VLSI Signal Processing

Final Project Report

Turbo – code Encoder and Decoder –
Fundamentals and VLSI Implementation Issues
Chapter 1 Introduction

Channel coding introduced in the digital communication is to increase the reliability of data transmission in the noisy channel. Since Turbo coding was proposed in 1993, its amazing channel capacity near Shannon limit has drawn the digital implementation in the mobile radio to combat the multipath fading introduced in the wireless communication environment.

Many materials have discussed Turbo coding. Except the first paper of the proposal of Turbo codes [3] in 1993, the original decoding algorithm of Turbo codes can be found from [2], which suggests the optimal decoding of linear codes by estimating the a posteriori probabilities of the states and transitions of a Markov process observed in a discrete memoryless channel. This decoding algorithm called BAHL et. al. algorithm can be applicable to the NSC (Non – systematic convolutional code), but will not be suitable in RSC (Recursive systematic code) as the basic elements in Turbo encoder. Therefore, C. Berrou, A. Glavieux, P. Thitimajshima [3] modified this decoding algorithm to consider the logarithm of the likelihood ratio (LLR) to determine the transmitted information bits with respect to the observed noisy systematic symbol and redundant symbol.

In this final report, we will review the fundamentals of Turbo encoder and decoder in chapter 3 and chapter 4. Chapter 2 will review the preliminaries of mathematical backgrounds required for the subsequent chapters. In chapter 5, we will investigate several important issues of the design and implementation of Turbo encoder and decoder in VLSI digital circuits. In the aspect of the hardware implementation, the simplified modified BAHL et. al. algorithm cannot be implemented into circuits directly due to the hardware cost caused by the complicated multiplications and additions required in the original algorithm. Therefore, we consider the Max – log transformation of this algorithm to simplify the complexity of the decoder design. Power down techniques introduced in Turbo decoder will save the power consumption of the decoder implementation, improving the data processing rate of Turbo decoder. Chapter 6 summarizes of the context of this final report and makes some discussions.
Chapter 2 Preliminaries

In this chapter, we will review some preliminaries required of mathematical backgrounds required for the subsequent chapters. Throughout this report, for simplicity, we assume the channel we discussed here is memoryless Gaussian channel. In other words, in the receiver end, the received signal is just corrupted by additive white Gaussian noise (AWGN). Optimal decision principles – maximum a posteriori probability principle will be reviewed in section 1.1 and Max – log transformation with correcting terms will be reviewed in section 1.2.

Section 1.1 A posteriori probability (APP) and the logarithm of the likelihood ratio (LLR)

When the signals were transmitted through the channel, it will suffer from the attenuation, the dispersion, and the corruption of the additive white Gaussian noise. Therefore, in terms of this unknown statistical property of the received signal, what we can do is to estimate the probability of the transmitted symbol subject to the observed received signal. Optimal decision approaches discussed in digital communication books are maximum likelihood (ML) estimation and maximum a posteriori probability (MAP) estimation.

Here, we simply review the principle of MAP estimation. If a memoryless information source emits the information symbols $S = \{S_0, S_1, \ldots, S_{J-1}\}$, and these symbols will be encoded, modulated, and transmitted though a discrete memoryless Gaussian channel, in the receiver end, at the time instant $k$, we observe the received signal $R_k$.

We would like to make the optimal decision to determine the transmitted symbol $S_k$ subject to this observed received signal by estimating

$$\text{Pr}\{S_k = S_j | R_k\} \quad j = 0, 1, \ldots, J - 1 \quad (2 - 1)$$

If $\text{Pr}\{S_k = S_m | R_k\} > \text{Pr}\{S_k = S_j | R_k\}$ where $m \neq j$, then we make the optimal decision such that $\hat{S}_k = S_m$.

In Viterbi algorithm, a decoding window $N$ of the Viterbi decoder must be specified. After $N$ iteration steps, this algorithm will select the survivor path with the minimum metrics as the estimated code word. It will seek the optimal code word, but
cannot minimize the error of each bit in this optimal code word. Turbo decoding algorithm also considers the observed signals in a frame with the size N, but estimates the APP of the information bit \( b_k \) at the time instant \( k \) subject to these observed received signals in a frame.

Assume the correct synchronization of the frame, the received frame vector \( \mathbf{R}_1^N \) can be expressed as

\[
\mathbf{R}_1^N = [ \mathbf{R}_1 \mathbf{R}_2 \ldots \mathbf{R}_k \ldots \mathbf{R}_N ] \quad (2 - 2)
\]

where \( \mathbf{R}_k \) is the received vector at the time instant \( k \)

Since we would like to make the optimal decision of \( b_k \) in terms of APP of \( b_k \) subject to the received frame vector, we can define a logarithm of the likelihood ratio (LLR) of \( b_k \)

\[
\hat{E}(b_k) = \ln \frac{\Pr\{ b_k = 1 \mid \mathbf{R}_1^N \}}{\Pr\{ b_k = 0 \mid \mathbf{R}_1^N \}} \quad (2 - 3)
\]

If \( \Pr\{ b_k = 1 \mid \mathbf{R}_1^N \} \geq \Pr\{ b_k = 0 \mid \mathbf{R}_1^N \} \) then we determine \( \hat{b}_k = 1 \), otherwise we determine \( \hat{b}_k = 0 \). According to the fact of the natural logarithm function,

\[
\log_e(x) \geq 0 \text{ when } x \geq 1 \\
\log_e(x) < 0 \text{ when } 0 < x < 1 \quad (2 - 4)
\]

This MAP principle can be mapped into \( \hat{E}(b_k) \) by

\[
\hat{b}_k = 1 \text{ when } \hat{E}(b_k) \geq 0 \\
\hat{b}_k = 0 \text{ when } \hat{E}(b_k) < 0 \quad (2 - 5)
\]

The introduction of LLR of \( b_k \) will be used later when we discuss Turbo decoding algorithm in the subsequent chapters.
Section 1.2 Max – log transformation with correcting terms

The intention of the invention of the logarithmic algebra is to transform the operation of multiplication into addition, and the division into subtraction to simplify the complicated operations of multiplications and divisions found in numerical computations. Consider the function of $K$ variables denote $\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K$,

$$f(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K) = \hat{O}\exp(\bar{a}_i) \quad (2 - 6)$$

and the logarithmic version of $f(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K)$

$$F(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K) = \ln [f(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K)] \quad (2 - 7)$$

