行政院國家科學委員會委託計畫成果報告

具有智慧型天線的寬頻的CDMA基地站收發機之研製-用於第三代行動通訊系統之編碼與調變(子計畫四)

Coding and Modulation for the Third Generation Mobile Communications

計劃編號：NSC87-2219-E-002-003
NSC89-2219-E-002-020
√NSC89-2219-E-002-037

執行期間：87年5月1日至90年7月31日

計畫主持人：林茂昭　教授

執行單位:國立台灣大學電信工程學研究所

日　　期：中華民國90年7月

摘要

在第三代無線行動通訊系統(3G),有人提出兩種適合於第三代無線行動通訊系統的錯誤更正碼,這兩種錯誤更正碼分別是碼率較低的傳統迴旋碼(256 個狀態)及二元渦輪碼,這兩種錯誤更正碼可提供很大的編碼增益。在這個計劃,我們提出一個具有很大的限制長度的籬柵碼(包含二元迴旋碼及編碼籬柵調變(TCM))。這個籬柵碼的編碼是由一個具有短的限制長度的迴旋碼(C)緊接著額外的處理單元所構成。 在這個計劃我們研究兩個具有上述特性的籬柵碼架構 。在第一個架構中,這個處理單元是由多個延遲處理器及多個訊號對應器所構成在第二個架構中,這個處理單元是由多個迴旋處理器及多個訊號對應器所構成,其中迴旋處理器是一個碼率為 1 的迴旋碼。上述兩種架構的籬柵碼都可藉由迴旋碼(C)的籬柵及一些已解出的回授訊息做次佳化的解碼。在這個計劃中我們不僅做性能分析並考慮具有額外處理單元的籬柵碼的硬體實現。 在 A 部分我們做具有額外處理單元的籬柵碼的性能分析,做一些上述架構的籬柵碼在可加性的白色高斯雜訊通道及獨立衰減通道及相關衰減通道的模擬。在 B 部分我們做具有額外處理單元的籬柵碼的電路設計。C 部分為結論。根據研究的結果,具有額外處理單元的籬柵碼可以達到低錯誤率及低複雜度的硬體電路實現,因此也可以應用在第三代無線行動通訊系統。

ABSTRACT

In the third generation (3G) wireless mobile systems, both powerful low-rate conventional convolutional codes (256 states) and binary turbo codes were suggested to obtain a large coding gain in the wireless channel. In this project, we suggest a class of trellis code (include both binary convolutional codes and trellis coded modulation (TCM)) with large constraint length. The suggested trellis code is encoded by using the encoder of a convolutional code C with a short constraint length followed by additional processing units. In this project, we study two such trellis coding schemes. For the first scheme, the processing units are composed of multiple pairs of delay processors and signal mappers. For the second scheme, the processing units are composed of a convolutional processor and a signal mapper, where a convolutional processor is a rate 1 convolutional code.The trellis code constructed from each of these schemes can be suboptimally decoded by using the trellis of the convolutional code C with some feedback information. In this project, we not only do the performance analysis but also consider the practical implementation of trellis codes with additional processing units. In Part A which is written in English, we focus on the performance analysis of trellis codes with additional processing units. Simulation of these trellis coding schemes in the additive white Gaussian noise (AWGN) channel, independent fading channel and correlated fading channel is provided. In Part B which is written in Chinese, we focus on the circuit design of trellis code with a convolutional processor by using FPGA. Finally, conclusion is given in Part C. Based on the result of this project, we conclude trellis coding with additional processing units can achieve high transmission reliability and low complexity of implementation and hence can be served as a candidate in the third generation wireless mobile system in addition to the conventional convolutional code and turbo code.

List of Content

I Introduction

It is already known that a trellis code encoded by using the encoder of a convolutional code C with a short constraint length followed by additional processing units is equivalent to a trellis code with a large constraint length. In [1] and [3], a trellis coding scheme for which the processing unit consists of a delay processor and a signal mapper was proposed and analyzed. In [14], a trellis coded modulation scheme (TCM) for which the processing unit consists of a convolutional processor and a signal mapper was proposed, where a convolutional processor is a rate one convolutional code. Although an efficient decoding algorithm has been devised and performance analysis on the additive white Gaussian noise (AWGN) channel has also been reported for this TCM [14], in this three-year project, we will devise a more powerful decoding algorithm and study their performance in various channels. In addition to the trellis code proposed in [1,3] and TCM proposed in [14], we will study the performance of convolutional codes with a convolutional processor and a signal mapper and trellis codes with multiple pairs of delay processors and signal mappers in the AWGN channel, independent Rayleigh fading channel and correlated Rayleigh fading channel by using the proposed decoding. In the first and second year of this project, we have applied these two trellis coding schemes in the AWGN channel and the independent Rayleigh fading channel. In the last year of this project, we consider the correlated Rayleigh fading channel. The remaining parts of this report (Part A) are organized as follows. In section II, we briefly review the trellis coding scheme proposed in [1,3]. In section III, we analyze the trellis coding scheme with multiple pairs of delay processors and signal mappers. In section IV, we analyze the trellis coding scheme with a convolutional processor and a signal mapper. Both the analyzed schemes can be suboptimally decoded by using the trellis of C. Simulation for various trellis codes has been implemented without using interleaving and iterative decoding. Simulation results in the AWGN channel, independent Rayleigh fading channel and correlated Rayleigh fading channel are given in section V. Comparison among trellis coding with additional processing units, conventional trellis coding and

