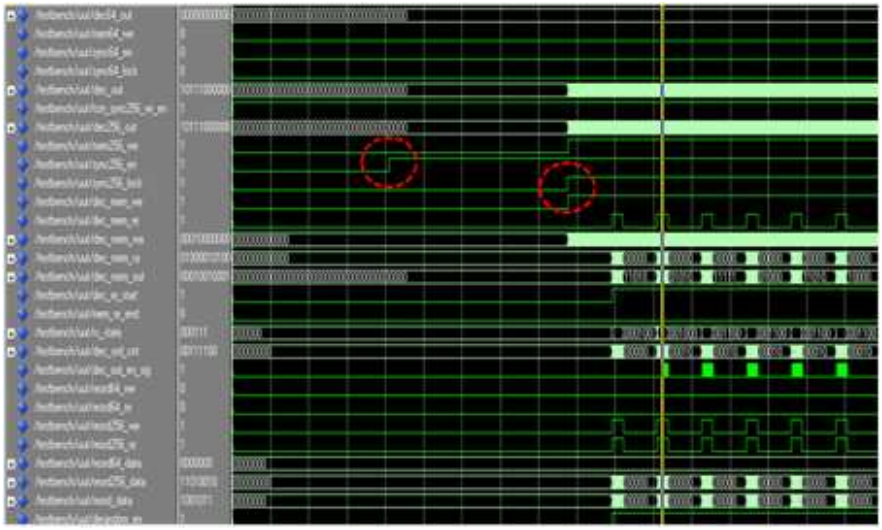
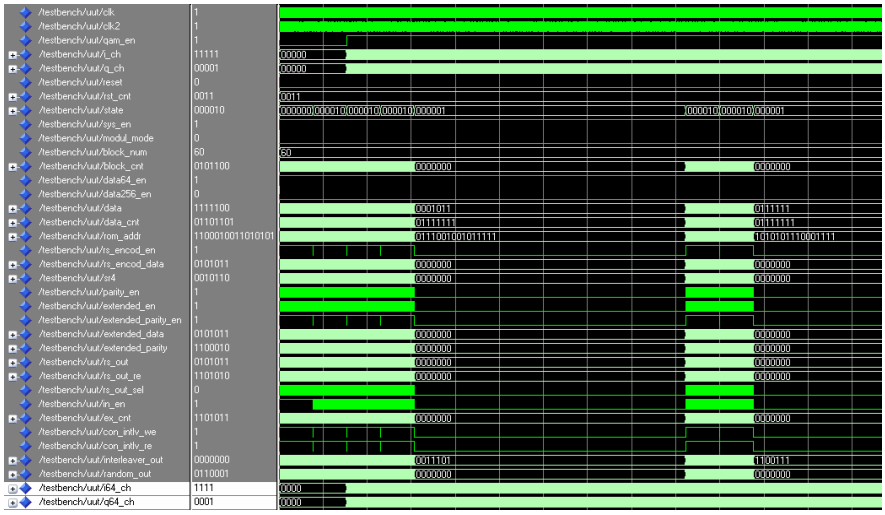