From the $K$ variables $\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K$, we can find the maximum $\bar{a}_M$, and (2 - 7) can be further rewritten as

$$F(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K) = \ln \{\exp(\bar{a}_M) + \exp(\bar{a}_M) \{\hat{O}\exp(\bar{a}_i - \bar{a}_M)\}\} \quad (2 - 8)$$

If $\bar{a}_M \gg \bar{a}_i$, we can see the best approximation of $F(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K)$, which will be

$$F(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K) \sim \max (\bar{a}_i) = \bar{a}_M \quad (2 - 9) \quad \{\bar{a}_i : 1 \leq i \leq K\}$$

We can define another notation $\max^*$ which symbolizes

$$\max^* (\bar{a}_M) = \max (\bar{a}_i) + \ln [1 + \hat{O}\exp(\bar{a}_i - \bar{a}_M)] \quad (2-10) \quad \{\bar{a}_i : 1 \leq i \leq K\} \quad \{\bar{a}_i : 1 \leq i \leq K\} \quad \bar{a}_i \not= \bar{a}_M$$

According to (2-10), (2-8) can be represented in terms of this introduced notation,

$$F(\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_K) = \max^* (\bar{a}_M) \quad (2 - 11) \quad \{\bar{a}_i : 1 \leq i \leq K\}$$
The Max – log transformation is primarily based on (2 – 11). Note that when the signal – to – noise ratio (SNR) is high, the approximation of (2 – 9) may have the good performance. But the performance of the approximation of (2 – 9) will degrade in the low SNR. In the wireless communication environment, the SNR of the received signal will be always low due to the channel fading and attenuation. Therefore, we cannot omit the correcting terms, and the correcting terms can be easily implemented into the small look-up table since the correcting terms only depend on the difference $(\bar{a}_1 - \bar{a}_M)$.

(2 – 10) can be generalized into this representation,

$$\max^*(x, y) = \max(x, y) + \ln[1 + \exp(-|x - y|)] \quad (2 – 12)$$

(2 – 12) will be useful when we consider the cell – based design of Turbo decoder in chapter 5.
Chapter 3 Fundamentals of Turbo Encoder

In this chapter, we will investigate the architecture of Turbo encoder – a parallel concatenation of two recursive systematic convolutional (RSC) codes. We will also compare the traditional non-systematic convolutional (NSC) codes with the recursive systematic convolutional (RSC) codes in several respects – similarity, statistical properties and error performance.

Section 3.1 NSC codes and RSC codes

Consider a NSC encoder with the generator matrix \( G(D) = [G_0(D) \ G_1(D)] \), its input has been concatenated with a feedback loop, and feed the information bit sequence \( \{b_k\} \) with the frame length \( N \) into the feedback loop.

\[
A(D) = \sum a_k D^k = \frac{1}{B(D)}
\]

\[
B(D) = 1 + D^2
\]

where \( A(D) = \sum a_k D^k \)

\( B(D) = \sum b_k D^k \)

Since \( [C_0(D) \ C_1(D)] = A(D) \ [G_0(D) \ G_1(D)] \), take (3-1) into account, we can express \( [C_0(D) \ C_1(D)] \) in terms of \( B(D) \),

\[
[C_0(D) \ C_1(D)] = B(D) \left[ \frac{G_0(D)}{(1 + D^2)} \quad \frac{G_1(D)}{(1 + D^2)} \right] \quad (3-2)
\]
where \( C_0(D) = \hat{0}C_{0,k}D^k \)
\( C_1(D) = \hat{0}C_{1,k}D^k \)

If we would like to make the NSC encoder concatenated with the feedback loop a systematic encoder, we may choose \( G_0(D) = 1 + D^2 \) to generate the coded bit sequence \( \{C_{0,k}\} \) directly sourcing from the information bit sequence \( \{b_k\} \). The newly proposed encoder illustrated in Figure 3 – 1 also possesses the recursion property because there exists a loop feedback from the coded bit sequences to the input when we consider the bit sequence \( \{a_k\} \) really affects the output of the NSC encoder.

Rewrite (3 – 2) when we choose \( G_0(D) = 1 + D^2 \)

\[
[C_0(D) \ C_1(D)] = B(D) [1 \ G_1(D) / G_0(D)] 
\]

If we reorder the information bit sequence \( \{b_k\} \) in pseudo–random manner, it is possible to make this rational polynomial have infinite quotient.

\[
C_1(D) = B(D)G_1(D) / G_0(D) = 1 + \hat{0}w_kD^k \quad w_k = \{0, 1\} \quad (3 - 4)
\]

The weight of a coded sequence is counted as the number of 1’s of this coded sequence. In this case, we see that the number of the occurrence of the number of 1’s of this coded sequence \( \{C_{1,k}\} \) may be as many as possible. This implies the increase of the performance of this convolutional code.

Assume that the occurrence of 1’s or 0’s of the information bit sequence \( \{b_k\} \) is likely equal, the statistical property of \( \{a_k\} \) can be found in terms of this relation

\[
a_k = b_k + a_{k-2} \quad (3-5)
\]

and

\[
Pr\{a_k = 0\} = Pr\{b_k = \hat{a} | a_{k-2} = \hat{a}\} = 1 / 2 \\
Pr\{a_k = 1\} = Pr\{b_k = \hat{a}^* | a_{k-2} = \hat{a}\} = 1 / 2
\]

where \( \hat{a} = \{0,1\} \) and \( \hat{a}^* \) means the binary inversion of \( \hat{a} \)

Therefore, the input sequence with respect to NSC encoder \( \{a_k\} \) has the same statistical property with the information bit sequence \( \{b_k\} \)
For convenience, we call this NSC encoder concatenated with the feedback loop recursive systematic convolutional (RSC) encoder. Since \( \{a_k\} \) and \( \{b_k\} \) have the same statistical properties, it is reasonable to declare that RSC encoder is equivalent to the NSC encoder.

**Section 3.2 The parallel concatenation of RSC encoders – Turbo encoder**

From section 3.1, we have learned that RSC encoder is equivalent to NSC encoder but has the better error correcting performance. In addition, randomized input information sequence \( \{b_k\} \) will drive the RSC encoder to generate coded bit sequence \( \{c_k\} \) with many 1’s to increase the weight of this coded bit sequence. Coding will increase unnecessary redundancy, however, puncturing mechanism will be invoked to preserve the desired transmission data rate.

In Figure 3-2, we propose this parallel concatenation architecture of Turbo encoder. We will observe that the information input sequence \( \{b_k\} \) will be fed into three paths – one is the systematic path, directly from the input to the output, into the RSC encoder (RSC\(_1\)) in the second path, and interleaved first (to be reordered in pseudo – random manner), as the input of another RSC encoder (RSC\(_2\)) in the third path. The output from two RSC encoders will be punctured (select one from two sources) to transmit the coded bits from RSC\(_1\) for \( n_1 \) times, and the coded bits from RSC\(_2\) for \( n_2 \) times.