turbo coding are also provided in section V. From this, trellis coding with additional processing units can achieve high transmission reliability and low complexity of implementation and hence can be served as a candidate in the third generation wireless mobile system in addition to the conventional convolutional code and turbo code. Finally, some remarks are given in section VI.

II Trellis Coding Using a Delay Processor and a Signal Mapper

In this section, we briefly review the design of trellis codes given in [1,3]. The encoding is given in Fig. 1. The encoder for a rate r/m convolutional code C converts an input message sequence $\bar{u} = \{\cdots, \hat{u}(0), \hat{u}(1), \cdots\}$ to a sequence $\bar{v} = \{\cdots, \hat{v}(0), \hat{v}(1), \cdots\}$, where $\hat{u}(t) = (u_1(t), \cdots, u_r(t))$ is an r-bit symbol and $\hat{v}(t) = (v_1(t), \cdots, v_m(t))$ is an m-bit symbol. The sequence $\bar{v}$ is then fed into the delay processor which produces a sequence $\bar{s} = \{\cdots, \hat{s}(0), \hat{s}(1), \cdots\}$, where $\hat{s}(t) = (s_1(t), \cdots, s_m(t))$ is an m-bit symbol. The relation between $v_j(t)$ and $s_j(t)$ can be described by

$$s_j(t) = v_j(t - \sum_{i=j}^{m} \lambda_i) = v_j(t - \tau_j) \quad \text{for } 1 \leq j \leq m, \tag{1}$$

where $\tau_j = \sum_{i=j}^{m} \lambda_i$, $\lambda_m = 0$ and $\lambda_i \geq 0$ for $1 \leq i \leq m-1$. The sequence $\bar{s}$ is then fed into the signal mapper to yield an output symbol sequence $\bar{z} = \{\cdots, z(0), z(1), \cdots\}$, where $z(t) = \omega(\hat{s}(t)) \in \Omega$ and Ω is a signal space consisting of 2^m signal points. The signal space Ω can be a signal constellation, such as $MPSK$ $(M = 2^m)$, or a collection of binary m-tuples, such as $\{0,1\}^m$.

In [1], constant delays are used in the delay processor and in [3] delays of various values are used. For simplification of presentation, we only consider constant delays here. Hence, we have $\lambda_j = \lambda$ for $1 \leq j \leq m-1$ and $\tau_j = (m-j)\lambda$ for $1 \leq j \leq m$. We can construct an m-level partition chain $\Omega_0/\Omega_1/\Omega_2/\cdots/\Omega_m$ such that every signal point z in $\Omega = \Omega_0$ corresponds to a unique binary m-tuple $\hat{s}=(s_1, s_2, \cdots, s_m)$, i.e., $z = \omega(\hat{s})$, where $z \in \Omega$ and s_i, $1 \leq i \leq m$, is the coset label of Ω_{i-1}/Ω_i [1,4]. Let $\Delta(z, z')$ denote a distance measure between signal points z, $z' \in \Omega$. If Ω is a signal constellation, then $\Delta(z, z')$ is the squared Euclidean distance

between z and z'. If $\Omega = \{0,1\}^m$, then $\Delta(z,z')$ is the Hamming distance between the binary representations of z and z'. We define Δ_j to be the least one of all the possible $\Delta(\omega(\hat{s}),\omega(\hat{s}'))$, where $\omega(\hat{s})$ and $\omega(\hat{s}')$ are in an arbitrary coset of Ω_{j-1}, $\omega(\hat{s})$ is in a coset of Ω_j labeled by $s_j = 1$ and $\omega(\hat{s}')$ is in a coset of Ω_j labeled by $s'_j = 0$. If Ω is the 8PSK signal constellation as given in [2], then $\{\Delta_1,\Delta_2,\Delta_3\} = \{0.586, 2, 4\}$. If Ω=$\{0,1\}^m$ and the mapping $\omega : \Omega \to \Omega$ is linear, then ω can be represented by an $m \times m$ matrix K. That means $z = \omega(\hat{s}) = \hat{s}K$. It can be checked that $\{\Delta_1,\Delta_2,\Delta_3,\Delta_4\} = \{1,2,2,4\}$, if $\Omega = \{0,1\}^4$ and

