國 立 交 通 大 學

電信工程學系

碩 士 論 文

關斯合併及增量冗餘混合式自動重傳機制之性能分析

Performance Analysis of Hybrid ARQ with Chase Combining and Incremental Redundancy

研 究 生：龔炳全

指導教授：蘇育德 博士

中 華 民 國 96 年 7 月

關斯合併及增量冗餘混合式自動重傳機制之性能分析

研究生：龔炳全　　　　　　　　　　　　指導教授：蘇育德 博士

國立交通大學電信工程學系碩士班

中文摘要

對於使用封包交換的無線網路來說，利用增量冗餘(Incremental redundancy)或關斯合併(Chase-combining)的混合式重傳機制在控制錯誤的系統中是非常有效率的。相較於傳統的自動重傳機制，他們通常能提供較好的錯誤比率效果並且有較高的吞吐量。

在本文中，我們針對這幾種協定衍伸出的一些適應性調變系統分析了他們的吞吐量和延遲效能。在我們的系統裡面，編碼和調變機制主要都是根據 IEEE 802.16e 的規格，採用的分別是迴旋渦輪碼(convolutional turbo code)以及四相位移鍵訊號(QPSK)、16 正交振幅調變(16QAM)和 64 正交振幅調變(64QAM)。我們考慮了加法性白色高斯雜訊(AWGN)和平坦瑞利衰落(flat Rayleigh fading)的環境。而在分析中，我們需要利用計算多維生成函數(generating function)來描繪轉換域(transform domain)的隱馬爾可夫程序(hidden Markov process)。我們提供了一些數值例子來展現並比較這兩種機制的效果，一般而言，增量冗餘的方式在兩種通道中都有較佳的性能比現。

Performance Analysis of Hybrid ARQ with Chase Combining and Incremental Redundancy

Student : Ping-Chuan Kung Advisor : Yu T. Su

Department of Communications Engineering

National Chiao Tung University

Abstract

Incremental redundancy (IR) or Chase-combining (CC) based hybrid ARQ (HARQ) protocols are very efficient error-control schemes for packet-switching wireless networks. With proper design, they outperform other ARQ protocols in both latency and throughput.

In this thesis we analyze the throughput and delay performance of several variations of these protocols with adaptive modulation. The coding and modulation schemes used in our system are primarily based on the IEEE 802.16e standard, i.e., convolutional turbo code (CTC), QPSK, 16QAM, and 64QAM, respectively. Both AWGN and flat Rayleigh fading environments are considered. Our analysis calls for the evaluation of the multi-dimensional generating function that characterizes the transform domain behavior of the underlying hidden Markov process. Numerical examples are provided for assessing the two classes of protocols. It is shown that, as far as performance is concerned, IR is a better choice although CC is easier to implement.

誌　　謝

首先得感謝我的指導教授 蘇育德博士這兩年來不只在研究上的敦敦教誨，使得此篇論文能更加順利的完成，讓我在通訊領域上有更加深入的了解，並且在人生的道路上給予我適時的指引讓我不至於迷失人生的方向。感謝口試委員蘇賜麟教授、陸曉峯教授、吳文榕教授以及王蒞君教授給予的寶貴意見，以補足這份論文上的缺失與不足之處。另外也要感謝實驗室的學長姐、同學以及學弟妹的幫忙還有鼓勵，讓我不僅在學習的過程中獲益匪淺，同時也為這兩年的生活增添了許多色彩。

最後，我更要感謝一直關心我、鼓勵我的家人以及朋友，沒有他們在背後的支持我無法這麼順利的完成論文，也因為有他們，使得我在繁忙的論文書寫中不時能浮現一張張的笑臉，給予我繼續向前的動力與勇氣，僅獻上此論文代表我最深的敬意。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ever-increasing demands on the quality, rate, and service choices of wireless information have stimulated the rapid development of wireless communication technologies and deployments of various wireless systems. Throughput, latency and error rate are the major performance and service quality concerns. These three performance measures, however, are not entirely independent. In a wireless packet-switching network, the correctness of each packet has to be proved before being mapped to upper layer for further processing. To meet the error rate requirement, an error-control mechanism has to be in place, which will reduce the throughput performance. On the other hand, better error rate performance often lead to lower latency because of less retransmission requests.

An error-control method called hybrid ARQ (automatic repeat request) that combines forward-error-correcting (FEC) codes with conventional cyclic redundancy check (CRC) code based ARQ [1] offers a higher reliability and throughput than those provided by pure FEC or CRC only [2]. A received packet is first verified by CRC and, if fails, the FEC decoder will try to correct the errors. Retransmission is requested only if the decoder is not able to correct the errors. System throughput can be enhanced if the FEC code structure is such that it can be decomposed into several parts with each part either self-decodable or combined-decodable. With the special FEC code structure, one needs not to transmit the complete encoded packet; instead, each part of a codeword can be transmitted successively if necessary. In other words, when a decoding failure is declared on a received packet which contains partial codeword only, the retransmitted packet shall be an incremental part of the original codeword such that either the incremental part or the combined parts can be decoded. Such an ARQ protocol is called incremental redundancy (IR) or Type II (or III if each part is self-decodable) hybrid ARQ.

Both types of hybrid ARQs can be considered as adaptive coding schemes. Further improvement can be obtained if the modulation used is also adapted to the channel condition. Such an adaptive modulation and coding scheme that combines Link Adaptation (LA) with IR is called Link Quality Control (LQC) in the enhanced general packet radio service (EGPRS) system. In this scheme, information is first sent with minimum coding, using high-order modulation and low rate coding schemes. This yields a high bit-rate if decoding is immediately successful. If decoding fails, additional coded bits (redundancy) are sent, using lower-order modulation and higher rate coding schemes, until decoding is successful. The more coded bits that have to be sent, the lower the resulting throughput.

Another technique to improve the retransmission performance called Chase Combining (CC) is through the combining of the received samples or the soft values associated with the same coded bit or symbol when identical copies of codewords are retransmitted.

The purpose of this thesis is to investigate the throughput and average latency performance of candidate IR and CC schemes that are compatible with the current IEEE 802.16e standard. The FEC code used is the class of turbo codes originally invented by Berrou *et. al.* [3] in 1993.

The rest of this thesis is organized as follows. Chapter 2 provides a brief overview of the ARQ protocols and related CRC, modulation and frame format defined by the IEEE 802.16e standard. The following chapter discusses possible receiver and decoder structure and algorithm. In Chapter 4 we present several candidate IR and CC schemes that are compatible with the standard and analyze their performance. Numerical performance is provided and comparison is made. Finally, the last chapter contains some concluding remarks and suggests a few potential research topics.

Chapter 2

Overview of the IEEE 802.16e Hybrid ARQ Mechanism

IEEE 802.16e specifies Hybrid ARQ (HARQ) procedures for error recovery. Soft combining of information associated with a retransmission and with previous erroneous transmissions is carried out to minimize the amount of redundant information and power transmitted over the air interface by the coding scheme of convolution code or convolutional turbo code (CTC). As the CTC has been shown to provide tremendous coding gains for both additive white Gaussian noise (AWGN) and flat Rayleigh-fading channels, we shall only consider CTC as the main coding scheme in our study.

In this chapter, we describe detailed HARQ implementation of CTC in IEEE 802.16e, i.e., the HARQ protocol Shown in Fig. 2.1.

2.1 Padding

MAC PDU (or concatenated MAC PDUs) is a basic unit processed in the channel coding and modulation blocks. When the size of MAC PDU (or concatenated MAC PDUs) is not the element in the allowed set for Hybrid ARQ, '1's are padded at the end of MAC PDU (or concatenated MAC PDUs). The amount of the padding is the same as the difference between the size of the PDU (or concatenated MAC PDUs) and the smallest element in the allowed set that is not less than the size of the PDU (or concatenated MAC PDUs). The padded packet is input into the CRC encoding block.

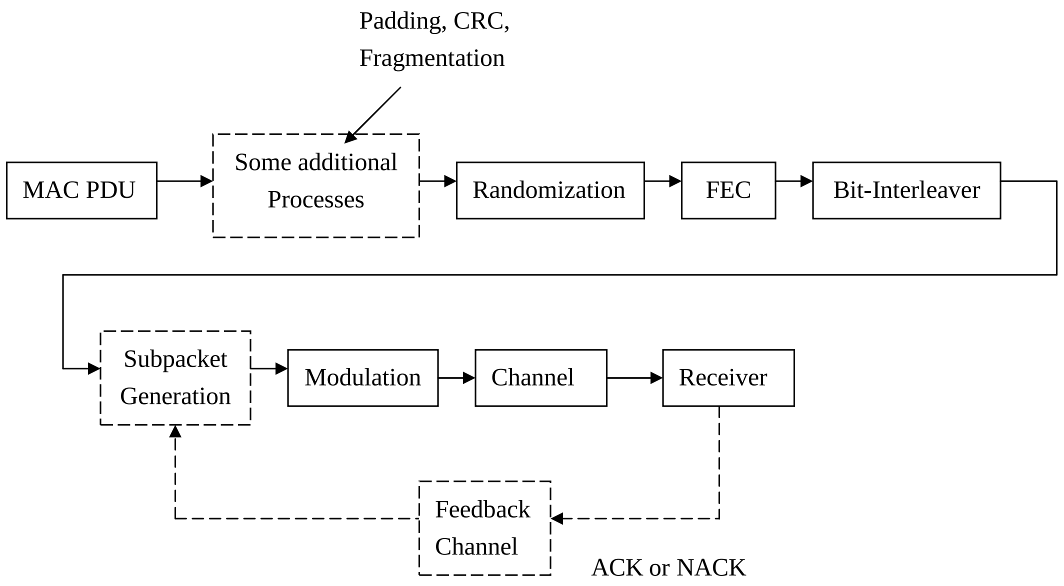


Figure 2.1: Block diagram of Hybrid ARQ mechanism based CTCs.

The allowed set is {32, 80, 128, 176, 272, 368, 464, 944, 1904, 2864, 3824, 4784, 9584, 14384, 19184, 23984} bits.

2.2 CRC encoding

When Hybrid ARQ is applied to a packet, error detection is provided on the padded packet through a Cyclic Redundancy Check(CRC).

The size of the CRC is 16 bits. CRC16-CCITT, as defined in ITU-T Recommendation X.25, shall be included at the end of the padded packet. The CRC covers both the padded bits and the information part of the padded packet. It uses the stop-and-wait protocol for retransmission.

After the CRC operation, the packet size shall belong to set {48, 96, 144, 192, 288, 384, 480, 960, 1920, 2880, 3840, 4800, 9600, 14400, 19200, 24000}.

2.3 Fragmentation

When the packet size after padding and CRC encoding is $n \times 4800$ bits, the bit stream is separately encoded in blocks of 4800 bits and concatenated as the same order of the separation before modulation. No operation is performed for the packet whose size after the padding and CRC encoding is not more than 4800 bits. The bits output from the fragmentation block are denoted by $r_1, r_2, \cdots, r_{N_{EP}}$, and this sequence is defined as encoder packet. N_{EP} is the number of the bits in an encoder packet and defined as encoder packet size. The values of N_{EP} are 48, 96, 144, 192, 288, 384, 480, 960, 1920, 2880, 3840, 4800, respectively.

2.4 Randomization

Randomization is performed on each encoder packet, which means that for each encoder packet the randomizer shall be initialized independently.

The PRBS (Pseudo-Random Binary Sequence) generator shall be $1 + x^{14} + x^{15}$ as shown in Fig. 2.2. Each data byte to be transmitted shall enter sequentially into the randomizer, MSB first. Preambles are not randomized. The seed value shall be used to calculate the randomization bits, which are combined in an XOR operation with the serialized bit stream of each FEC block.

The scrambler is initialized with the vector [LSB] 0 1 1 0 1 1 1 0 0 0 1 0 1 0 1 [MSB].

2.5 Convolutional turbo codes(CTC)

2.5.1 CTC encoder

The CTC encoder, including its constituent encoder, is depicted in Figure 2.3. It uses a double binary Circular Recursive Systematic Convolutional code. The bits of the data to be encoded are alternately fed to A and B, starting with the MSB of the first

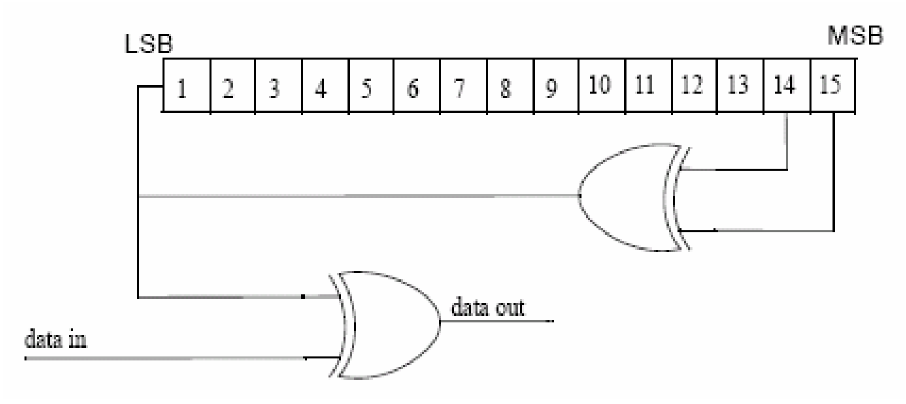


Figure 2.2: PRBS generator of the randomization.

byte being fed to A. The encoder is fed by blocks of k bits or N couples ($k = 2{*}N$ bits). For all the frame sizes, k is a multiple of 8 and N is a multiple of 4. Further, N shall be limited to: $8 \leq N/4 \leq 1024$.