![Figure 3-1 A parallel concatenation of RSC encoders – Turbo encoder](image-url)
Chapter 4 Fundamentals of Turbo Decoder

It may be exciting to discuss Turbo decoder and its related decoding algorithm – modified BAHL et.al. algorithm for RSC codes. Unlike Viterbi algorithm, which is maximum likelihood estimation, Turbo decoding algorithm is based upon a posteriori probability (APP) and LLR (the logarithm of the likelihood ratio). Therefore, in this chapter, we will discuss and derive simplified modified BAHL et. al. algorithm in advance. In the last section, we will investigate the serial concatenation of decoders – Turbo decoder.

Section 4.1 Simplified modified BAHL et. al. algorithm

Turbo encoder discussed in chapter 2 will encode two coded bits \( \{X_k, Y_k\} \) subject to the information bits \( \{b_k\} \). After modulation, these coded bits will be transmitted through a discrete memoryless Gaussian channel. At the time instant \( k \), the received vector \( R_k = [x_k, y_k] \),

\[
\begin{align*}
x_k &= (2X_k - 1) + n_I \\
y_k &= (2Y_k - 1) + n_Q
\end{align*}
\]

Where \( n_I \) and \( n_Q \) are two independent Gaussian random variables with zero mean, and the same variance \( \sigma^2 \).

Using Bayes rule, we may rewrite (2 - 3) as

\[
\mathbb{E}(b_k) = \ln \frac{\Pr\{ b_k = 1 , R^N_i \}}{\Pr\{ b_k = 0 , R^N_i \}} \quad (4 - 2)
\]

Consider a RSC encoder with the constraint length \( K \), at the time instant \( k \), the encoder state \( S_k \) may be one element selected from this integer set \( M = \{0, 1, 2, \ldots, 2^{K-1} - 1\} \). Take all possibilities of the value of \( S_k \) at the time instant \( k \) into consideration, (4 – 2) can be rewritten in the form of the joint probability of \( b_k, S_k \) and \( R^N_i \)

\[
\mathbb{E}(b_k) = \ln \frac{\mathcal{O}\Pr\{ b_k = 1, S_k = m, R^N_i \}}{\mathcal{O}\Pr\{ b_k = 1, S_k = m, R^N_i \}} \quad (4 - 3)
\]
We introduce a definition to simplify (4 – 3)

\[ \hat{e}^i_k(m) = \Pr\{ b_k = i, S_k = m, R_1^N \} \quad i = 0,1 \quad (4 – 4) \]

and (4 – 3) can be simplified as

\[ \hat{e}^i_k(m) = \ln \frac{\hat{E}(b_k)}{\hat{E}(b_k)} \quad (4 – 5) \]

Using Bayes rule, (4 – 4) can be rewritten as

\[ \hat{e}^i_k(m) = \Pr\{ b_k = i, S_k = m, R_1^N \} \]
\[ = \Pr\{ b_k = i, S_k = m, R_1^k, R_{k+1}^N \} \]
\[ = \Pr\{ R_{k+1}^N | b_k = i, S_k = m, R_1^k \} \Pr\{ b_k = i, S_k = m, R_1^k \} \]
\[ = \Pr\{ R_{k+1}^N | S_k = m \} \Pr\{ b_k = i, S_k = m, R_1^k \} \quad (4 – 6) \]

The received vectors after time instant k will not be influenced by the previous received vectors \( R_1^k \) and \( b_k \) when \( S_k = m \) is known, since the channel is memoryless, and the encoding process is modeled by Markov process.

For convenience, we define two probability functions,

\[ \hat{a}^i_k(m) = \Pr\{ b_k = i, S_k = m, R_1^k \} \]
\[ \hat{a}_k(m) = \Pr\{ R_{k+1}^N | S_k = m \} \]
\[ \quad (4 – 7) \]

Accordingly,

\[ \hat{e}^i_k(m) = \hat{a}^i_k(m) \hat{a}_k(m) \quad (4 – 8) \]

From (4 – 8), if we estimate the value of \( \hat{a}^i_k(m) \) and \( \hat{a}_k(m) \), \( \hat{e}^i_k(m) \) can be obtained by the multiplication of \( \hat{a}^i_k(m) \) and \( \hat{a}_k(m) \).

How to estimate \( \hat{a}_k(m) \) and \( \hat{a}_k(m) \) in the iteration manner? This problem can solved by introducing the transition of the encoder states in the trellis diagram into \( \hat{a}^i_k(m) \) and \( \hat{a}_k(m) \).
\[ \hat{a}_k(m) = \Pr \{ b_k = i, S_k = m, R_k^k \} \]
\[ = \Pr \{ b_{k-1} = j, b_k = i, S_{k-1} = m', S_k = m, R_k^k \} \]
\[ = \Pr \{ b_{k-1} = j, b_k = i, S_{k-1} = m', S_k = m, R_k, R_k^{k-1} \} \cdot \]
\[ = \Pr \{ b_{k-1} = j, b_k = i, S_{k-1} = m', S_k = m, R_k, R_k^{k-1} \} \cdot \]
\[ = \hat{a}_k(m') \]
\[ = \hat{a}_k(m', R_k, m, m') \cdot j_k^{k-1}(m') \]

(4 – 8)

where \( m' \) belongs to the integer set, which contains the possible values of \( S_{k-1} \) that will transit to \( S_k = m \) subject to this bit sequence \( b_k = i \) at the time instant \( k \), and \( \hat{a}_k(m', R_k, m, m') \) is defined as

\[ \hat{a}_k(m', R_k, m, m') = \Pr \{ b_k = i, S_k = m, R_k | S_{k-1} = m' \} \]  

(4 – 9)

\[ \hat{a}_k(m) = \Pr \{ R_{k+1}^N | S_k = m \} \]
\[ = \Pr \{ S_{k+1} = m', b_{k+1} = j', R_{k+1}^N, S_k = m \} \]
\[ = \Pr \{ S_{k+1} = m', b_{k+1} = j', R_{k+1}^N, R_{k+1}, S_k = m \} \cdot \]
\[ = \Pr \{ S_{k+1} = m', b_{k+1} = j', R_{k+1}, S_k = m \} \]
\[ = \Pr \{ S_{k+1} = m', b_{k+1} = j', R_{k+1}, S_k = m \} \]
\[ = \Pr \{ S_{k+1} = m', b_{k+1} = j', R_{k+1} | S_k = m \} \cdot \]
\[ = \Pr \{ S_{k+1} = m', b_{k+1} = j', R_{k+1} | S_k = m \} \]
\[ = \hat{a}_{k+1}(m') \cdot \hat{a}_k(m', R_k, m, m') \]
\[ = \hat{a}_{k+1}(m') \hat{a}_k(m', R_k, m, m') \]  

(4 – 10)

where \( m'' \) belongs to the integer set, which contains the possible values of \( S_{k+1} \) that will be visited by \( S_k = m \) when the bit sequence \( b_{k+1} = j' \).
From (4 – 8) and (4-10), we know the definition of $\tilde{a}_k^i(R_k, m, m')$ has occupied the important position that must be available in the iteration algorithm to compute $\hat{a}_k^i(m)$ and $\hat{a}_k(m)$.