그림 4-14. Sync-word 디텍션

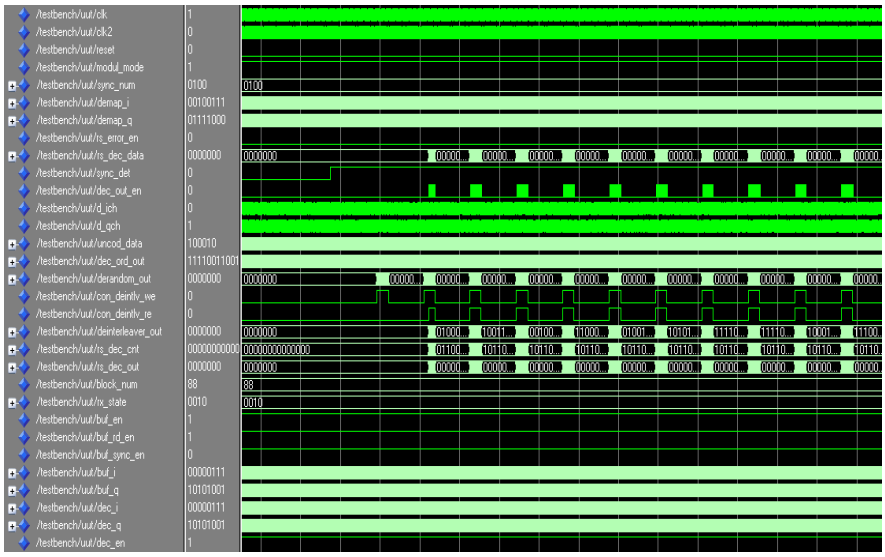
Fig 4-14. The detection of sync-word.

제 3장에서의 구현시 문제점 중 UW detection에 대한 것으로 최초의 sync-word를 찾았을 때 Sync_en의 값이 ‘1’이 되었고, 4프레임 동안 검사 후 sync_lock의 값이 ‘1’이 되며 sync detection에 대해 신뢰성을 가지게 되어 오더링 후 RS 심볼을 출력하는 것을 볼 수 있다.

그림 4-15는 송수신단의 전체적인 timing simulation이다. 그림 4-15에서 RS 심볼 데이터로 동작하는 부분에서 메모리 동작의 상태에 따라 연속적인 동작이 이루어 지지 않음을 볼 수 있고, TCM 심볼 데이터로 동작하는 부분은 데이터가 연속적으로 처리됨을 볼 수 있다.



(a) 송신단 전체 타이밍 시뮬레이션



(b) 수신단 전체 동작 타이밍 시뮬레이션

그림 4-15. 송수신단의 전체 타이밍 시뮬레이션

Fig 4-15. The total timing simulation of TX and RX.

4.3 최종 구현 결과

구현에 사용한 FPGA 칩은 ‘Vertex II Pro xc2vp30-5’이고 그림 4-16와 같다.

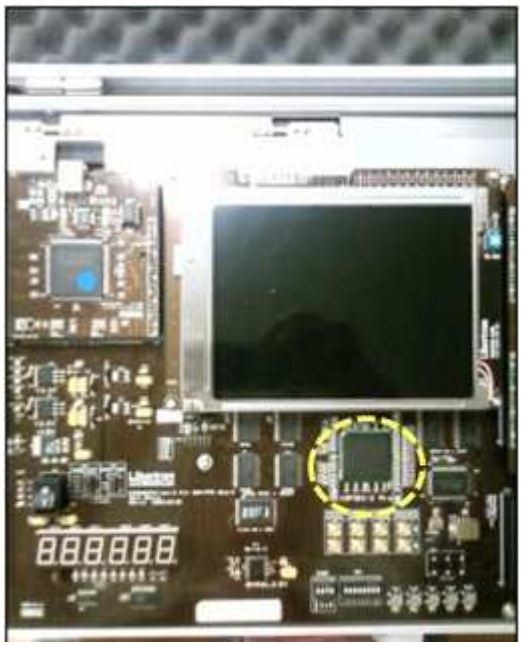


그림 4-16. 구현에 사용한 FPGA 칩(Vertex II Pro xc2vp30-5)

Fig 4-16. The FPGA chip (Vertex II Pro xc2vp30-5) used implementation.

이 FPGA 칩은 총 30만 게이트가 지원된다. FPGA 구현 툴은 ISE 7.1을 사용하였다.

구현의 결과로 그림 4-17와 같이 로직 아날라이저로 값을 확인하였다.