The polynomials defining the connections are described in octal and symbol notations as follow:

1. For the feedback branch: 0xB, equivalently $1 + D + D^3$ (in symbolic notation).

2. For the Y parity bit: 0xD, equivalently $1 + D^2 + D^3$.

3. For the W parity bit: 0x9, equivalently $1 + D^3$.

First, the encoder (after initialization by the circulation state Sc_1, see 2.5.3) is fed the sequence in the natural order (position 1) with the incremental address $i = 0, ..., N-1$. This first encoding is called C_1 encoding. Then the encoder (after initialization by the circulation state Sc_2, see 2.5.3) is fed by the interleaved sequence (switch in position 2) with incremental address $j = 0, ..., N-1$. This second encoding is called C_2 encoding.

The order in which in the encoded bit shall be fed into the subpacket generation block (2.5.4) is:

$$A, B, Y_1, Y_2, W_1, W_2 =$$

$$A_0, A_1, ..., A_{N-1}, B_0, B_1, ..., B_{N-1}, Y_{1,0}, Y_{1,1}, ..., Y_{1,N-1}, Y_{2,0}, Y_{2,1}, ..., Y_{2,N-1},$$

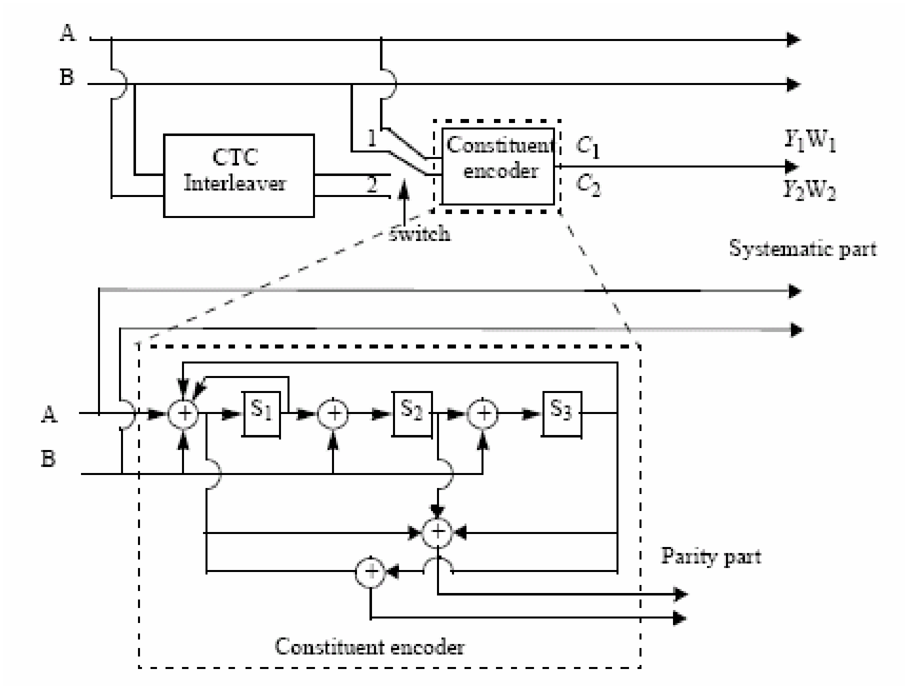


Figure 2.3: A CTC encoder.

$$W_{1,0}, W_{1,1}, ..., W_{1,N-1}, W_{2,0}, W_{2,1}, ..., W_{2,N-1}$$

2.5.2 CTC interleaver

The interleaver requires the parameters P_0, P_1, P_2 and P_3, shown in Table 2.1.

The two-step interleaver shall be performed by:

Step 1: switch alternate couples

Let the sequence $u_0 = [(A_0, B_0), (A_1, B_1), (A_2, B_2), (A_3, B_3), ..., (A_{N-1}, B_{N-1})]$ be the input to first encoding C_1

for i=0...$N-1$

if (i mod 2==1) let $(A_i, B_i) \to (B_i, A_i)$ (i.e., switch the couple)

This step gives a sequence $u_1 = [(A_0, B_0), (B_1, A_1), (A_2, B_2), (B_3, A_3), ...(B_{N-1}, A_{N-1})] = [u_1(0), u_1(1), u_1(2), u_1(3), ..., u_1(N-1)]$.

Step 2: $P(j)$

The function $P(j)$ provides the address of the couple of the sequence u_1 that shall be mapped onto the address j of the interleaved sequence (i.e., $u_2(j) = u_1(P(j))$).

```
for $j = 0...N-1$
    switch $j$ mod 4:
    case 0:$(j) = (P_0 \cdot j + 1)_{modN}$
    case 1:$(j) = (P_0 \cdot j + 1 + N/2 + P_1)_{modN}$
    case 2:$(j) = (P_0 \cdot j + 1 + P_2)_{modN}$
    case 3:$(j) = (P_0 \cdot j + 1 + N/2 + P_3)_{modN}$
```

This step gives a sequence $u_2 = [u_1(P(0)), u_1(P(1)), u_1(P(2)), u_1(P(3)), ..., u_1(P(N-1))] = [(B_{P(0)}, A_{P(0)}), (A_{P(1)}, B_{P(1)}), (B_{P(2)}, A_{P(2)}), (A_{P(3)}, B_{P(3)}), ..., (A_{P(N-1)}, B_{P(N-1)})]$. Sequence u_2 is the input to the second encoding C_2.

Date block size (bytes)	**N**	**P0**	**P1**	**P2**	**P3**
6	24	5	0	0	0
12	48	13	24	0	24
18	72	11	6	0	6
24	96	7	48	24	72
36	144	17	74	72	2
48	192	11	96	48	144
60	240	13	120	60	180
120	480	53	62	12	2
240	960	43	64	300	824
360	1440	43	720	360	540
480	1920	31	8	24	16
600	2400	53	66	24	2

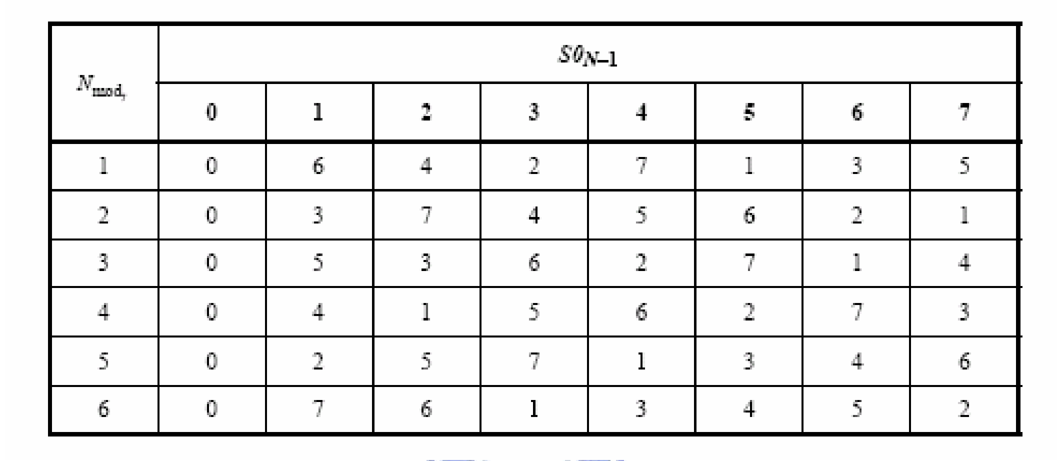
Table 2.1: CTC channel coding per modulation.

2.5.3 Determination of CTC circulation states

The state of the encoder is denoted $S (0 \leq S \leq 7)$ with $S = 4s_1 + 2s_2 + s_3$ (See Fig. 2.3) The circulation states Sc_1 and Sc_2 are determined by the following operations:

1. Initialize the encoder with state 0. Encode the sequence in the natural order for the determination of Sc_1 or in the interleaved order for determination of Sc_2. In both cases the final state of the encoder is $S0_{N-1}$.

2. According to the length N of the sequence, use Table 2.2 to find Sc_1 or Sc_2.



N_{mod}	$S0_{N-1}$							
	0	1	2	3	4	5	6	7
1	0	6	4	2	7	1	3	5
2	0	3	7	4	5	6	2	1
3	0	5	3	6	2	7	1	4
4	0	4	1	5	6	2	7	3
5	0	2	5	7	1	3	4	6
6	0	7	6	1	3	4	5	2

Table 2.2: Circulation state lookup table (Sc).

2.5.4 Subpacket generation

Proposed FEC structure punctures the mother codeword to generate a subpacket with various coding rates. Fig. 2.4 shows a block diagram of subpacket generation. 1/3 CTC encoded codeword goes through interleaving block and the puncturing is performed. Fig. 2.5 shows block diagram of the interleaving block. The puncturing is performed to select the consecutive interleaved bit sequences that starts at any point of whole codeword. For the first transmission, the subpacket is generated to select the consecutive interleaved bit sequences that starts from the first bit of the systematic part of the mother codeword. The length of the subpacket is chosen according to the needed coding rate reflecting the channel condition.

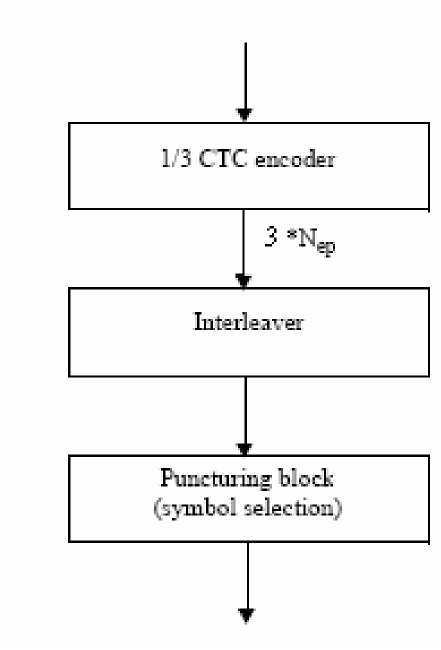


Figure 2.4: Block diagram of subpacket generation.

2.5.4.1 Symbol separation

All of the encoded symbols shall be demultiplexed into six subblocks denoted A, B, Y_1, Y_2, W_1, W_2. The encoder output symbols shall be sequentially distributed into six subblocks with the first N encoder output symbols going to the A subblock, the second N encoder output going to the B subblock, the third N to the Y_1 subblock, the forth N to the Y_2 subblock, the fifth N to the W_1 subblock, the sixth N to the W_2 subblock.

2.5.4.2 Subblock interleaving

The six subblocks shall be interleaved separately. The interleaving is performed by the unit of symbol. The sequence of interleaver output symbols for each subblock shall be generated by the procedure described below. The entire subblock of symbols to be interleaved is written into any array at address from 0 to the number of the symbols minus one $(N-1)$, and the interleaved symbols are read out in a permuted order with

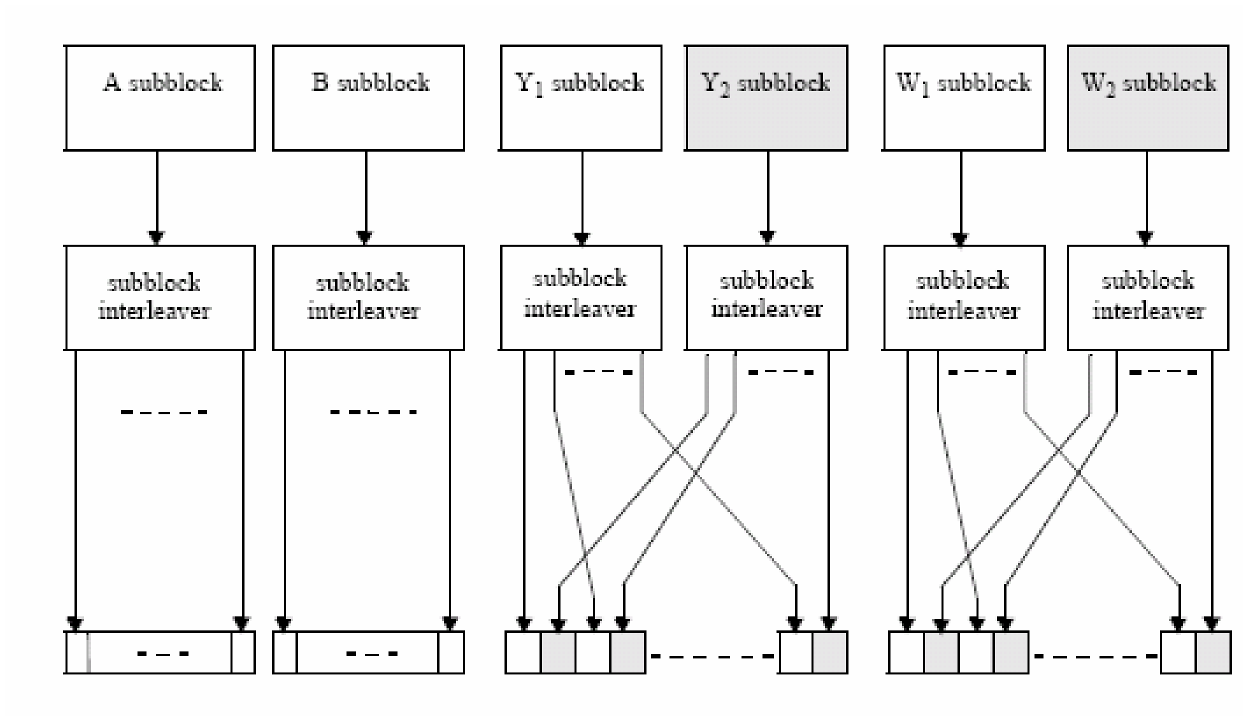


Figure 2.5: Block diagram of the interleaving scheme.

the i-th symbol being read from an address, $AD_i(i = 0...N-1)$, as follows:

1. Determine the subblock interleaver parameters, m and J. Table 2.3 gives these parameters.

2. Initialize i and k to 0.

3. Form a tentative output address $T_k$according to the formula:

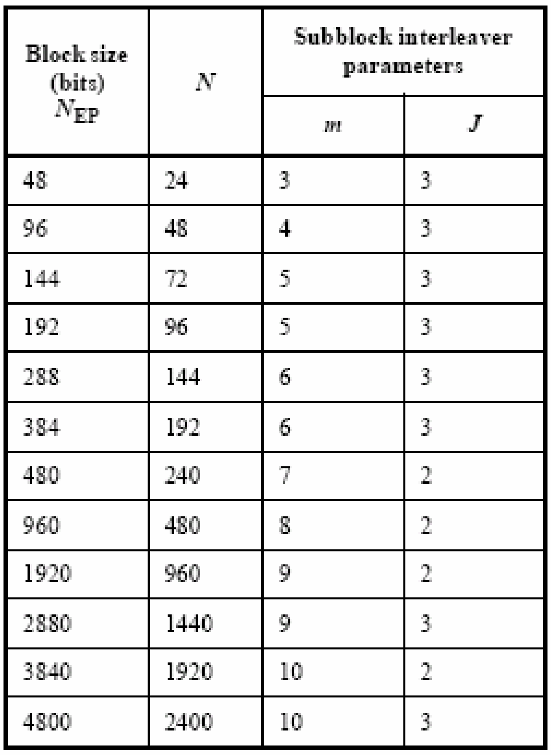
   $T_k = 2^m(k \bmod J) + BRO_m(\lfloor k/J \rfloor)$

   where $BRO_m(y)$ indicates the bit-reversed m-bit value of y (i.e., $BRO_3(6)$=3).