$\tilde{a}_k^i(R_k, m, m')$ can be further rewritten as

$$
\tilde{a}_k^i(R_k, m, m') = \Pr \{ b_k = i, S_k = m, R_k | S_{k-1} = m' \} \\
= \frac{\Pr \{ b_k = i, S_k = m, S_{k-1} = m', R_k \}}{\Pr \{ s_{k-1} = m' \}} \\
= \frac{\Pr \{ R_k | b_k = i, S_k = m, S_{k-1} = m' \} \cdot \Pr \{ b_k = i | S_k = m, S_{k-1} = m' \} \cdot \Pr \{ S_{k-1} = m' \}}{\Pr \{ s_{k-1} = m' \}}.
$$

(4 – 11)

Since $R_k = [x_k \ y_k]$, and $x_k$ and $y_k$ are uncorrelated,

$$
\tilde{a}_k^i(R_k, m, m') = \Pr \{ x_k | b_k = i, S_k = m, S_{k-1} = m' \} . \Pr \{ y_k | b_k = i, S_k = m, S_{k-1} = m' \} . \Pr \{ b_k = i | S_k = m, S_{k-1} = m' \} . \Pr \{ S_{k-1} = m' \}.
$$

(4 – 12)

From (4 – 1), $x_k$ is the noisy version of the transmitted systematic signal, and $\Pr \{ b_k = i | S_k = m, S_{k-1} = m' \}$ is a deterministic value, which is 0 or 1. $\Pr \{ S_k = m | S_{k-1} = m' \} = \Pr \{ b_k = i \}$ since the branch transition depends on the information bit $b_k$.

We can summarize (4 – 12) into the following conclusion,

When the transition state $S_k = m, S_{k-1} = m'$ is given,

(a) Subject to this transition state, the assumption $b_k = i$ holds,

$$
\tilde{a}_k^i(R_k, m, m') = \Pr \{ x_k | b_k = i \} . \Pr \{ y_k | b_k = i, S_k = m, S_{k-1} = m' \} . \Pr \{ b_k = i \}.
$$

(b) If the constraint (a) does not hold

$$
\tilde{a}_k^i(R_k, m, m') = 0
$$

(4 – 13)
At last, we give the simplified BAHL et. al. algorithm. Initialize the boundary condition,

\[
\begin{align*}
\hat{a}_0^i(m) &= 1 \text{ if } m = 0, \hat{a}_0^i(m) = 0 \text{ if } m \neq 0 \quad \text{with respect to } i = 0,1 \\
\hat{a}_N(m) &= 1 \text{ if } m = 0, \hat{a}_N(m) = 0 \text{ if } m \neq 0,
\end{align*}
\]  

(4 – 14)

After the boundary condition is readily set, we compute \(\hat{a}_k^i(R_k, m, m')\) and \(\hat{a}_k^i(m)\) according to (4 – 13) and (4 – 8), respectively, when the received vector \(R_k = [x_k, y_k]\) arrived at the time instant \(k\).

When \(N\) received vectors have been completely received, we can compute \(\hat{a}_k(m)\) according to (4 –10) in a backward order in terms of \(\hat{a}_k^i(R_k, m, m')\). Then, we multiply \(\hat{a}_k^i(m)\) and \(\hat{a}_k(m)\) to obtain \(\hat{e}_k^i(m)\), which can be used to compute \(\hat{E}(b_k)\) from (4 – 5).

**Section 4.2 Simplified modified BAHL et. al. algorithm – Revisit and Extrinsic information**

From (4 – 8), we can isolate \(\Pr\{x_k | b_k = i\}\) from these double summation to obtain

\[
\hat{a}_k^i(m) = \Pr\{x_k | b_k = i\} \cdot \hat{a}_k^i(y_k, m, m') \cdot \hat{a}_{k-1}^i(m) \quad (4 – 15)
\]

From (4 - 10), we cannot isolate \(\Pr\{x_k | b_k = j'\}\) from these double summation, however, since \(\Pr\{x_k | b_k = j'\}\) associates with one summation with respect to \(j'\) starting from 0 to 1. From (4 – 5), we rewrite \(\hat{E}(b_k)\) in another complex form,

\[
\hat{E}(b_k) = \ln \frac{\hat{a}_k^1(m) \hat{a}_k^0(m)}{\hat{a}_k^0(m) \hat{a}_k^0(m)}
\]

\[
\Pr\{x_k | b_k = 1\} \cdot (\hat{a}_k^1(y_k, m, m') \cdot \hat{a}_{k-1}^1(m)) = \ln \frac{(\hat{a}_k^1(x_k, y_k, m, m') \cdot \hat{a}_{k-1}(x_k, y_k, m, m'))}{(\hat{a}_k^0(x_k, y_k, m, m') \cdot \hat{a}_{k-1}(x_k, y_k, m, m'))}
\]

\[
\Pr\{x_k | b_k = 0\} \cdot (\hat{a}_k^0(y_k, m, m') \cdot \hat{a}_{k-1}^0(m)) = \ln \frac{(\hat{a}_k^0(x_k, y_k, m, m') \cdot \hat{a}_{k-1}(x_k, y_k, m, m'))}{(\hat{a}_k^1(x_k, y_k, m, m') \cdot \hat{a}_{k-1}(x_k, y_k, m, m'))}
\]

\[
= I x_k + W_k \quad (4 – 16)
\]
where

$$I_{x_k} = \ln \frac{\Pr\{x_k \mid b_k = 1\}}{\Pr\{x_k \mid b_k = 0\}}$$

$$W_k = \ln \frac{\hat{a}_k}(\hat{a}_k(y_k, m, m') \hat{a}_{k-1}(m)) \cdot (\hat{a}_{k+1}(m') \hat{a}_{k+1}(x_k, y_k, m, m'))}{\hat{a}_k(y_k, m, m') \cdot (\hat{a}_{k+1}(m') \hat{a}_{k+1}(x_k, y_k, m, m'))}$$