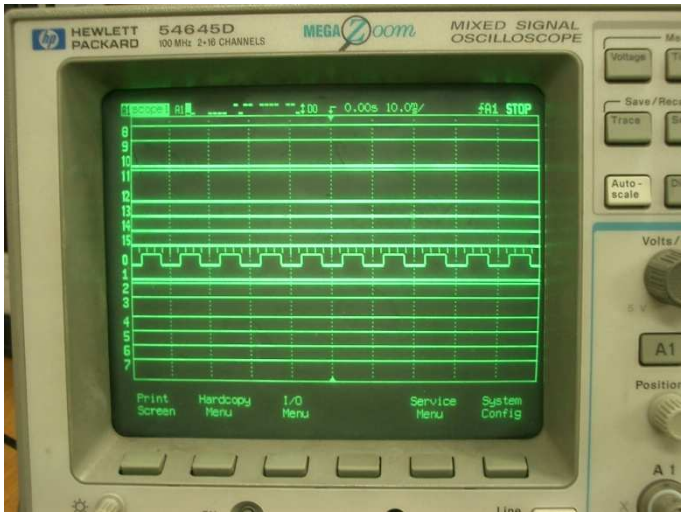


그림 4-17. 로직 아날라이저로 구현 결과 확인

Fig 4-17. The result of measurement at the logic analyzer.

다음 표 4-1은 구현시 FPGA 칩사용에 관한 내용이다.

표 4-1. FPGA 칩 구현시 사용된 게이트에 관한 차트

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops:	6,468	27,392	23%	
Number of 4 input LUTs:	15,667	27,392	57%	
Logic Distribution:				
Number of occupied Slices:	10,650	13,696	77%	
Number of Slices containing only related logic:	10,650	10,650	100%	
Number of Slices containing unrelated logic:	0	10,650	0%	
Total Number 4 input LUTs:	16,909	27,392	61%	
Number used as logic:	15,667			
Number used as a route-thru:	1,242			
Number of bonded IOBs:	146	416	35%	
Number of PPC405s:	0	2	0%	
Number of Block RAMs:	46	136	33%	
Number of GCLKs:	2	16	12%	
Number of GTs:	0	8	0%	
Number of GT10s:	0	0	0%	

구현시 총 사용된 슬라이스 수는 10,650개이고, 전체의 77%를 사용하였다. 메모리 또한 총 33%를 사용하여 충분한 구현 능력을 보여주고 있다.

다음 표 4-2는 구현시 클락 성능에 대한 것을 나타낸다.

표 4-2. 구현시 클락 성능

Performance Summary

Property	Value
Number of Unrouted Signals:	All signals are completely routed.
Number of Failing Constraints:	0

표 4-2에서 알 수 있듯이 디바이스에 주어진 기본 클락 사용에 대하여 계산량 초과가 생기지 않았다.

제 5 장 결 론

본 논문에서는 ITU-T J38 CATV 하향 스트림 채널 코딩 시스템을 구현하여 각 블록에 대한 알고리즘과 timing simulation을 보였고, 이러한 연접 부호 시스템을 구현할 때의 문제점과 이를 극복할 수 있는 구현 기법을 제시하였다.

ITU-T J38 CATV 하향 스트림 채널 코딩 시스템은 RS 부호, Interleaver, Randomizer, TCM(64-QAM, 256-QAM)으로 구성되어 있다. RS 부호는 기존의 알고리즘에서 Extended 심볼을 추가하여 에러 정정 능력을 2에서 3으로 향상 시킨 Extended RS 부호를 사용하였다. Interleaver는 Convolutional Interleaver를 사용하였는데, 구현시 많은 레지스터를 사용하게 되면 계산량이 많아지고 하드웨어의 처리에 부하가 많이 걸리는 문제가 있다. 이를 극복하기 위해 메모리를 사용하여 레지스터 사용과 같은 효과를 나타내었다. 또한 데이터의 DC 신호 방지를 위해 Randomizer를 사용하였고, 64-QAM, 256-QAM의 두가지 모드를 사용하는 TCM을 구현하였다.