4. If T_k is less than $NAD_i = T_k$ and increment i and k by 1. Otherwise, discard T_k and increment k only.

5. Repeat step 3. and 4. until all N interleaver output address are obtained.

The parameters for the subblock interleavers are specified in Table 2.3.



Block size (bits) N_{EP}	N	Subblock interleaver parameters	
		m	J
48	24	3	3
96	48	4	3
144	72	5	3
192	96	5	3
288	144	6	3
384	192	6	3
480	240	7	2
960	480	8	2
1920	960	9	2
2880	1440	9	3
3840	1920	10	2
4800	2400	10	3

Table 2.3: Parameters for the subblock interleavers.

2.5.4.3 Symbol grouping

The channel interleaver output sequence shall consist of the interleaved A and B subblock sequence, followed by a symbol-by-symbol multiplexed sequence of the interleaved Y_1 and Y_2 subblock sequences, followed by a symbol-by-symbol multiplexed sequence of the interleaved W_1 and W_2 subblock sequences. The symbol-by-symbol multiplexed sequence of interleaved Y_1 and Y_2 subblock sequences shall consist of the first output bit from the Y_1 subblock interleaver, the first output bit from the Y_2 subblock interleaver,the second output bit from the Y_1 subblock interleaver, the second output bit from the Y_2 subblock interleaver, etc. The symbol-by-symbol multiplexed sequence of interleaved W_1 and W_2 subblock sequences shall consist of the first output bit from the W_1 subblock interleaver, the first output bit from the W_2 subblock interleaver, the second output bit from the W_1 subblock interleaver, the second output bit from the W_2

subblock interleaver, etc. Fig. 2.5 shows the interleaving scheme.

2.5.4.4 Symbol selection

Lastly, symbol selection shown in Fig. 2.6 is performed to generate the subpacket. The puncturing block is referred as symbols selection in the viewpoint of subpacket generation.

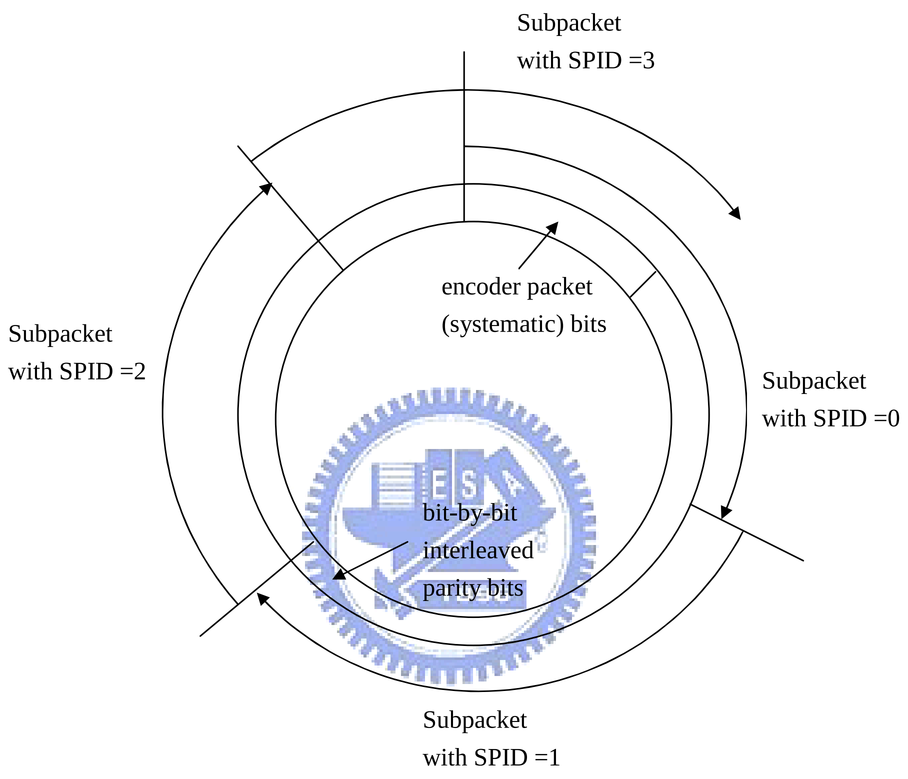


Figure 2.6: Subpacket generation.

Mother code is transmitted with one of the subpackets. The symbols in a subpacket are formed by selecting specific sequences of symbols from the interleaved CTC encoder output sequence. The resulting subpacket sequence is a binary sequence of symbols for the modulator.

Let k be the subpacket index. k=0 for the first transmission and increases by one for the next subpacket. When there are more than one FEC block in a burst, the subpacket index for each FEC block shall be the same.

N_{EP} be the number of bits in the encoder packet (before encoding).

N_{SCH} be the number of allotted slots.

m_k be the modulation order for the k-th packet (m_k=2 for QPSK, 4 for 16-QAM, and 6 for 64-QAM).

$SPID_k$ be the subpacket ID for the k-th subpacket, (for the first subpacket, $SPID_{k=0}$=0).

Also, let the scrambled and selected symbols be numbered from zero with the 0-th symbol being the first symbol in the sequence. Then, the index of the i-th symbol for the k-th subpacket shall be:

$$S_{k,i} = (F_k + i) mod(3 \cdot N_{EP})$$

where

$i = 0, ..., L_k - 1$, $L_k = 48 \cdot N_{SCH} \cdot mk$, $F_k = (SPID_k \cdot L_k) mod(3 \cdot N_{EP})$.

The N_{EP}, N_{SCH}, m_k, and $SPID$ values are determined by the BS and can be inferred by the SS through the allocation size in the DL-MAP and UL-MAP. The above symbol selection makes the following possible.

1. The first transmission includes the systematic part of the mother code.
2. The allocation of the subpacket can be determined by the SPID itself without the knowledge of previous subpacket.

The second property is very important for HARQ retransmission.

2.6 Modulation order of DL traffic burst

For DL, the modulation order (2 for QPSK, 4 for 16-QAM, and 6 for 64-QAM) shall be set for all the allowed transmission formats as shown in Table 2.4. The transmission format is defined by N_{EP} (Encoding Packet Size) and N_{SCH} (number of allotted slots). N_{EP} per an encoding packet can be chosen from the set $\{144, 192, 288, 384, 480, 960, 1920, 2880, 3840, 4800\}$ while N_{SCH} per an encoding packet is $\{1, \cdots, 480\}$. In Table 2.4, the numbers in the first row are N_{EP}'s and the numbers in the remaining rows are N_{SCH}'s and related parameters.

The supportable modulation schemes are QPSK, 16-QAM, and 64-QAM. When the N_{EP} and the N_{SCH} are given, the modulation order is determined by the value of MPR (Modulation order Product code Rate). The MPR means the effective number of the information bits transmitted per a subcarrier and is defined by Equation (2.1).

$$MPR = \frac{N_{EP}}{48 \cdot N_{SCH}} \tag{2.1}$$

Then, the modulation order is specified by the following rule:

If $0 < MPR < 1.5$, then a QPSK (modulation order 2) is used.

If $1.5 < MPR < 3.0$, then a 16QAM (modulation order 4) is used.

If $3.0 < MPR < 5.4$, then a 64QAM (modulation order 6) is used.

The effective code rate is equal to MPR divided by the modulation order (i.e., 2 for QPSK).

2.7 Date modulation

Following the subpacket generation block, the data bits are entered serially to the constellation mapper. Gray-mapped QPSK and 16-QAM (as shown in Fig. 2.7) shall be supported, whereas the support of 64-QAM is optional. The constellations (as shown in Fig. 2.7) shall be normalized by multiplying the constellation point with the indicated factor c to achieve equal average power.

The constellation-mapped data shall be subsequently modulated onto the allocated data subcarriers.

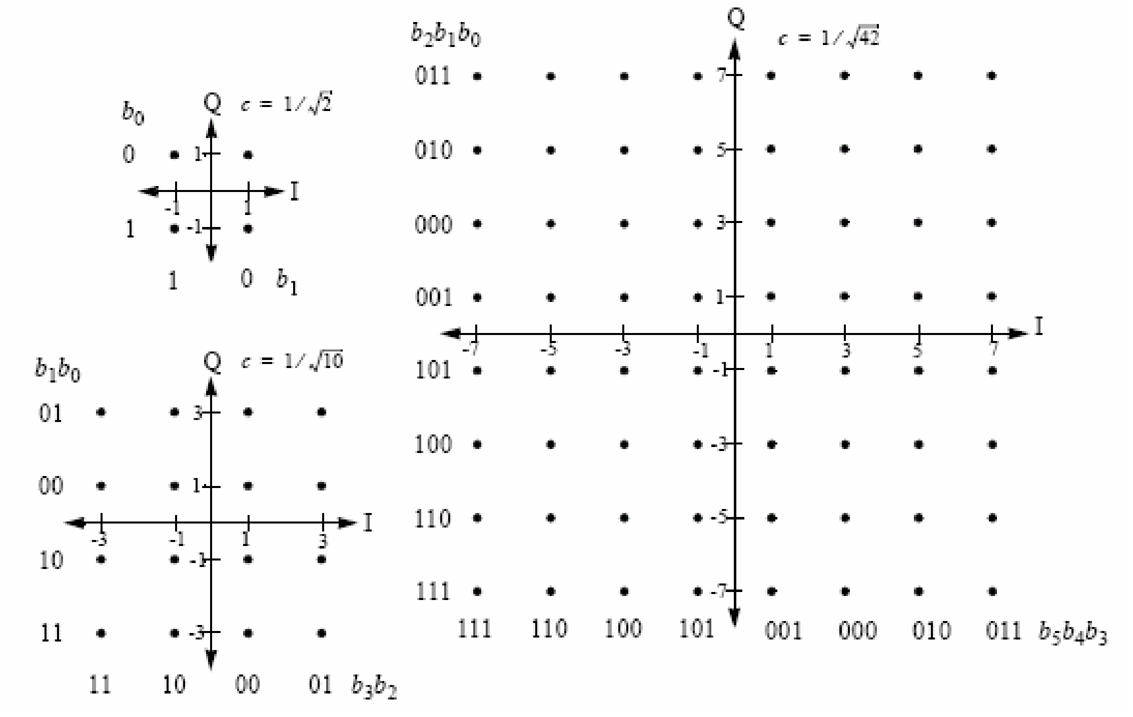


Figure 2.7: QPSK, 16-QAM, and 64-QAM constellations

2.8 TDD vs. FDD mode

IEEE 802.16e standard specifies both TDD and FDD modes of operation, there are several reasons to focus on TDD. TDD operation provides several benefits including the flexibility to partition downlink and uplink resources as a function of asymmetric traffic demand and better channel reciprocity to support closed loop performance enhancing techniques. Furthermore, transceiver complexity/cost is reduced since duplexers are no longer needed and performance is improved with the elimination of duplexer-related losses.

In the case of TDD, the uplink and downlink transmissions occur at different times and usually share the same frequency. A TDD frame (see Fig. 2.8) has a fixed duration and contains one downlink and one uplink subframe. The frame is divided into an integer number of PSs(Physical Slots), which help to partition the bandwidth easily. The TDD framing is adaptive in that the bandwidth allocated to the downlink versus the uplink can vary. The split between uplink and downlink is a system parameter and is controlled at higher layers within the system.

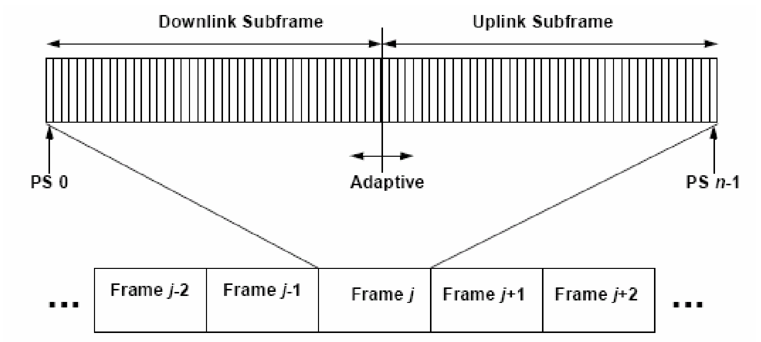


Figure 2.8: TDD frame structure.