From (4 – 1), $x_k$ is a Gaussian random variable with mean 1 (if $b_k = 1$) or –1 (if $b_k = 0$) and variance $\sigma^2$. $I_{x_k}$ can be determined by considering this limit

$$I_{x_k} = \ln \frac{\Pr\{x_k \mid b_k = 1\}}{\Pr\{x_k \mid b_k = 0\}} = \lim_{\Delta x \to 0} \ln \frac{f_1(x_k)\Delta x}{f_2(x_k)\Delta x} = \frac{2}{\sigma^2} (4 – 17)$$

where

$$f_1(x) = \frac{1}{(2\sigma^2)^{1/2}} \exp\left(-\frac{(x - 1)^2}{2\sigma^2}\right) \quad f_2(x) = \frac{1}{(2\sigma^2)^{1/2}} \exp\left(-\frac{(x + 1)^2}{2\sigma^2}\right)$$

$y_k$, the noisy version of the redundant information introduced by Turbo encoder, as the parameter of the function $W_k$. In general,

$$W_k \geq 0 \text{ when the information bit } b_k = 1$$
$$W_k < 0 \text{ when the information bit } b_k = 1$$

This extrinsic information $W_k$ will help improve the performance of the decoder because it will enhance $E(b_k)$ to combat the bit error resulted from the channel noise. Thus, we may conclude that

- The redundant information introduced by Turbo encoder will bring the extrinsic information in the decoder to improve the performance of the decoder.
- More redundant information, will improve the performance of the decoder by more introduced extrinsic information.
Section 4.3 The serial concatenation of decoders – Turbo decoder

From section 4.2, we know that more redundant information introduced by Turbo encoder will bring more extrinsic information for the decoder to make better decision. In Figure 4 - 1, we will see the serial concatenation of two decoders. Modified BAHL et. al. algorithm will be implemented in these decoders. Like Turbo encoder with the puncturing mechanism, Turbo encoder also introduces another similar function – demultiplex / insertion block, which will demultiplex the noisy version of the redundant information $y_k$ into decoder 1 when $Y_k = Y_{1,k}$, or decoder 2, when $Y_k = Y_{2,k}$. When the redundant information of a given encoder (RSC_1 or RSC_2) is not emitted, the corresponding decoder input is set to analog zero.

In Figure 4 – 1, we will find a loop feedback from the decoder 2 as the input of the decoder 2. This intention will indeed introduce another redundant information from $Y_{2,k}$ because the decoder 1 does not directly receive the noisy version of $Y_{2,k}$. The output of the decoder 2 will be interleaved first in order to make the signal feedback from the decoder 2 more uncorrelated with the received signals $x_k$ and $y_k$.

We can approximate the signal feedback from the decoder as a Gaussian random variable with variance $\hat{\sigma}_z^2$, and this signal denotes $z_k$. With the extra input signal $z_k$, the decoder output $\hat{E}_1(b_k)$ will be

$$\hat{E}_1(b_k) = \frac{2}{\hat{\sigma}_x^2} x_k + \frac{2}{\hat{\sigma}_z^2} z_k + W_{1k} \quad (4 - 18)$$

Because $z_k$ is generated by the decoder 2, we cannot feed $\hat{E}_1(b_k)$ directly into the decoder 2 but the version of $\hat{E}_1(b_k)$ subject to $z_k = 0$. After deinterleaving, the output of the decoder 2 must remove the information introduced by $\hat{E}_1(b_k)$ because $\hat{E}_1(b_k)$ is generated by the decoder 1.

$$\hat{E}_1^*(b_k) = \hat{E}_1(b_k) \big|_{z_k = 0} \quad (4 - 19)$$
$$z_k = \hat{E}_2(b_k) \big|_{\hat{E}_1(b_k) = 0} \quad (4 - 20)$$

Through a memoryless slicer, we can determine the information bit sequence $\{b_{k}^*\}$ in the receiver end in terms of the output of decoder 2 after deinterleaving $\hat{E}_2(b_k)$. 
Figure 4 – 1 The serial concatenation of decoders – Turbo decoder
Chapter 5 VLSI implementation issues of Turbo encoder and decoder

Since we have reviewed the fundamentals of Turbo encoder and decoder in chapter 3 and chapter 4, respectively, we can focus on the VLSI implementation issues of Turbo encoder and decoder in this chapter. For the reason of the later use, we rewrite (4 – 8), (4 – 10), and (4 – 13) in the chapter, and denominate them sophisticated names.

- The forward recursion metric

\[
\hat{\alpha}_i^k(m) = \Pr \{ b_k = i, S_k = m, R^k \} = \hat{\alpha}_k^j(R_k, m, m') \hat{\alpha}_{k-1}(m') \quad (5 – 1)
\]

- The backward recursion metric

\[
\hat{\alpha}_k^j(m) = \Pr \{ R_{k+1} | S_k = m \} = \hat{\alpha}_{k+1}(m'') \hat{\alpha}_{k+1}(R_k, m, m'') \quad (5 – 2)
\]

- The branch metric

\[
\hat{\alpha}_i^j(R_k, m, m') = \Pr \{ b_k = i, S_k = m, R_k | S_{k-1} = m' \} = \Pr \{ x_k | b_k = i \} \cdot \Pr \{ y_k | b_k = i, S_k = m, S_{k-1} = m' \} \Pr \{ b_k = i \}
\]

if the assumption \( b_k = i \) holds subject to this transition state. \( (5 – 3) \)

Besides, in the real implementation of the decoder design, the input parameter of the decoder contains not only the received vector \( R_k = [x_k \ y_k] \) arrived at the time instant \( k \) but also the extrinsic information generated from the other decoder. As what has been discussed in section 4.3, the feedback extrinsic information is approximated as a random variable with normal distribution of variance \( \sigma^2_L \). Therefore, the notation \( R_k \) shown in (5 – 3) should consist of three elements with respect to the decoder \( n \),

\[
R_k = [x_k \ y_{n,k} \ W_{n',k}] \quad (5 – 4)
\]

where \( W_{n',k} \) represents the extrinsic information feedback from the decoder \( n' \).
Section 5.1 Trellis termination

In chapter 4, the simplified modified BAHL et. al. algorithm indicates that the initial transition state of RSC encoder will start at the zero position, after encoding for the coming information frame with the size N, the transition state will terminate at the zero position. This assumption will not be necessarily true when the frame size is not large enough due to the truncation. Therefore, we need to introduce some dummy bits (0’s) with the size K to force the transition state to terminate at the zero position with respect to the coming information frame.