이러한 시스템에서 수신단에서의 데이터 프레임 동기를 맞추기 위해 5개의 레지스터를 TCM Decoder 전단에 사용하여, 이를 이용해 프레임 동기를 맞추었다. 이 방법을 사용함으로서 심볼을 비트 단위로 처리하지 않아 불 필요한 클락 사용을 방지할 수 있었다. 또한, 이렇게 각 블록마다 다른 클락이 필요한 시스템을 구현하기 위해 심볼간의 변환이 있는 곳에 두 개의 메모리를 사용함으로서 기본 클락하나를 이용한 구

현을 하였다.

구현시 전체 컨트롤은 FSM(Finite State Machine)을 사용하였고, 구현에 사용된 FPGA칩은 Xilinx사의 ‘Vertex II pro xc2vp30-5’을 사용하여, 측정 결과를 로직 아날라이저로 확인하였다. 구현 결과 칩 용량에 충분한 슬라이스를 사용하여 무리가 없었고, 또한 디바이스에서 제공되는 클락에 대해 계산량 초과가 생기지 않은 성공적인 구현을 하였다.

마지막으로 본 논문에서는 Interleaver의 모드를 128*1만을 구현했지만 앞으로, 표준에 나와있는 다른 모드(64*2, 32*4, 16*8, 8*16등)를 사용할 수 있도록 구현하는 것과, 메모리를 더 줄일 수 있는 방안이 필요하다. 아울러 내부 부호에 적용된 TCM, 외부 부호에 적용된 RS 부호보다 성능이 우수한 LDPC(Low Density Parity Check) 또는 Turbo 부호를 적용하는 방안이 과제이다.

참고 문헌

[1] G. C. Clark, Jr. and J.B. Cain, ”Error-Correction Coding for Digital Communications”, Plenum Press, 1981.

[2] “Digital Multi-programme Systems for Television, Sound and Data Services for Cable Distribution”, ITU-T J38

[3] S. Lin and D. Costello, Jr., “Error Control Coding : Fundamentals and Applications”, Prentice-Hall, 1983

[4] ITU-T Recommendation H. 222.0 (1995) | ISO/IEC 13818-1 : 1996, Information technology – Generic coding of moving pictures and associated audio information : Systems.

[5] ITU-R Recommendation BO. 1211 (1995), *Digital multi-programme emission systems for television, sound and data services for satellites operating in the 11/12 GHz frequency range*.

[6] G. Ungerboeck, “Trellis Coded Modulation with Redundant Signal Sets, Part I : Introduction” , IEEE Communications Magazine, Vol. 25, No. 2, pp. 5-11, February 1987

[7] G. Ungerboeck, “Trellis Coded Modulation with Redundant Signal Sets, Part II : Introduction” , IEEE Communications Magazine, Vol. 25, No. 2, pp. 12-21, February 1987

[8] Dojun Rhee and Robert H. Morelos-Zaragoza, “Error Performance Analysis of a Concatenated Coding Scheme with 64/256-QAM Trellis Coded Modulation for the North American Cable Modem Standard”, ISIT 1998, Cambridge, MA, USA, August 16-August 21.

[9] W. J. Weber, “Differential Encoding for Multiple Amplitude and Phase Shift Keying Systems”, IEEE Trans. Comm, vol. COM-26, no. 3, pp. 385-391, March 1978.

[10] Carden, Frank, "A Quantized Euclideans soft Decision Maximum Likelihood Sequence Decoder : A Concept for Spectrally Efficient TM Systems", Proceedings of the International Telemetering Conference, Vol. XXIV, pp. 375-384, October 1998.

[11] Carden, Frank, and Brian Kopp, “A Quantized Euclideans soft Decision Maximum Likelihood Sequence Decoder of TCM”, IEEE Military Communications Conference, Vol. 2, pp. 279-682, October 1988.

[12] 김재홍, “확장된 리드-솔로몬 부호의 오증 연산방법 및 복호 방법”, 대한 민국 특허청 공개 특허 특2000-0024730