N_{EP}	**144**	**192**	**288**	**384**	**480**	**960**	**1920**	**2880**	**3840**	**4800**
Sch	1.00	1.00								
MPR	3.00	4.00								
MOD	6.00	6.00								
Rate	1/2	2/3								
Rate	0.50	0.67								
Sch	2.00	2.00	2.00	2.00	2.00					
MPR	1.50	2.00	3.00	4.00	5.00					
MOD	4.00	4.00	6.00	6.00	6.00					
Rate	3/8	1/2	1/2	2/3	5/6					
Rate	0.38	0.50	0.50	0.67	0.83					
Sch	3.00	3.00	3.00	3.00	3.00					
MPR	1.00	1.33	2.00	2.67	3.33					
MOD	2.00	2.00	4.00	4.00	6.00					
Rate	1/2	2/3	1/2	2/3	5/9					
Rate	0.50	0.67	0.50	0.67	0.56					
Sch		4.00	4.00	4.00	4.00	4.00				
MPR		1.00	1.50	2.00	2.50	5.00				
MOD		2.00	4.00	4.00	4.00	6.00				
Rate		12	3/8	1/2	5/8	5/6				
Rate		0.50	0.38	0.50	0.63	0.83				
Sch	5.00		5.00	5.00	5.00	5.00				
MPR	0.60		1.20	1.60	2.00	4.00				
MOD	2.00		2.00	4.00	4.00	6.00				
Rate	3/10		3/5	2/5	1/2	2/3				
Rate	0.30		0.60	0.40	0.50	0.67				
Sch	6.00	6.00	6.00	6.00	6.00	6.00				
MPR	0.50	0.67	1.00	1.33	1.67	3.33				
MOD	2.00	2.00	2.00	2.00	4.00	6.00				
Rate	1/4	1/3	1/2	2/3	5/12	5/9				
Rate	0.25	0.33	0.50	0.67	0.42	0.56				
Sch		8.00		8.00	8.00	8.00	8.00			
MPR		0.50		1.00	1..25	2.50	5.00			
MOD		2.00		2.00	2.00	4.00	6.00			
Rate		1/4		1/2	5/8	5/8	5/6			
Rate		0.25		0.50	0.63	0.63	0.83			

Table 2.4: Transmission format and modulation level for DL.

N_{EP}	**144**	**192**	**288**	**384**	**480**	**960**	**1920**	**2880**	**3840**	**4800**
Sch	9.00		9.00				9.00			
MPR	0.33		0.67				4.44			
MOD	2.00		2.00				6.00			
Rate	1/6		1/3				20/27			
Rate	0.17		0.33				0.74			
Sch					10.00	10.00	10.00			
MPR					1.00	2.00	4.00			
MOD					2.00	4.00	6.00			
Rate					1/2	1/2	2/3			
Rate					0.50	0.50	0.67			
Sch	12.00	12.00	12.00	12.00				12.00		
MPR	0.25	0.33	0.50	0.67				5.00		
MOD	2.00	2.00	2.00	2.00				6.00		
Rate	1/8	1/6	1/4	1/3				5/6		
Rate	0.13	0.17	0.25	0.33				0.83		
Sch						13.00	13.00	13.00		
MPR						1.54	3.08	4.62		
MOD						4.00	6.00	6.00		
Rate						5/13	20/39	10/13		
Rate						0.38	0.51	0.77		
Sch					15.00	15.00	15.00	15.00		
MPR					0.67	1.33	2.67	4.00		
MOD					2.00	2.00	4.00	6.00		
Rate					1/3	2/3	2/3	2/3		
Rate					0.33	0.67	0.67	0.67		
Sch		16.00		16.00					16.00	
MPR		0.25		0.50					5.00	
MOD		2.00		2.00					6.00	
Rate		1/8		1/4					5/6	
Rate		0.13		0.25					0.83	
Sch	18.00		18.00						18.00	
MPR	0.17		0.33						4.44	
MOD	2.00		2.00						6.00	
Rate	1/12		1/6						20/27	
Rate	0.08		0.17						0.74	

N_{EP}	**144**	**192**	**288**	**384**	**480**	**960**	**1920**	**2880**	**3840**	**4800**
Sch					20.00	20.00	20.00	20.00	20.00	20.00
MPR					0.50	1.00	2.00	3.00	4.00	5.00
MOD					2.00	2.00	4.00	6.00	6.00	6.00
Rate					1/4	1/2	1/2	1/2	2/3	5/6
Rate					0.25	0.50	0.50	0.50	0.67	0.83
Sch								22.00		22.00
MPR								2.73		4.55
MOD								4.00		6.00
Rate								15/22		25/33
Rate								0.68		0.76
Sch		24.00	24.00	24.00						
MPR		0.17	0.25	0.33						
MOD		2.00	2.00	2.00						
Rate		1/12	1/8	1/6						
Rate		0.08	0.13	0.17						
Sch							26.00		26.00	26.00
MPR							1.54		3.08	3.85
MOD							4.00		6.00	6.00
Rate							5/13		20/39	25/39
Rate							0.38		0.51	0.64
Sch					30.00	30.00	30.00	30.00	30.00	
MPR					0.33	0.67	1.33	2.00	2.67	
MOD					2.00	2.00	2.00	4.00	4.00	
Rate					1/6	1/3	2/3	1/2	2/3	
Rate					0.17	0.33	0.67	0.50	0.67	
Sch				32.00						32.00
MPR				0.25						3.13
MOD				2.00						6.00
Rate				1/8						25/48
Rate				0.13						0.52
Sch		36.00								
MPR		0.17								
MOD		20..								
Rate		1/12								
Rate		0.08								

N_{EP}	144	192	288	384	480	960	1920	2880	3840	4800
Sch										38.00
MPR										2.63
MOD										4.00
Rate										25/38
Rate										0.66
Sch					40.00	40.00	40.00	40.00	40.00	
MPR					0.25	0.50	1.00	1.50	2.00	
MOD					2.00	2.00	2.00	4.00	4.00	
Rate					1/8	1/4	1/2	3/8	1/2	
Rate					0.13	0.25	0.50	0.38	0.50	
Sch								44.00		
MPR								1.36		
MOD								2.00		
Rate								15/22		
Rate								0.68		
Sch				48.00						
MPR				0.17						
MOD				2.00						
Rate				1/12						
Rate				0.08						
Sch										50.00
MPR										2.00
MOD										4.00
Rate										1/2
Rate										0.50
Sch									52.00	
MPR									1.54	
MOD									4.00	
Rate									5/13	
Rate									0.38	
Sch					60.00	60.00	60.00	60.00	60.00	
MPR					0.17	0.33	0.67	1.00	1.33	
MOD					2.00	2.00	2.00	2.00	2.00	
Rate					1/12	1/6	1/3	1/2	2/3	
Rate					0.08	0.17	0.33	0.50	0.67	

N_{EP}	**144**	**192**	**288**	**384**	**480**	**960**	**1920**	**2880**	**3840**	**4800**
Sch										64.00
MPR										1.56
MOD										4.00
Rate										25/64
Rate										0.39
Sch										76.00
MPR										1.32
MOD										2.00
Rate										25/38
Rate										0.66
Sch						80.00	80.00		80.00	
MPR						0.25	0.50		1.00	
MOD						2.00	2.00		2.00	
Rate						1/8	1/4		1/2	
Rate						0.13	0.25		0.50	
Sch								90.00		
MPR								0.67		
MOD								2.00		
Rate								1/3		
Rate								0.33		
Sch										100.0
MPR										1.00
MOD										2.00
Rate										1/2
Rate										0.50
Sch						120.0	120.0	120.0	120.0	
MPR						0.17	0.33	0.50	0.67	
MOD						2.00	2.00	2.00	2.00	
Rate						1/12	1/6	1/4	1/3	
Rate						0.08	0.17	0.25	0.33	
Sch										150.0
MPR										0.67
MOD										2.00
Rate										1/3
Rate										0.33

N_{EP}	**144**	**192**	**288**	**384**	**480**	**960**	**1920**	**2880**	**3840**	**4800**
Sch							160.0		160.0	
MPR							0.25		0.50	
MOD							2.00		2.00	
Rate							1/8		1/4	
Rate							0.13		0.25	
Sch								180.0		
MPR								0.33		
MOD								2.00		
Rate								1/6		
Rate								0.17		
Sch										200.0
MPR										0.50
MOD										2.00
Rate										1/4
Rate										0.25
Sch							240.0	240.0	240.0	
MPR							0.17	0.25	0.33	
MOD							2.00	2.00	2.00	
Rate							1/12	1/8	1/6	
Rate							0.08	0.13	0.17	
Sch										300.0
MPR										0.33
MOD										2.00
Rate										1/6
Rate										0.17
Sch									320.0	
MPR									0.25	
MOD									2.00	
Rate									1/8	
Rate									0.13	
Sch								360.0		
MPR								0.17		
MOD								2.00		
Rate								1/12		
Rate								0.08		

N_{EP}	**144**	**192**	**288**	**384**	**480**	**960**	**1920**	**2880**	**3840**	**4800**
Sch										400.0
MPR										0.25
MOD										2.00
Rate										1/8
Rate										0.13
Sch									480.0	
MPR									0.17	
MOD									2.00	
Rate									1/12	
Rate									0.08	

Chapter 3

Turbo Decoding: Structure and Algorithm

This chapter considers the receiving aspect of the HARQ protocols based on the specifications given in the previous chapter. We discuss de-mapper and soft-in soft-out turbo decoder structure and performance. However, to comply with the IEEE 802.16e standard, we need to make some modifications.

3.1 Decoding CTC-coded Signals

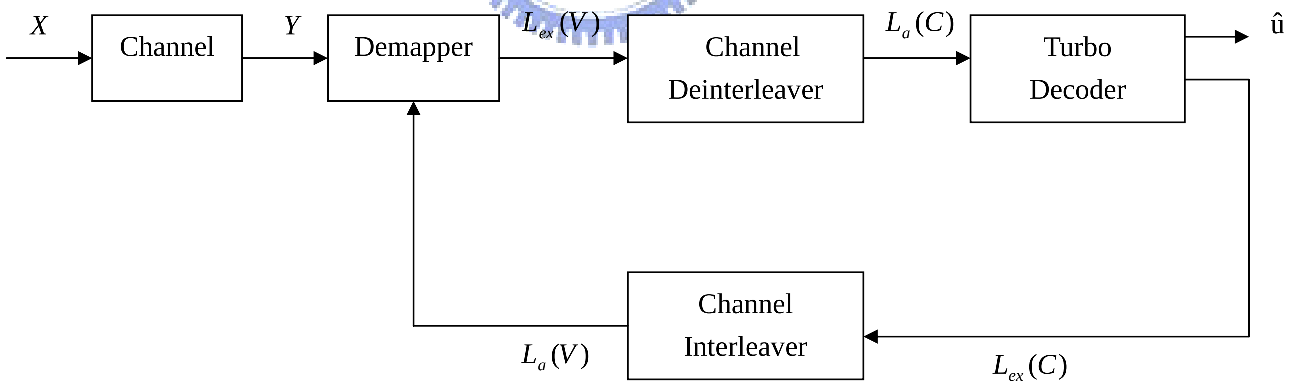


Figure 3.1: Receiver block diagram for decoding a CTC-coded waveform.

The received signal can be represented as $Y = HX + N$, where H is the channel gain and N is the complex additive Gaussian noise. Here we used the method with separate steps, demapper and decoder. They are separated by bit interleavers used to return the coded bit information to original sequence. In Fig. 3.1, C is the coded bits and V is the interleaved coded bits. The details of the demapper and soft-in soft-out Turbo decoder are described below:

3.1.1 Demapper

This block is used to demodulate channel symbol and obtain bit information for decoding. The received signals are $Y = \{y_0, y_1, ...\}$, where y_t represents the received signal at time t. The interleaved coded bits are $V = \{V_0, V_1, ...\}$ where V_t represents the interleaved coded bits at time t. $V_t = [V_t^0, V_t^1, ..., V_t^m]$, where m is the modulation order (i.e. 2 for QPSK, 4 for 16-QAM, 6 for 64-QAM).

The bit information is computed by using the maximum a-posterior probability criterion. The a-posterior probability of coded bit can be calculated as

$$p\,(V_t^i = c \,|\, y_t) = \sum_{w \in \Omega_c^i} p\,(w \,|\, y_t) = \sum_{w \in \Omega_c^i} \frac{p\,(y_t \,|\, w) p\,(w)}{p\,(y_t)} \tag{3.1}$$

where $\Omega_c^i = \{\mu(\,[V_t^0, V_t^1, ..., V_t^m]\,) \,|\, V_t^i = c\}$ is a subset of modulation constellation, μ is the mapper operator, c=0 or 1 and w is a modulation symbol. For the fading channel, the conditional probability of received signal can be represented as the complex Gaussian distribution

$$p\,(y_t \,|\, w) = \frac{1}{2\pi\sigma^2} e^{-\frac{|\,y_t - H_t w\,|^2}{2\sigma^2}} \tag{3.2}$$

where σ^2 is the noise variance.

We use the log likelihood ratio (LLR) to deal with the bit information. The a-posterior LLR of coded bit is defined as

$$L(V_t^i \,|\, y_t) = \ln \left[\frac{p\,(V_t^i = 0 \,|\, y_t)}{p\,(V_t^i = 1 \,|\, y_t)} \right] \tag{3.3}$$

Substituting (3.1) into (3.3) and assuming independent bits (random enough interleavers), we have

$$L(V_t^i \,|\, y_t) = \ln \left[\frac{\sum_{w \in \Omega_0^i} p\,(y_t \,|\, w) p\,(w)}{\sum_{w \in \Omega_1^i} p\,(y_t \,|\, w) p\,(w)} \right]$$

$$= \ln\left[\frac{\sum_{w\in\Omega_0^i} p\left(y_t \,|\, w\right)\prod_{i'=0}^{m_k-1} p_a\left(V_t^{i'} = V^{i'}(w)\right)}{\sum_{w\in\Omega_1^i} p\left(y_t \,|\, w\right)\prod_{i'=0}^{m_k-1} p_a\left(V_t^{i'} = V^{i'}(w)\right)}\right] \tag{3.4}$$

where $V^{i'}(w) \in \{0,1\}$ denotes the value of the i'th bit for the symbol w.