From the structure of Turbo encoder illustrated in Figure 3 - 1, we learned that the second RSC encoder received the information bit sequence through the interleaver. If the coming information frame N (N – K information bits plus the K dummy zeros) forces the transition state of the first RSC encoder in the second path (see Figure 3-1) to start and terminate at the same zero position, we cannot show that this information frame will force the transition state of the second encoder to terminate at the zero position since the information frame has been interleaved in advance. As a consequence, we have to redefine the boundary condition of the decoders.

For the decoder 1 (which receives the noisy version of the redundant information emitted from encoder 1),

\[
\begin{align*}
\hat{a}_i^0(m) &= 1 \text{ if } m = 0, \hat{a}_i^0(m) = 0 \text{ if } m \neq 0 \quad \text{with respect to } i = 0,1 \\
\hat{a}_N(m) &= 1 \text{ if } m = 0, \hat{a}_N(m) = 0 \text{ if } m \neq 0, \quad (5 - 5)
\end{align*}
\]

But for the decoder 2 associating with the encoder 2,

\[
\begin{align*}
\hat{a}_i^0(m) &= 1 \text{ if } m = 0, \hat{a}_i^0(m) = 0 \text{ if } m \neq 0 \quad \text{with respect to } i = 0,1 \\
\hat{a}_N(m) &= \hat{a}_N^1(m) \quad \text{with respect to } i = 0,1 \quad (5 - 6)
\end{align*}
\]

where we can note that we left the trellis of the encoder 2 “open”. The initial value of the backward recursion metric uses the value of the forward recursion metric at the last iteration step.
Section 5.2 Max – log transformation with correcting terms of simplified modified BAHL et. al. algorithm

We reconsider the logarithmic version of the forward recursion metric (5 – 1), the backward recursion metric (5 – 2), and the branch metric (5 – 3),

\[ A_k^i(m) = \ln [\tilde{a}_k^i(m)] \]  
\[ B_k(m) = \ln [\tilde{a}_k(m)] \]  
\[ \bar{A}_k(R_k, m, m') = \ln[\tilde{a}_k^i(R_k, m, m')] \]  

From (5 - 7), (5 – 8), and (5 – 9), we can derive \( A_k^i(m) \) and \( B_k(m) \) as

\[ A_k^i(m) = \ln [\tilde{a}_k^i(m)] \]
\[ = \ln [\tilde{a}_k^i(R_k, m, m') \tilde{a}_k^{i-1}(m')] \]
\[ = \ln \{ \tilde{a}_k^i(R_k, m, m') \} \]
\[ = \ln \{ \tilde{a}_k^i(R_k, m, m') + \tilde{a}_k^{i-1}(m') \} \]
\[ = \ln \{ \exp [\bar{A}_k(R_k, m, m') + A_k^{i-1}(m)] \} \]  

\[ (5 – 10) \]

Similarly,

\[ B_k(m) = \ln \{ \tilde{a}_k^i(R_k, m, m') + \tilde{a}_k^{i-1}(m') \} \]  

\[ (5 – 11) \]

According to (2 – 12), (5 – 10) and (5 – 11) can be reformulated as

\[ A_k^i(m) = \ln \{ \tilde{a}_k^i(R_k, m, m') + A_k^{i-1}(m) \} \]
\[ = \max \{ \max [\tilde{a}_k^i(m'), \tilde{a}_k^{i-1}(m')] \} \]

\[ (5 – 12) \]

where \( \tilde{a}_k^i(m') = \bar{A}_k(R_k, m, m') + A_k^{i-1}(m), \tilde{a}_k^{i-1}(m') = \bar{A}_k(R_k, m, m') + A_k^{i-1}(m) \)

Similarly,

\[ B_k(m) = \max \{ \max [\bar{A}_k(R_k, m, m'), \bar{A}_k^{i-1}(m')] \} \]

\[ (5 – 13) \]

where \( \bar{A}_k(R_k, m, m') + B_k^{i+1}(m'), \bar{A}_k^{i+1}(R_k, m, m') + B_k^{i+1}(m') \)
From (5 – 12) and (5 – 13), we replace the complicated multiplications and additions required to compute the forward recursion metric and backward recursion metric with the simple compare and select operation, suitable for the implementation of VLSI circuits.

The critical computation unit is the branch metric. Consider the transmitter, at the time instant \( k \), for the encoder \( n \), when the current encoder state \( S_k = m \), and the previous encoder state \( S_{k-1} = m' \), subject to the information bit \( b_k = i \), through binary modulation, the baseband representation of the transmitted signal can be shown by

The systematic part, \( x_k = 2b_k - 1 \),

The redundant part, \( y_{n,k} = 2Y_{n,k} - 1 \), subject to \( b_k = i \), \( S_k = m \), and \( S_{k-1} = m' \)

These signals will be transmitted through a memoryless Gaussian channel. In the decoder \( n \), the received signals, \( x_k, y_{n,k} \), which are the noisy version of the systematic symbol \( x_k \) and the redundant symbol \( y_{n,k} \), respectively, can be assumed to have the normal distribution, with the variance \( \sigma^2 \), associating with the memoryless Gaussian channel.

Then, according to (5 – 3) and (5 – 9)

\[
\Pr\{ x_k \mid b_k = i \} = \frac{1}{\sqrt{(2\pi\sigma^2)^2}} \exp\left(-\frac{(x_k - x_k)^2}{2\sigma^2}\right)
\]

\[
\Pr\{ y_{n,k} \mid b_k = i, S_k = m, S_{k-1} = m' \} = \frac{1}{\sqrt{(2\pi\sigma^2)^2}} \exp\left(-\frac{(y_{n,k} - y_{n,k})^2}{2\sigma^2}\right)
\]

\[
\Pr\{ W_{n'.k} \mid b_k = i \} = \frac{1}{\sqrt{(2\pi\sigma^2_L)^2}} \exp\left(-\frac{(W_{n'.k} - x_k)^2}{2\sigma^2_L}\right)
\]

\[(5 – 14)\]
and from (5 – 8)

\[
\bar{A}_k(R_k, m, m') = \ln[\bar{a}_k(R_k, m, m')] \\
= \ln[\Pr\{x_k | b_k = i\}] + \\
\ln[\Pr\{W_{n',k} | b_k = i\}] + \\
\ln[\Pr\{y_k | b_k = i, S_k = m, S_{k-1} = m'\}] + \\
\ln[\Pr\{b_k = i\}] \\
\tag{5 – 15}
\]