The a-priori LLR of V_t^i is defined as

$$L_a(V_t^i) = \ln\left[\frac{p_a(V_t^i = 0)}{p_a(V_t^i = 1}\right] \tag{3.5}$$

, thus we can obtain

$$p_a(V_t^i = c) = \frac{\exp\{-L_a(V_t^i)\times c\}}{1+\exp\{-L_a(V_t^i)\}},\ for\ c\ =\ 0\ or\ 1 \tag{3.6}$$

Substituting (3.2) and (3.6) into (3.4), we have

$$L(V_t^i \,|\, y_t) = \ln\left[\frac{\sum_{w\in\Omega_0^i}\frac{1}{2\pi\sigma^2}e^{-\frac{|y_t-H_tw|^2}{2\sigma^2}}\prod_{i'=0}^{m_k-1}\frac{\exp\{-L_a(V_t^{i'})\times V^{i'}(w)\}}{1+\exp\{-L_a(V_t^{i'})\}}}{\sum_{w\in\Omega_1^i}\frac{1}{2\pi\sigma^2}e^{-\frac{|y_t-H_tw|^2}{2\sigma^2}}\prod_{i'=0}^{m_k-1}\frac{\exp\{-L_a(V_t^{i'})\times V^{i'}(w)\}}{1+\exp\{-L_a(V_t^{i'})\}}}\right]$$

$$= \ln\left[\frac{\sum_{w\in\Omega_0^i}\exp\{-\frac{|y_t-H_tw|^2}{2\sigma^2}-\sum_{i'=0}^{m_k-1}L_a(V_t^{i'})\times V^{i'}(w)\}}{\sum_{w\in\Omega_1^i}\exp\{-\frac{|y_t-H_tw|^2}{2\sigma^2}-\sum_{i'=0}^{m_k-1}L_a(V_t^{i'})\times V^{i'}(w)\}}\right] \tag{3.7}$$

The a-posterior LLR of the coded bit can also be written as

$$L(V_t^i \,|\, y_t) = \underbrace{\ln\left[\frac{p\,(y_t \,|\, V_t^i = 0)}{p\,(y_t \,|\, V_t^i = 1)}\right]} + \underbrace{\ln\left[\frac{p\,(V_t^i = 0)}{p\,(V_t^i = 1)}\right]}$$

$$= \text{extrinsic information} + \text{a-priori probability}$$

$$= \ln\left[\frac{\sum_{w\in\Omega_0^i}\exp\{-\frac{|y_t-H_tw|^2}{2\sigma^2}-\sum_{i'=0,i'\neq i}^{m_k-1}L_a(V_t^{i'})\times V^{i'}(w)\}}{\sum_{w\in\Omega_1^i}\exp\{-\frac{|y_t-H_tw|^2}{2\sigma^2}-\sum_{i'=0,i'\neq i}^{m_k-1}L_a(V_t^{i'})\times V^{i'}(w)\}}\right] + L_a(V_t^i) \tag{3.8}$$

The extrinsic information term output by the demapper is

$$L_{ex}(V_t^i) = \ln\left[\frac{\sum_{w\in\Omega_0^i}\exp\{-\frac{|y_t-H_tw|^2}{2\sigma^2}-\sum_{i'=0,i'\neq i}^{m_k-1}L_a(V_t^{i'})\times V^{i'}(w)\}}{\sum_{w\in\Omega_1^i}\exp\{-\frac{|y_t-H_tw|^2}{2\sigma^2}-\sum_{i'=0,i'\neq i}^{m_k-1}L_a(V_t^{i'})\times V^{i'}(w)\}}\right] \tag{3.9}$$

where the a-priori information $L_a(V_t^i)$ comes from the output of the decoder in Fig. 3.1. Because $L_a(V_t^i)$ is not available at the first demapping, we assume it is equally likely, and (3.9) becomes

$$L_{ex}(V_t^i) = \ln\left[\frac{\sum_{w\in\Omega_0^i}\exp\{-\frac{|y_t-H_tw|^2}{2\sigma^2}\}}{\sum_{w\in\Omega_1^i}\exp\{-\frac{|y_t-H_tw|^2}{2\sigma^2}\}}\right] \tag{3.10}$$

Then $L_{ex}(V_t^i)$ is deinterleaved and sent to the decoder.

After the first decoding, the extrinsic information of coded bits $L_{ex}(C)$ is delivered by the decoder to the interleaver and becomes $L_a(V)$, the a-priori probability of the demapper. The process to exchange information between demapper and decoder is continued until the final decoding output $\hat{u}$.

3.1.2 Soft-in soft-out Turbo decoder

Due to the double binary property, we cannot simply judge original message on one LLR value of a posteriori probabilities as that of the classical Turbo decoder. Author in [8] mentioned a modified MAP algorithm or BCJR algorithm which must calculate three LLRs values $L_1 = \ln\left(\frac{p\,(u_t=(01)\,|\,r)}{p\,(u_t=(00)\,|\,r)}\right)$, $L_2 = \ln\left(\frac{p\,(u_t=(10)\,|\,r)}{p\,(u_t=(00)\,|\,r)}\right)$ and $L_3 = \ln\left(\frac{p\,(u_t=(11)\,|\,r)}{p\,(u_t=(00)\,|\,r)}\right)$ to decode double binary Turbo code, and consequently the computational complexity is increased. But if carefully considering the principle of MAP algorithm, we can find that there is no need to compute the LLR values in double binary Turbo decoder.

An efficient decoding scheme for double binary circular turbo codes suggested by [9] is used to find the maximum value of $p\,(u_t\,|\,r)$. For the double binary Turbo decoder, we can compute four probabilities $p\,(u_t = (0,0)\,|\,r)$, $p\,(u_t = (0,1)\,|\,r)$, $p\,(u_t = (1,0)\,|\,r)$ and $p\,(u_t = (1,1)\,|\,r)$ directly, then select the maximum one as the decoded data.

Before selecting the maximum one as the decoded data, we should exchange coded bits' information between demapper and decoder in several iterations. After deinterleaving, the output of the demapper, the a-priori probabilities of the coded bits $L_a(C)$ is utilized to decode and can be described below:

$$
\begin{aligned}
L_a(C) &= \{L_a(A), L_a(B), L_a(Y_1), L_a(Y_2), L_a(W_1), L_a(W_2)\} \\
&= \{L_a(A_0), L_a(A_1), ..., L_a(A_{N-1}), L_a(B_0), L_a(B_1), ..., L_a(B_{N-1}), \\
&\quad L_a(Y_{1,0}), L_a(Y_{1,1}), ..., L_a(Y_{1,N-1}), L_a(Y_{2,0}), L_a(Y_{2,1}), ..., L_a(Y_{2,N-1}), \\
&\quad L_a(W_{1,0}), L_a(W_{1,1}), ..., L_a(W_{1,N-1}), L_a(W_{2,0}), L_a(W_{2,1}), ..., L_a(W_{2,N-1})\} \qquad (3.11)
\end{aligned}
$$

A, B represent the double binary systematic part of the codeword, whereas Y_1, W_1 and Y_2, W_2 are the redundancy of the first and second encoders, respectively.

After decomposing the a-prioir probability of the coded bits $L_a(C)$ by (3.11), we can get the a-priori probabilities of $A_t, B_t, Y_{1,t}, Y_{2,t}, W_{1,t}, W_{2,t}$ respectively.

The soft-in soft-out turbo decoder is illustrated in Fig. 3.2.

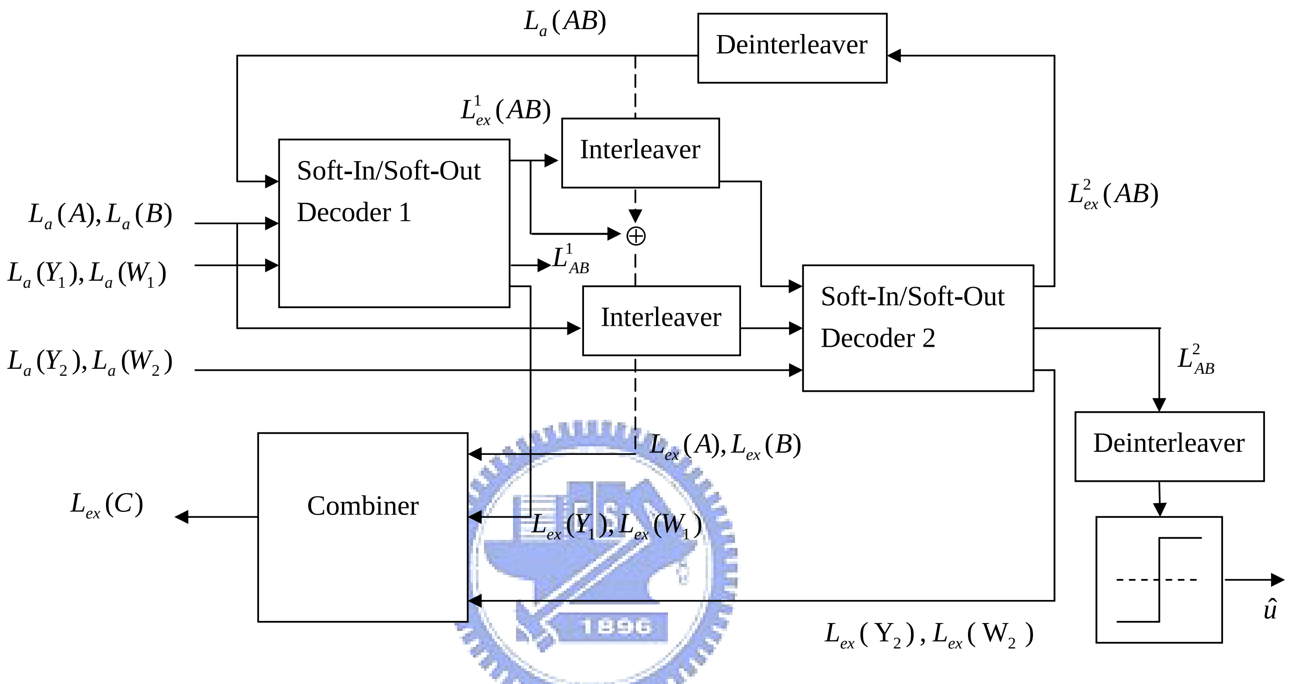


Figure 3.2: Turbo decoder block diagram.

We begin our development of the BCJR algorithm by rewriting the APP value $p\,(u_t = (0,0)\,|\,r)$ as follows:

$$p\,(u_t = (0,0)\,|\,r) = \frac{p\,(u_t = (0,0), r)}{p\,(r)} = \frac{\sum_{(s',s)\in\sum_t^{00}} p\,(s_t = s', s_{t+1} = s, r)}{p(r)} \qquad (3.12)$$

where $\sum_t^{00}$ is the set of all state pairs $s_t = s'$ and $s_{t+1} = s$ that correspond to the data symbol $u_t = (0,0)$ at time t. We can reformulate the expressions $p\,(u_t = (0,1)\,|\,r)$, $p\,(u_t = (1,0)\,|\,r)$ and $p\,(u_t = (1,1)\,|\,r)$ in the same way.

We evaluate the joint pdf $p(s', s, r)$:

$$p\,(s', s, r) = p\,(s', s, r_{0\sim t-1}, r_t, r_{t+1\sim K}) \qquad (3.13)$$

where K is the end state.

Now, application of Bayes' rule yields

$$\begin{aligned} p\left(s', s, r\right) &= p\left(r_{t+1\sim K} \,|\, s', s, r_{0\sim t-1}, r_t\right) p\left(s', s, r_{0\sim t-1}, r_t\right) \\ &= p\left(r_{t+1\sim K} \,|\, s', s, r_{0\sim t-1}, r_t\right) p\left(s, r_t \,|\, s', r_{0\sim t-1}\right) p\left(s', r_{0\sim t-1}\right) \\ &= p\left(r_{t+1\sim K} \,|\, s\right) p\left(s, r_t \,|\, s'\right) p\left(s', r_{0\sim t-1}\right) \end{aligned} \tag{3.14}$$

where the last equality follows from the fact that the probability of the received branch at time t depends only on the state and data symbol at time t. Defining

$$\alpha_t(s') \equiv p(s', r_{0\sim t-1}) \tag{3.15}$$

$$\gamma_t(s', s) \equiv p\left(s, r_t \,|\, s'\right) \tag{3.16}$$

$$\beta_{t+1}(s) \equiv p\left(r_{t+1\sim K} \,|\, s\right) \tag{3.17}$$

We can write (3.14) as

$$p\left(s', s, r\right) = \beta_{t+1}(s)\gamma_t(s', s)\alpha_t(s') \tag{3.18}$$

The branch metric $\gamma_t(s', s)$ can be expressed as

$$\begin{aligned} \gamma_t(s', s) &= p\left(s, r_t \,|\, s'\right) = \frac{p(s', s, r_t)}{p(s')} \\ &= \left[\frac{p(s', s)}{p(s')}\right]\left[\frac{p\left(s', s, r_t\right)}{p\left(s', s\right)}\right] \\ &= p\left(s \,|\, s'\right) p\left(r_t \,|\, s', s\right) = p(u_t) p\left(r_t \,|\, s', s\right) \end{aligned} \tag{3.19}$$

For Soft-In/Soft-Out Decoder 1:

$$\gamma_t(s', s) = p(u_t) \cdot p(A_t = c_3) \cdot p(B_t = c_2) \cdot p(Y_{1,t} = c_1) \cdot p(W_{1,t} = c_0) \tag{3.20}$$

and for Soft-In/Soft-Out Decoder 2:

$$\gamma_t(s', s) = p(u_t) \cdot p(A_t = c_3) \cdot p(B_t = c_2) \cdot p(Y_{2,t} = c_1) \cdot p(W_{2,t} = c_0) \tag{3.21}$$

where $p(A_t)$ can be calculate as (3.6)

$$p(A_t = c_3) = \frac{\exp\{-L_a(A_t) \times c_3\}}{1 + \exp\{-L_a(A_t)\}}, \text{ for } c_3 \;=\; 0 \text{ or } 1 \tag{3.22}$$

so are $p(B_t = c_2)$, $p(Y_{1,t} = c_1)$, $p(W_{1,t} = c_0)$, $p(Y_{2,t} = c_1)$ and $p(W_{2,t} = c_0)$.

We show the expressions of the probabilities recursively:

$$\alpha_{t+1}(s) = \sum_{s' \in \sigma_t} \gamma_t(s', s)\alpha_t(s'), \qquad t = 0, 1, ..., K-1 \tag{3.23}$$

where σ_t is the set of all state at time t and K is the length of the input sequence.

$$\beta_t(s') = \sum_{s' \in \sigma_{t+1}} \gamma_t(s', s)\beta_{t+1}(s) \qquad t = K-1, k-2, ..., 0 \tag{3.24}$$

where σ_{t+1} is the set of all state at time t+1

We can also use the natural logarithm of the probabilities, $\alpha_t^* = \ln(\alpha_t)$, $\beta_t^* = \ln(\beta_t)$, and $\gamma_t^* = \ln(\gamma_t)$ to express the forward and backward recursions,

$$\gamma_t^*(s', s) = \ln p(u_t) + \ln p(A_t = c_3) + \ln p(B_t = c_2) + \ln p(Y_{1,t} = c_1) + \ln p(W_{1,t} = c_0) \tag{3.25}$$

or

$$\gamma_t^*(s', s) = \ln p(u_t) + \ln p(A_t = c_3) + \ln p(B_t = c_2) + \ln p(Y_{2,t} = c_1) + \ln p(W_{2,t} = c_0) \tag{3.26}$$

$$\begin{aligned} \alpha_{t+1}^*(s) &= \ln\left[\sum_{s' \in \sigma_t} \exp(\gamma_t^*(s', s) + \alpha_t^*(s'))\right] \\ &= \overset{*}{\max_{s' \in \sigma_l}}[\gamma_t^*(s', s) + \alpha_t^*(s')] \qquad t = 0, 1, ..., K-1 \end{aligned} \tag{3.27}$$

$$\begin{aligned} \beta_t^*(s') &= \ln\left[\sum_{s' \in \sigma_t} \exp(\gamma_t^*(s', s) + \beta_{t+1}^*(s))\right] \\ &= \overset{*}{\max_{s' \in \sigma_l}}[\gamma_t^*(s', s) + \beta_{t+1}^*(s)] \qquad t = K-1, K-2, \cdots, 0 \end{aligned} \tag{3.28}$$

Because of the characteristic of tail biting described by 2.5.3, we don't need to know the initial condition of the forward recursion and backward recursion. Instead, we use the training length T_L illustrated like Fig. 3.3. To know the initial condition of the forward recursion, first, setting the initial condition of the state $K - T_L$ all equally and run the algorithm forward from it. After running to the end state K, we set the initial condition of the forward recursion as same as the condition of the end state, i.e. $\alpha_0^*(s) = \alpha_K^*(s)$ for all state s. It's the same idea of deciding the initial condition of the backward recursion. First, setting the initial condition of the state T_L all equally and run the algorithm backward from it. After running to the first state 0, we set the initial condition of the backward recursion as same as the condition of the first state, i.e. $\beta_K^*(s) = \beta_0^*(s)$ for all state s. After that, we run the algorithm as usual and choose the most likely probability as our estimated results.

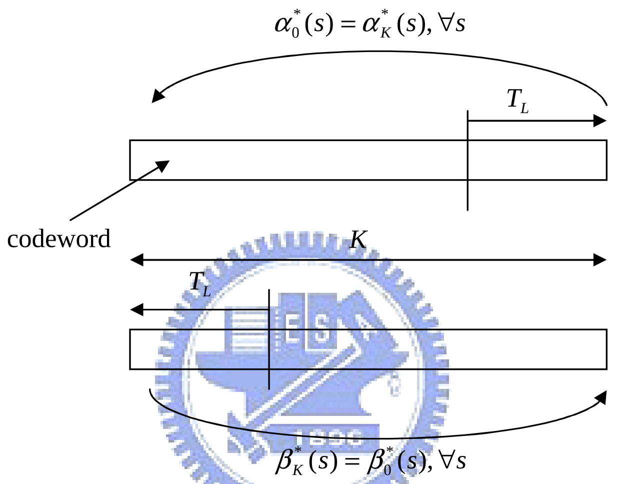

$\alpha_0^*(s) = \alpha_K^*(s), \forall s$
T_L
codeword
K
T_L
$\beta_K^*(s) = \beta_0^*(s), \forall s$

Figure 3.3: training length (T_L).

Chapter 4

Hybrid ARQ Techniques

Hybrid automatic repeat request (Hybrid-ARQ) schemes combine ARQ protocols with forward error correction codes (FEC) to provide better performance than ordinary ARQ, particularly over wireless channels, at the cost of increased implementation complexity. Basically, Hybrid ARQ schemes may be classified as Type-I, Type-II and Type-III Hybrid ARQ schemes depending on the level of complexity employed in there implementation. In this chapter, we'll introduce conventional Hybrid ARQ methods used two combining measures and then discuss an adaptive Type-II Hybrid ARQ scheme which does some modifications based on them.

4.1 Conventional HARQ methods

A simple (**Type-I**) hybrid ARQ combines FEC and pure ARQ by encoding the data block by an error-detection code (such as CRC code) and an FEC prior to transmission. When the coded data block is received, the receiver first detects if it is error free. When the incoming block fails to pass the error-detection mechanism, then, unlike the pure ARQ protocol, a retransmission request will not be issued until the receiver fails to correct it. Both throughput and delay performance can be further improved by taking advantages of the code structure and inherent diversity. Chase combining refers to the class of techniques that combine failed blocks with the retransmitted block to enhance the decoders performance at the cost of increased storage requirement. For some codes, one can partition a codeword into several parts with each part or the combinations of two or more parts decodable. The transmitter can then send these parts sequentially until an ACK is received in the return link. Such an error control scheme is called **Type II or Type III Hybrid ARQ** with incremental redundancy (IR), depending on whether each IR is self-decodable. The IR scheme encodes each re-transmission differently rather than simply repeating the same coded bits as in Chase combining. Hence it is expected to give better performance since coding is effectively done across retransmissions.

Hybrid ARQ can be used in stop-and-wait mode or in selective repeat mode. Stop-and-wait is simpler, but waiting for the receiver's acknowledgement reduces efficiency, thus multiple stop-and-wait hybrid ARQ processes are often done in parallel practically: when one hybrid ARQ process is waiting for an acknowledgement, another process can temporary use the channel to send data.

4.2 Packet combining methods

If the transmitted packet at the first time still has errors detected by the CRC after error correction, transmitter will need to retransmit. At the receiver, when receiving a packet of retransmitted data, we need to combine it with former packets in order to get higher throughput. We propose two methods below, symbol combining and LLR combining.

4.2.1 Symbol combining

From Fig. 3.1, we know that if we want to combine retransmitted symbols together, it can be modified as Fig. 4.1.

$\{X_1, X_2, ..., X_n\}$ are n times of retransmitted packets, and $\{Y_1, Y_2, ..., Y_n\}$ are n times of received packets after passing through AWGN or flat Rayleigh fading channels. $Y_j = \{y_{j,0}, y_{j,1}, ...\}$, where $y_{j,l}$ represents the lth symbol at the jth time.

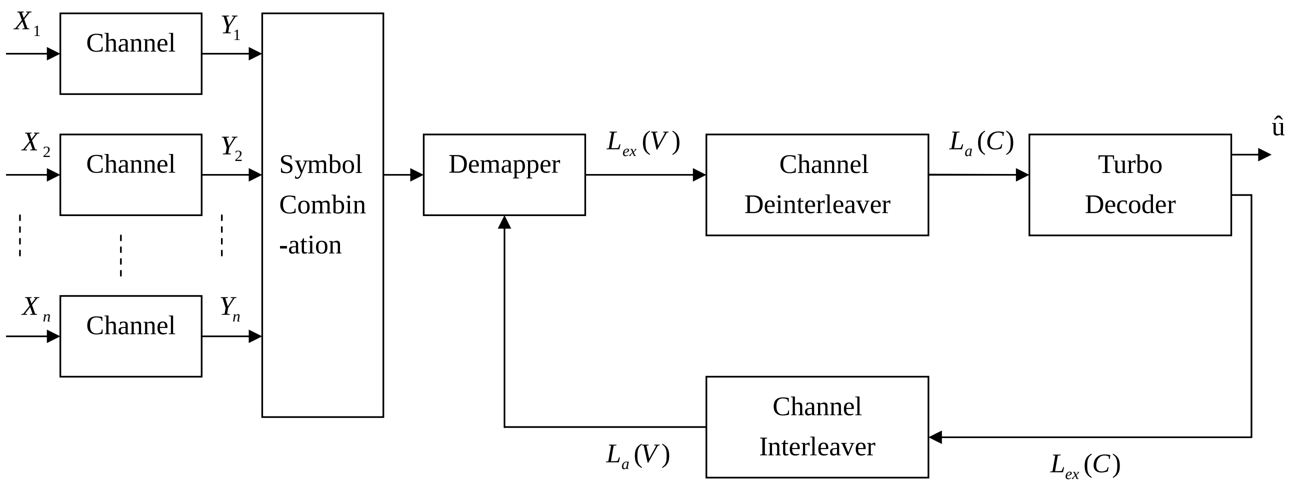


Figure 4.1: The block diagram of symbol combining.

To combine n times of packets together, (3.3) can be modified as below

$$
\begin{aligned}
L(V_t^i \mid y_{1,t}, y_{2,t}, ..., y_{n,t}) &= \ln \left[\frac{p\,(V_t^i = 0 \mid y_{1,t}, y_{2,t}, ..., y_{n,t})}{p\,(V_t^i = 1 \mid y_{1,t}, y_{2,t}, ..., y_{n,t})} \right] \\
&= \ln \left[\frac{p\,(y_{1,t}, y_{2,t}, ..., y_{n,t} \mid V_t^i = 0) p\,(V_t^i = 0)}{p\,(y_{1,t}, y_{2,t}, ..., y_{n,t} \mid V_t^i = 1) p\,(V_t^i = 1)} \right] \\
&= \ln \left[\frac{\prod_{j=1}^{n} p\,(y_{j,t} \mid V_t^i = 0) p\,(V_t^i = 0)}{\prod_{j=1}^{n} p\,(y_{j,t} \mid V_t^i = 1) p\,(V_t^i = 1)} \right] \\
&= \underbrace{\ln \left[\frac{\sum_{V_t^i=0} [\prod_{j=1}^{n} p\,(y_{j,t} \mid V_t)]}{\sum_{V_t^i=1} [\prod_{j=1}^{n} p\,(y_{j,t} \mid V_t)]} \right]} + \underbrace{\ln \left[\frac{p\,(V_t^i = 0)}{p\,(V_t^i = 1)} \right]} \qquad (4.1) \\
&= \text{extrinsic information} \quad + \quad \text{a priori probability}
\end{aligned}
$$

4.2.2 LLR combining

In order to combine n times of retransmitted packets based on LLR, Fig. 3.1 needs some modifications. After modifying, the block diagram can be shown as Fig. 4.2.

$\{V_1, V_2, ..., V_{n-1}\}$ are the former LLR values before the nth retransmission, where V_j is the jth LLR value computed by the jth (re)transmission. We combine the nth LLR value with former LLR values by $\sum_{j=1,...,n} L_{ex}(V_j)$.

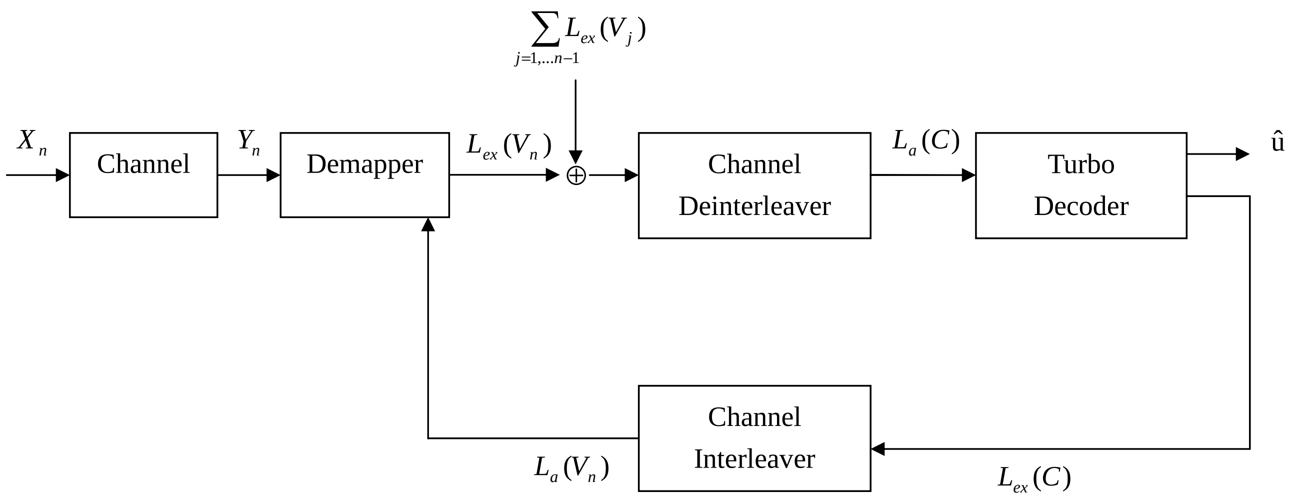


Figure 4.2: The block diagram of LLR-based combination.