From the appendix, we can show that

\[
\ln[\Pr\{x_k | b_k = i\}] = \frac{x_k}{\bar{\sigma}^2} \\
\ln[\Pr\{W_{n',k} | b_k = i\}] = \frac{W_{n',k}}{\bar{\sigma}_L^2} \\
\ln[\Pr\{y_k | b_k = i, S_k = m, S_{k-1} = m'\}] = \frac{y_{n,k}}{\bar{\sigma}^2} \\
\ln[\Pr\{b_k = 1\}] = W_{n',k} - \max^*(0, W_{n',k}) \\
\ln[\Pr\{b_k = 0\}] = - \max^*(0, W_{n',k}) \\
\tag{5 – 16}
\]

where we neglect the constant factors to simplify the computation.

From (5 – 15) and (5 – 16), we have to estimate the channel variance \(\bar{\sigma}^2\) and the variance \(\bar{\sigma}_L^2\) of the extrinsic information feedback from the other decoder to compute the branch metric. We can assume that the channel variance \(\sigma^2\) preserves constant during the observation time of a frame, (the assumption may work well if the period of an information frame is less than the coherent time, which is beyond the scope of this report), such that we can divide \(\sigma^2\) of both sides of (5 – 15), and we can obtain

\[
\bar{A}_k(R_k, m, m') = x_k + \frac{x_k}{\bar{\sigma}^2} + \frac{W_{n',k}}{\bar{\sigma}_L^2 / \bar{\sigma}^2} + \ln[\Pr\{b_k = i\}] \\
\tag{5 – 17}
\]
In [5], it suggests that we can introduce a factor $Q$ to represent $\frac{\partial L}{\partial \hat{\theta}}$, and initialize $Q$ as a large value. Since the observed sample $W_{n',k}$ increases as the increase of the iteration, the variance of $W_{n',k}$ will decrease. This decline coefficient can be chosen as 2 for each iteration, and this operation can be easily implemented with the shifters. The final term of (5 - 17), $\ln \{Pr\{ b_k = i\}\}$, may be omitted, where we assume that the occurrence of 1's or 0's is likely equal, but this will degrade the performance of this decoder because the occurrence of 1's or 0's in an observed information frame is not likely equal. Therefore, we have a tradeoff of the reduction of the hardware cost and the improvement of the decoder performance. The improvement of the decoder performance will not cost the hardware implementation very much, however, since the estimation of $\ln \{Pr\{ b_k = i\}\}$ can be implemented into circuits using the simple compare and select operation, and addition units.

**Section 5.3 The cell structure of the recursion metric**

In the view of VLSI architectures, (5 – 10) constitutes the unit performing the computation of all forward recursion metrics. We can draw the block diagram of (5 – 10) in Figure 5 – 1, and this architecture is called á cell.

![Diagram of a cell](image_url)

**Figure 5 – 1 The architecture of á cell, which computes the forward recursion metric**

The lattice architecture of á cell can reduce the routing area. The component sign detector will decide whether the difference $\hat{a}^0_{k}(m') - \hat{a}^1_{k}(m')$ is positive or negative. The component ABS will compute the absolute value of the input signal. LUT is the look – up table, which stores the value for the correcting terms of max – log transformation. The output of the sign detector will control the multiplexer to select $\hat{a}^0_{k}(m')$ or $\hat{a}^1_{k}(m')$, and signal the ABS block to perform the absolute operation.
The hardware reduction of a cell may be amazing. The hardware of the multiplexer, the sign detector, and the ABS unit, can be implemented using the simple logic gates. Complicated multiplications and additions seen in (5 – 1) are avoided, but substituted four adders, the multiplexer, the sign detector, the ABS unit, and the small look-up table.

The block diagram of (5 – 12) has the same architecture with a cell except for the input signals of this cell. We call this block diagram â cell. á cell and â cell constitute the units performing the computation of all values of the forward recursion metrics and backward recursion metrics. Cell – based design is the design rule for the VLSI implementation of Turbo decoder.

Section 5.4 The power down technique

In order to guarantee the minimum bit error rate of the decoded bits, it is inevitable to iterate the serial concatenation of decoders – Turbo decoder illustrated in Figure 4 – 1, to strengthen the extrinsic information to help the decoder to make the better LLR associating the information bit \( b_k \) at the time instant \( k \) after several iterations. It requires the smart estimation of the number of the iteration to improve the decoder performance. It is still a trade-off problem because larger iteration times will dissipate more power, but provide the better performance.

The power down technique applicable to Turbo decoder is to save the power consumption but preserve the performance of the decoder. Several iterations will strengthen the extrinsic information, and if the extrinsic information has been beyond a threshold that guarantees the performance of the decoder, we should stop the iteration to save the power. This implies that if we compare the decoded bits in a frame of the current iteration with those of the previous iteration, if the Hamming distance (the number of the different bits between a decoded word) is less than a desired distance, (or say 0, to attain the situation of no errors), we can stop the iteration.

Based upon this argument, we can implement a power down decision unit with simple logic gates. During each iteration, we need to compare the decoded bit with the previous decoded bit for \( N \) times, where \( N \) is the frame size. If one is different from the other, we can use a counter to accumulate the number of different bits. After the iteration, what we do is to check whether the accumulation number of the counter is less than or equal to the desired threshold, (or say, no accumulation occurs during this
iteration). If this condition holds, we can stop the iteration without the degradation of
the decoder performance but the saving of more power. This adaptive design issue
may be better than the design of decoder with the constant iteration number.

Figure 5 – 2 illustrates the block diagram of power down decision unit. The
comparator can be easily implemented using the modulo – 2 adder (XOR logic gate),
to generate the signal to trigger the counter. After the iteration, the accumulation
result of the counter will be sent to the threshold comparator, to decide whether the
iteration stops or continues.

\[ E_2(b_k) \]

to the next buffer

from the previous buffer

\[ + \]

counter

Threshold
comparator

Iteration
stops or
continues

Figure 5 – 2 The block diagram of power down decision unit
Chapter 6 Summary and Discussion

Chapter 3 primarily states the derivation of RSC encoder can be thought as the concatenation a traditional NSC encoder with a feedback loop. After reading the contents of chapter 3, we have learned that the RSC encoder is equivalent to NSC encoder except that RSC encoder supplies the systematic bit sequence to help the decoder make the optimal decision based upon MAP principle, and more 1’s will occur in the coded sequence to increase the free distance of this coded sequence due to the randomized information bit sequence and rational generator polynomial.