4.2.3 Performance comparison

We report some simulation results in this subsection. For the CC method, we consider two equal packets with QPSK, 16QAM or 64QAM modulation. For the IR method, we choose CTC with N_{EP}=4800, rate=1/2. The FER performance over AWGN channels are shown in Fig. 4.3, Fig. 4.4 and Fig. 4.5, respectively.

Although these two combining performances are almost the same in QPSK modulation, symbol combining outperforms LLR combining about 0.4dB and 0.6dB in 16QAM and 64QAM modulations over AWGN channel respectively. However, the procedures of symbol combining is more complex than LLR combining. Besides, instead of storing codewords' extrinsic information, i.e. $\sum_{j=1,\dots,n-1} L_{ex}(V_j)$, symbol combining needs more registers to store every retransmitted packets.

4.3 Compare Chase combining and Incremental redundancy

In this section, we compare the performance of Chase combining with Incremental redundancy based on IEEE 802.16e CTC. In the Incremental redundancy, we choose transmitted subpacket in order for retransmissions, i.e. $SPID_{k=0} = 0$, $SPID_{k=1} = 1$

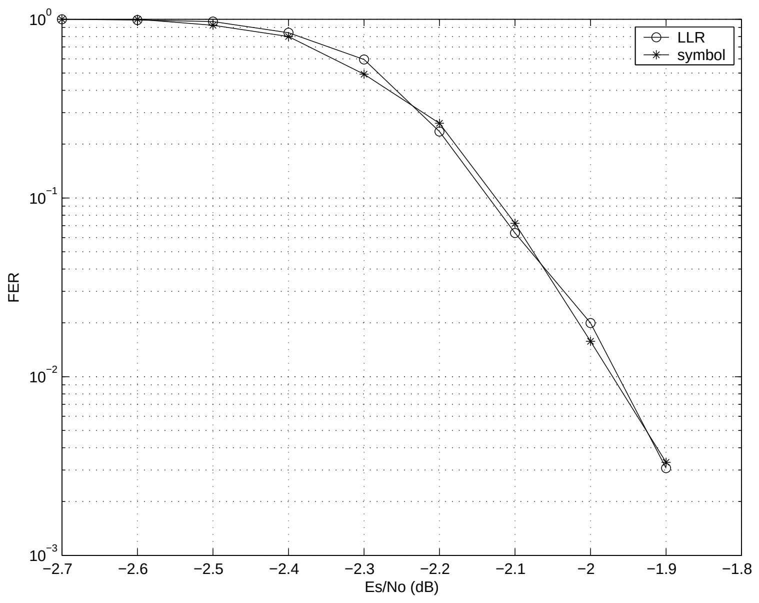


Figure 4.3: LLR vs. Symbol combining for r=1/2, QPSK, 2 frame combining, using CC over AWGN channel.

,...etc. The detail has been described in 2.5.4.4. When there are repeating parts, combining them by the methods described in 4.2. Fig. 4.6 and Fig. 4.7 are the procedures of Chase combining and Incremental redundancy, respectively.

We choose symbol combining for QPSK, 16QAM modulations and transmit the packets over AWGN channel. Fig. 4.8 and Fig. 4.9 show the results.

No matter what modulations we use, we wee that Incremental redundancy is better than Chase combining over AWGN channel. However, Incremental redundancy has more complexity than Chase combining in simulations.

4.4 An adaptive Type-II Hybrid ARQ method

We consider three modulation options, QPSK, 16QAM, and 64QAM available for WiMAX systems. In order to keep the benefit of higher throughput of 64QAM and better reliability of QPSK, we discuss an type-II hybrid ARQ scheme with adaptive modulation. This idea is similar to Link Quality Control (LQC) in the enhanced general packet radio service (EGPRS) system [10].

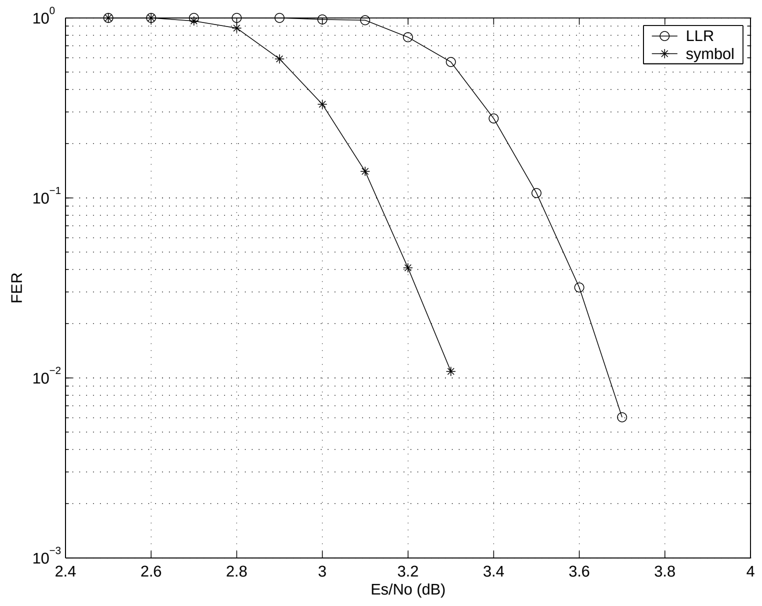


Figure 4.4: LLR vs. Symbol combining for r=1/2, 16QAM, 2 frame combining, using CC over AWGN channel.

As the best modulation is a function of the channel condition (e.g., channel gain to noise ratio) which is not always available, we use a simple channel measurement scheme for coding/modulation strategy selection. The state transition diagram shown in Fig. 4.10 describes a typical behavior of the transmission-retransmission procedure when an adaptive Hybrid ARQ is employed, where L, M_i, and H_i correspond to low, moderate and high error rate conditions, respectively and N is the number of packets that are received in the same channel condition before a new modulation and/or coding option is activated. Since the decoder performance is also a function of the channel condition. When a series of packets are successfully decoded (CRC-approved), the channel condition is likely to be good and the forthcoming packet can use higher order modulation while still meet the bit error rate (BER) requirement. In case there is a CRC detection error, the sender then uses a lower order modulation and the receiver combines the result with prior transmission by Chase combining. The sender is assumed to be initially in State I and uses 64QAM signal.

We use a graphic representation of the transform domain behavior of an adaptive

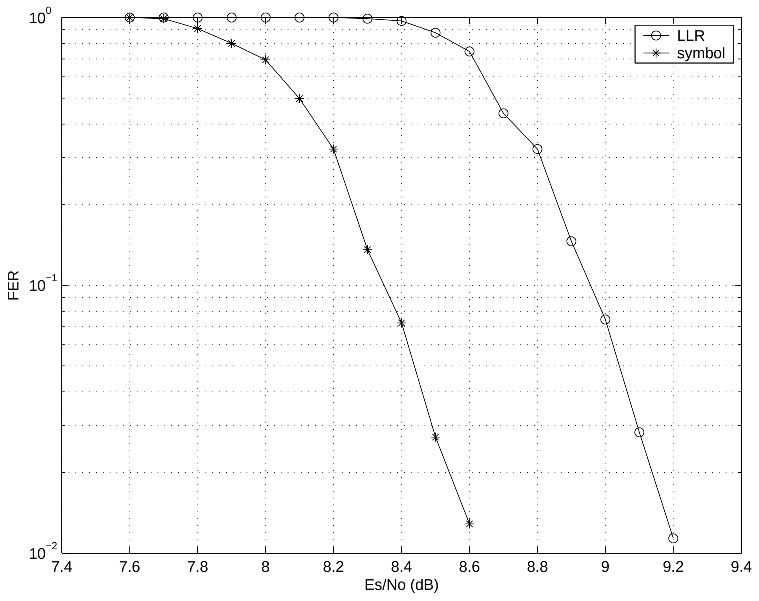


Figure 4.5: LLR vs. Symbol combining for r=0.52, 64QAM, 2 frame combining, using CC over AWGN channel.

HARQ protocol of interest. Such a representation helps us in deriving a two-dimensional generating function of the packet transmission process. The state diagram and transform domain representation is shown in Fig. 4.11, where I is the initial state, A is the end state (acceptance), P_{ci} is the probability of successful ith retransmission, P_{Fi} is the probability of unsuccessful ith retransmission, N_i is the number of the transmitted blocks and T is the transmitted delay.

4.5 Numerical Results

The following figure is obtained by computer simulation in which we have assumed that (i) infinite buffer size is available, (ii) the feedback channel is error-free, (iii) TDD mode of IEEE.16e is used and (iv) perfect channel estimation.

Fig. 4.12 and 4.13 display the comparisons of throughput and average transmit attempts over AWGN channel. It is clear that the throughput of each modulation scheme saturates at a level determined by the corresponding code rate and modulation order. The proposed adaptive method is the combination of 3 kinds of modulations in

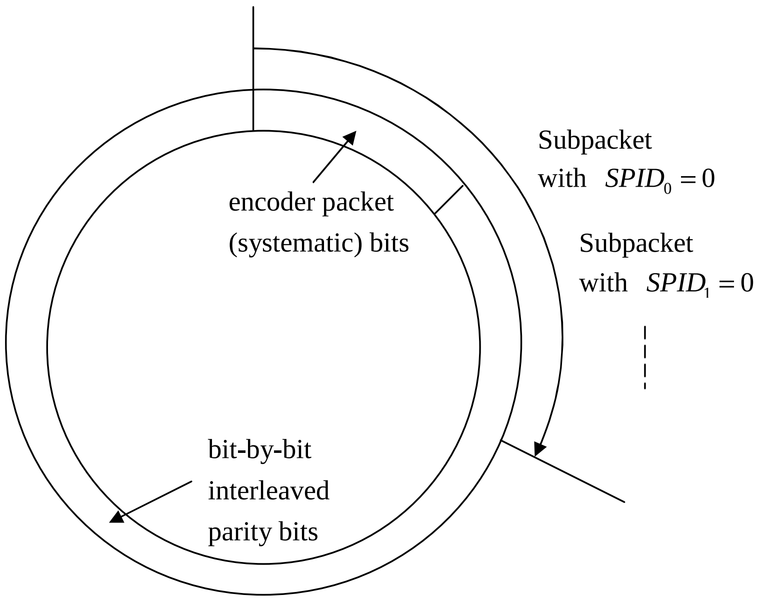


Figure 4.6: Chase Combining.

fact. No matter how channel's condition is, it can perform well. The average transmit attempts represent the delay before successful transmission. In most of the case, using adaptive method, the transmitter needs to transmit 1.2 times per packet in average, which is much less than 16QAM and 64QAM at low SNR.

Fig. 4.14 and 4.15 compare the throughput and average transmit attempts over flat Rayleigh fading channel. The results are similar to the case of AWGN.

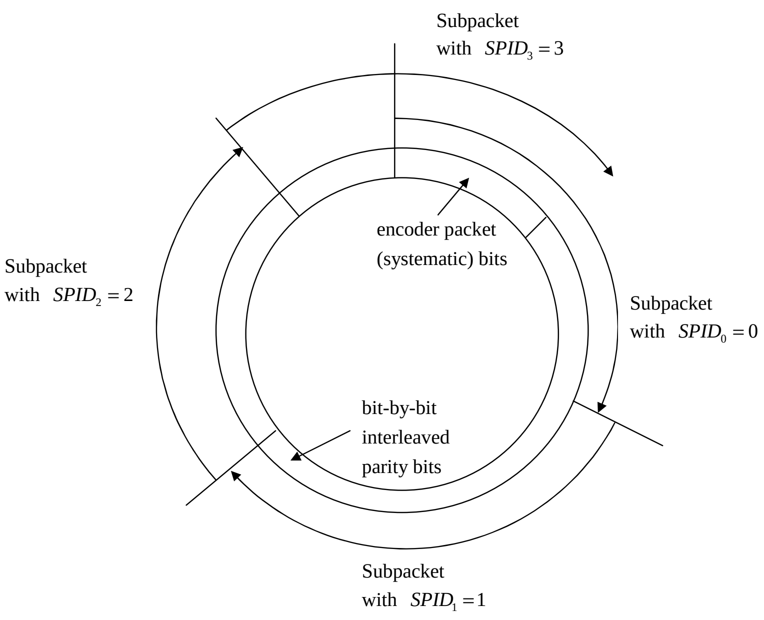

Subpacket with $SPID_3 = 3$
encoder packet (systematic) bits
Subpacket with $SPID_2 = 2$
Subpacket with $SPID_0 = 0$
bit-by-bit interleaved parity bits
Subpacket with $SPID_1 = 1$

Figure 4.7: Incremental redundancy (transmitted in order).

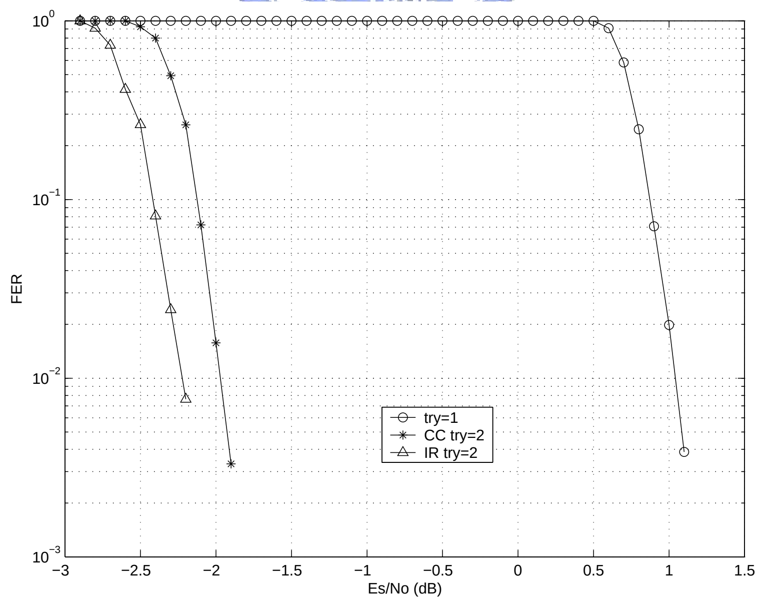


Figure 4.8: CC vs IR for QPSK, AWGN channel.

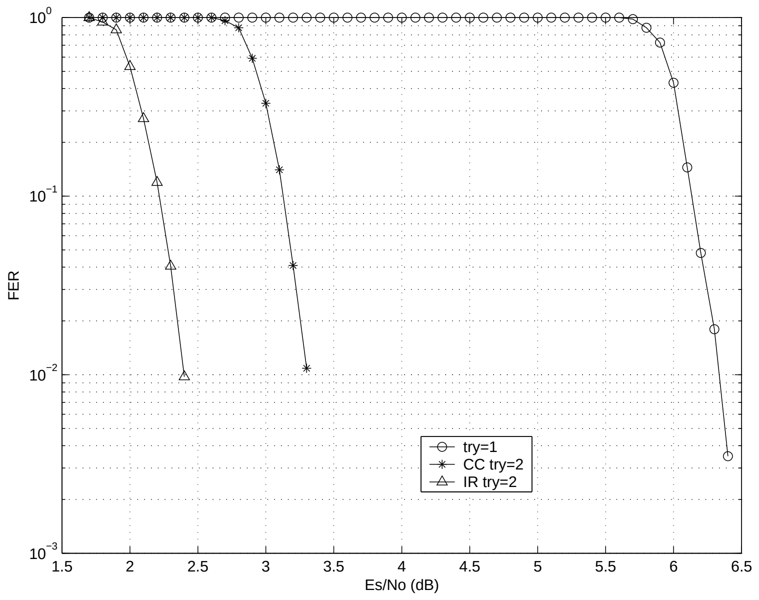


Figure 4.9: CC vs IR for 16QAM over AWGN channel.

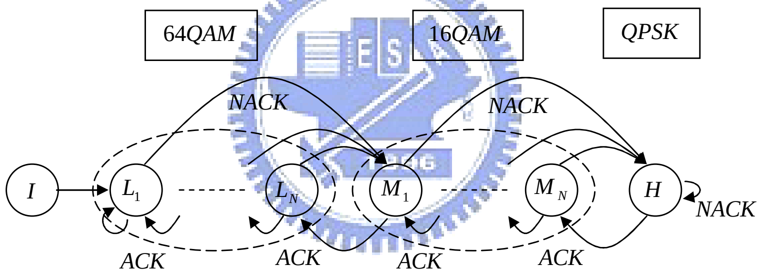


Figure 4.10: transition diagram for the proposed adaptive HRQ method.

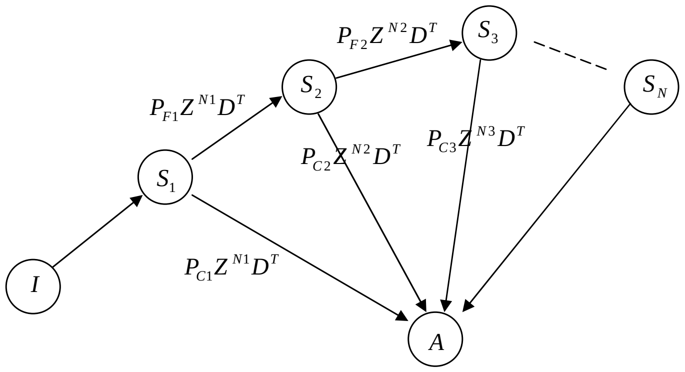


Figure 4.11: state diagram and transform domain representation.

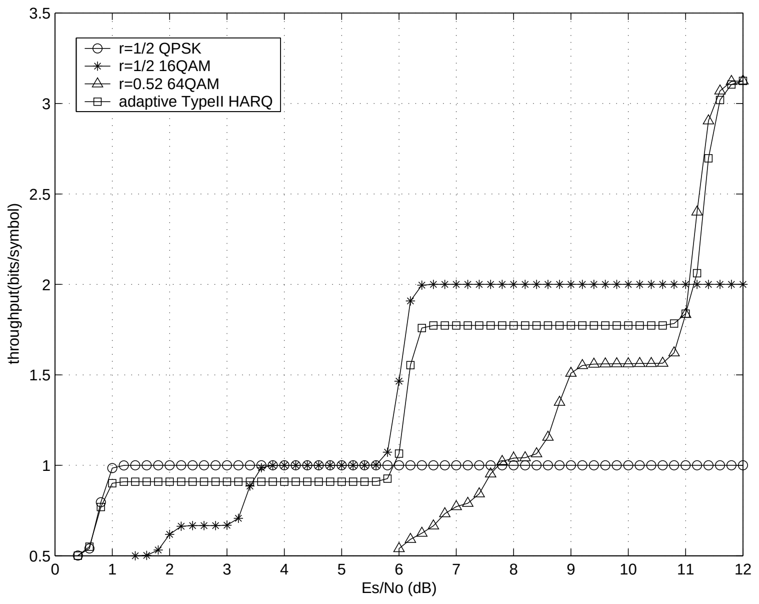


Figure 4.12: throughput comparison over AWGN channel.

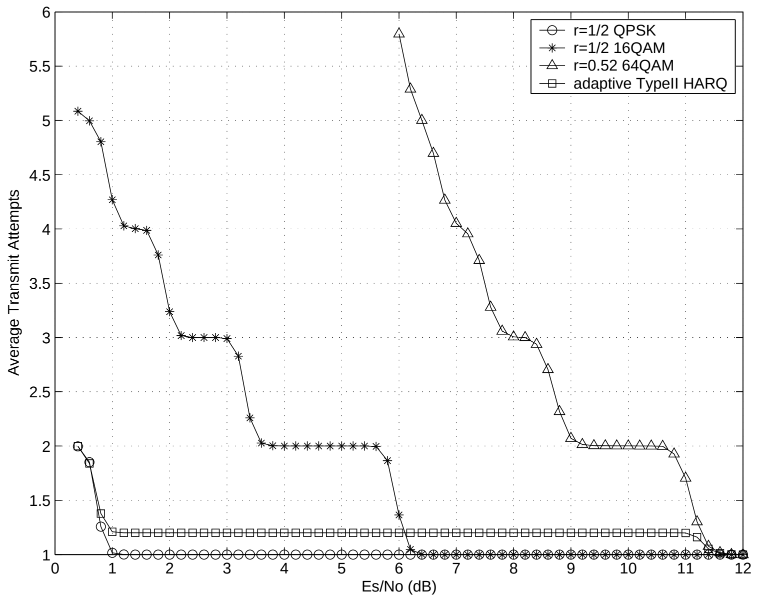


Figure 4.13: average transmit attempts over AWGN channel.

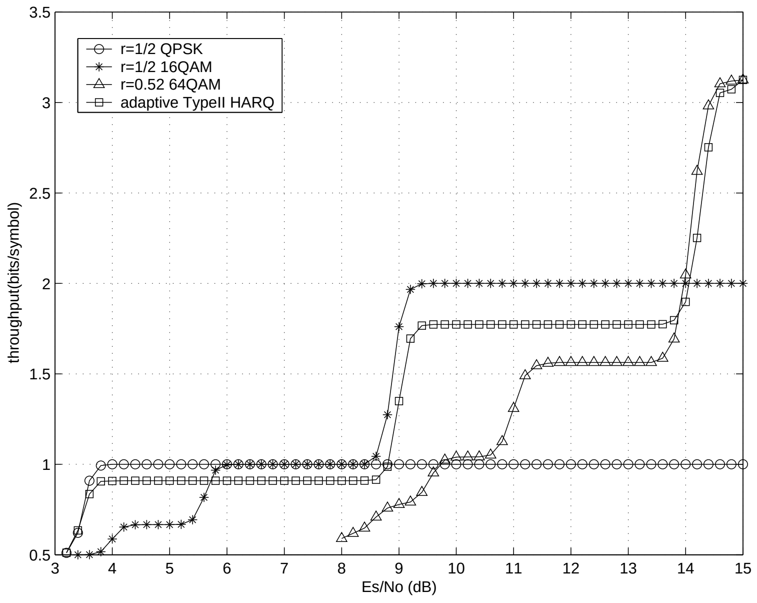


Figure 4.14: throughput comparison over flat Rayleigh fading channel.

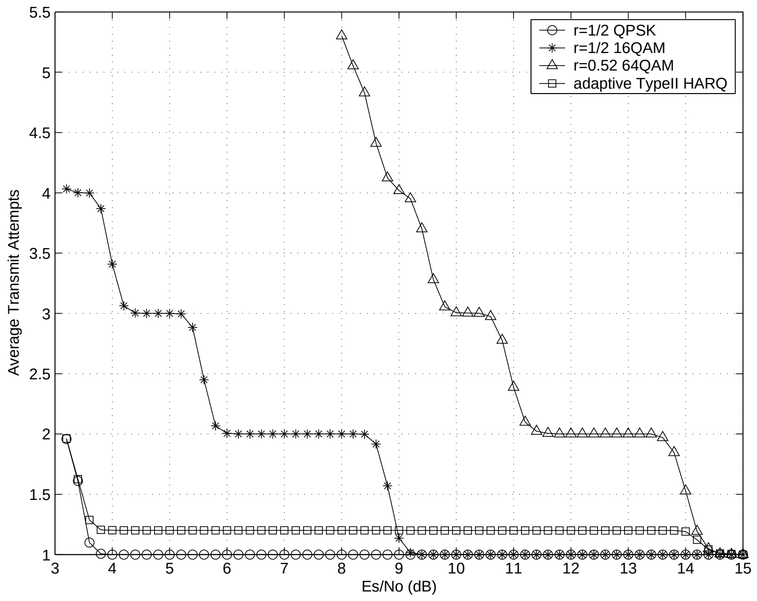


Figure 4.15: average transmit attempts over flat Rayleigh fading channel.

Chapter 5

Conclusion

We have analyzed the throughput and delay performance of adaptive Type II hybrid ARQ protocols. Two CC methods, namely LLR-based and symbol-based are investigated. The symbol-based CC provides better performance at the expense of increased complexity in memory and computing time. The comparison is based on a physical layer specification similar to that defined in the IEEE 802.16e standard with convolutional turbo code. Our simulation results indicate that IR is superior to CC for both QPSK and 16-QAM signals. Since the 802.16e standard makes it difficult to implement link adaptation with HARQ, we have loosened our assumption on fully compatible with the standard. It is found that performance is improved with the proposed link quality control mechanism.

The adaptive method used is a simple link quality indicator based on the number of consecutive ACKs or NACKs. More precise link quality indicator will surely enhance the system performance. Similarly, more flexible modulation and coding options will lead to higher throughput and lower latency. For an OFDMA cellular system, when the channel (subcarrier) conditions measured by the mobile terminals become available to the base station, adaptive channel assignment and scheduling along with more flexible HARQ are called for to maximize the overall system performance. In short, there are many interesting issues and extensions of our work remain unanswered, awaiting for future researchers' imaginations and devotions.

Bibliography

[1] S. Lin and D. J. Costello, Jr., *Error Control Coding : Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice Hall, 1983

[2] F. Babich, E. Valentinuzzi, and F. Vatta, "Performance of hybrid ARQ schemes for the LEO satellite channel", *Proc. IEEE GLOBECOM 2001*, San Antonio, TX, vol. 4, pp.2709-2713, Nov.2001

[3] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes", *IEEE Trans. Commun.*, vol. 44, no. 10 pp. 1261-1271, Oct. 1996

[4] D. Divalar and F. Pollara, "Multiple Turbo codes for deepspace communications", *JPA TDA Progress Reports*, vol. 42, pp. 66-77, May 1995

[5] D. Divalar and F. Pollara, "Turbo codes for PCS applications", *Proc. IEEE ICC'95*, Seattle, WA, vol. 1, pp. 54-59, June 1995

[6] D. Chase, "Code combining - A maximum likelihood decoding approach for combining an arbitrary number of noisy packets", *IEEE Tran, on Commun.*, vol. 38, No. 8, Aug. 1990

[7] S. Kallel, "Analysis of a Type II Hybrid ARQ Schemes with code combining", *IEEE Journal on selected Area in Commun.* vol.Sac-2 No. 4, July 1984

[8] Yingzi Gao, Soleymani, M.R., "Triple-binary circular recursive systematic convolutional Turbo codes", *the 5th International Symposium on Wireless personal Multimedia Communications*, Volume 3, 27-30 Oct. 2002 Page(s):951 - 955 vol.3

[9] C. Zhan, T.Arslan, A. T. Erdogan, S. MacDougall, “An efficient decoder scheme for double binary circular turbo codes” Vololume 4, 2006 Page(s):IV - IV Digital Object Identifier 10.1109/ICASSP.2006.1660947

[10] D. Molkdar, W. Featherstone and S. Lambotharan, “An overview of EGPRS: the packet data component of EDGE”

作 者 簡 歷

龔炳全，臺北市人，1983 年出生

臺北市立建國高級中學	1998/09 ~ 2001/06
國立中正大學電機工程學系	2001/09 ~ 2002/06
國立交通大學電信工程學系	2002/09 ~ 2005/06
國立交通大學電信工程學系系統組	2005/09 ~ 2007/07

Graduate Course:

1. Coding Theory
2. Spread Spectrum Communications
3. Adaptive Signal Processing
4. Digital Communications
5. Digital Signal Processing
6. Detection and Estimation Theory
7. Receiver Technology
8. Wireless Communications and Signal Processing