If we have read [3] and [4], we may find it difficult to realize the decoding algorithm because we are not able to point out where these mathematical formulas come from and why they will look like this complex form. Therefore, I restudied the primitive decoding algorithm – BAHL et. al. algorithm [2], and gained the basic decoding principle -

For NSC codes, the systematic bit sequence will not be emitted. Hence, in the receiver end, if we would like to minimize the bit error rate when tracing the trellis diagram in terms of N received vectors, it is essential to find APP of the encoder state for a node in the trellis diagram, and APP of the transition state of the encoder for a branch in the trellis diagram. These two constraints become the basis of this decoding algorithm – BAHL et. al. algorithm.

For RSC codes, however, the systematic bit sequence will be emitted. Therefore, in the receiver end, we need to consider not only the two constraints remarked in the primitive decoding algorithm, but also the APP of the information sequence in terms of N received vectors.

This motivation drove me to derive the simplified version of modified BAHL et. al. algorithm. In [3] and [4], redundant terms of those mathematical formulas may not benefit the practical digital implementation. For those who first study this algorithm, they will wonder the mysterious derivation of this algorithm, but be not able to present the basic concept of this algorithm, or to explain this algorithm on their languages. Thus, in section 4.1 of this report, we will see another approach to present this modified BAHL et. al. algorithm in simple but readable manner. These derived mathematical formulas will be represented in the simplified form, but will be helpful when we apply the computer simulation of this algorithm, or implement this algorithm in the VLSI circuits.
Chapter 5 indicates the VLSI design issues of Turbo encoder and decoder. Although I did not outline the VLSI architecture of Turbo encoder and decoder, I still pointed out several VLSI design issues to be considered when we implemented Turbo encoder and decoder into circuits – the reduction of the hardware cost, the cell – based design architecture, and the saving of the power.

Chapter 3 and chapter 4 are the reviews of the fundamentals of Turbo encoder and decoder. Chapter 3 and chapter 4 primarily emphasize the theoretical domain of Turbo encoder and decoder, not mention the implementation constraints. The view of VLSI implementation of Turbo encoder and decoder can be found in Chapter 5, where the simplified modified BAHL et. al. algorithm has been transformed into the logarithmic domain, where the complicated multiplications and additions required in the computation of recursion metrics have been removed, but substituted the simple compare and select unit, the addition units, and the small look – up table.

In chapter 5, the design issues of Turbo encoder and decoder imply the combination of the communication theory and the experience of VLSI circuit design. In order to avoid the statistical estimation of the channel and the extrinsic information feedback from the other decoder, we use an approximated value to estimate the statistical property of the channel and the extrinsic information. The max – log simplification of the simplified modified BAHL et. al. algorithm has to be corrected with the correcting terms due to the low SNR of the received signal for the wireless applications. Cell – based architecture design of the decoder considers the structural design issue to reduce the routing area, and uplift the data processing rate. The power down technique not only preserves the performance of the decoder, but also considers the power consumption constraint of the low – power VLSI circuit design.

This report covers wide knowledge of digital transmission theory, coding theory, and the architecture design of VLSI signal processing. As the information makes progress with time goes by, we cannot confine our knowledge into a certain research interest, but receive the attacks from several aspects of knowledge domain. Mathematical skills are required when we have to simplify the algorithms into another form for the hardware reduction and the power saving of VLSI circuit implementations.
Appendix

Part I:

In this part, we would like to verify the final result of (5 – 12). Thus, we will prove the validity of this equality,

\[
\ln \{ \hat{\Omega} \{ \exp[\hat{a}_k^0(m')] + \exp[\hat{a}_k^1(m')] \} \} = \max_{\{m'\}} \{ \max_{\{m'\}} [\hat{a}_k^0(m'), \hat{a}_k^1(m')] \} \quad (A-1)
\]

Consider this trivial equality

\[
\exp[\hat{a}_k(m')] = \exp[\hat{a}_k^0(m')] + \exp[\hat{a}_k^1(m')] \quad (A-2)
\]

Substitute (A – 2) into the left side of (A – 1), and according to (2 – 11),

\[
\ln \{ \hat{\Omega} \{ \exp[\hat{a}_k^0(m')] + \exp[\hat{a}_k^1(m')] \} \} = \ln \{ \hat{\Omega} \exp[\hat{a}_k(m')] \} \\
= \max_{\{m'\}} \{ \hat{a}_k(m') \} \quad (A-3)
\]

where \( \hat{a}_k(m') = \ln \{ \exp[\hat{a}_k^0(m')] + \exp[\hat{a}_k^1(m')] \} \}

From (2 – 12),

\[
\hat{a}_k(m') = \max_{\{m'\}} [\hat{a}_k^0(m'), \hat{a}_k^1(m')] \quad (A-4)
\]

Substitute (A – 4) into (A – 3), the validity of (A – 1) has been verified.

Part II:

The extrinsic information generated from the decoder is the LLR associating with the information bit sequence \( b_k \) at the time instant \( k \), which implies the information

\[
W_{n',k} = \ln \frac{\Pr\{ b_k = 1 \}}{\Pr\{ b_k = 0 \}} \quad (A-5)
\]

where \( W_{n',k} \) represents the extrinsic information feedback from the decoder \( n' \).
if the assumption $b_k = 1$ holds subject to the transition state $S_k = m, S_{k-1} = m'$,

$$W_{n',k} = \ln \frac{\Pr\{ b_k = 1 \}}{\Pr\{ b_k = 0 \}} = \ln \frac{\Pr\{ b_k = 1 \}}{1 - \Pr\{ b_k = 1 \}} \quad (A - 6)$$

Rearrange (A – 6), we can probability value of $\Pr\{ b_k = 1 \}$,

$$\Pr\{ b_k = 1 \} = \frac{\exp(W_{n',k})}{1 + \exp(W_{n',k})} \quad (A - 7)$$

Transform (A – 7) into the logarithm domain, and according to (2 – 12), we can deduce that

$$\ln[\Pr\{ b_k = 1 \}] = W_{n',k} + \ln[1 + \exp(W_{n',k})] = W_{n',k} + \max*(0, W_{n',k}) \quad (A – 8)$$

Similarly, we can also deduce that

$$\ln \{\Pr\{ b_k = 0 \}\} = - \max*(0, W_{n',k}) \quad (A – 9)$$
References