$$K = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{2}$$

Let $\bar{z}$ and $\bar{z}'$ be the output symbol sequences associated with $\bar{v} = \{\cdots,\hat{v}(0),\hat{v}(1),\cdots\}$ and $\bar{v}' = \{\cdots,\hat{v}'(0),\hat{v}'(1),\cdots\}$ respectively. Assume $\hat{v}(t) = \hat{v}'(t)$ for $t < 0$ and $\hat{v}(0) \neq \hat{v}'(0)$. The pairwise distance measure between sequences $\bar{z}$ and $\bar{z}'$ is lower bounded [3] by

$$\Delta_{LB}((\bar{v},\bar{v}'),\lambda,\Delta_j) = \sum_{j=1}^{m}\sum_{t=0}^{\lambda-1}(v_j(t) \oplus v'_j(t))\Delta_j. \tag{3}$$

Define Δ_{free} as the free distance of the trellis code (binary convolutional code) if $\Omega = \{1,0\}^m$, and as the squared free distance of the trellis code (TCM) if Ω is a signal constellation. Then, we have the following theorem [3].

Theorem 1 *For the trellis code shown in Fig. 1 with $\lambda_j = \lambda$, $1 \leq j \leq m-1$, its Δ_{free} is lower bounded by*

$$\Delta_{LB}(C,\lambda,\Delta_j) = \min_{\bar{v}\in C,\hat{v}(0)\neq\hat{0}}\sum_{j=1}^{m}\sum_{t=0}^{\lambda-1} v_j(t)\Delta_j, \tag{4}$$

where $\hat{0}$ is the zero m-tuple. □

Example 1 : Let $r = 2, m = 3$ and Ω be the 8PSK signal set [2] with $\{\Delta_1,\Delta_2,\Delta_3\}$=$\{0.586, 2, 4\}$. The resultant code is a TCM for which its coding rate is 2 bits per 8PSK signal point. Let ν

be the number of memory bits in the encoder of C. Consider two cases : (a) $\nu = 2$ and (b) $\nu = 4$. The generator matrices are respectively given by

$$G_E = \begin{pmatrix} 1 & 1 & 1 \\ 5 & 7 & 0 \end{pmatrix}, and \quad G_E = \begin{pmatrix} 4 & 7 & 3 \\ 5 & 3 & 4 \end{pmatrix}.$$

From (4), we can calculate the squared free distance, D^2_{free}, of this TCM. D^2_{free} is at least 6.34 if $\lambda \geq 11$ for case (a) and at least 8.93 if $\lambda \geq 23$ for case (b). □

Example 2 : Let $r = 2$, $m = 4$ and $\Omega = \{1, 0\}^4$ with the mapping described in (2) and $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\}$=$\{1, 2, 2, 4\}$. The resultant code is a rate 1/2 convolutional code. Consider two cases : (a) $\nu = 2$ and (b) $\nu = 4$. The generator matrices are respectively given by

$$G_E = \begin{pmatrix} 0 & 3 & 3 & 2 \\ 3 & 3 & 2 & 1 \end{pmatrix}, and \quad G_E = \begin{pmatrix} 0 & 2 & 7 & 7 \\ 7 & 5 & 7 & 2 \end{pmatrix}.$$

The free distance, d_{free}, is at least 12 if $\lambda \geq 5$ for case (a) and at least 16 if $\lambda \geq 9$ for case (b). □

A suboptimum decoding which only needs the trellis of C can be designed [1,3]. Let $y(t)$ be the received symbol which is the possibly error-corrupted form of $z(t)$. For the t-th time unit, we calculate the bit metric for $v_j(t)$ based on the received $y(t + \tau_j)$ and the previously recovered code bit, $v_i(t - (j - i)\lambda)$, $1 \leq i < j$. Then, the bit metrics for $v_1(t)$, $\cdots$, $v_m(t)$ are summed up to form the branch metric for $\hat{v}(t)$. With the branch metrics for all the possible $\hat{v}(t - i)$, $i \geq 0$, the Viterbi decoder of C can recover $\hat{u}(t - \lambda + 1)$ and $\hat{v}(t - \lambda + 1)$, where λ is also used as the truncation length of decoding. The decoding delay is $m\lambda - 1$ time units.

III Trellis Coding Using Multiple Pairs of Delay Processors and Signal Mappers

As indicated in [1, 3], we observe that a delay processor and a signal mapper following the encoder of a convolutional code C can result in a convolutional code C' of a large free distance.

be the number of memory bits in the encoder of C. Consider two cases : (a) $\nu = 2$ and (b) $\nu = 4$. The generator matrices are respectively given by

$$G_E = \begin{pmatrix} 1 & 1 & 1 \\ 5 & 7 & 0 \end{pmatrix}, and \quad G_E = \begin{pmatrix} 4 & 7 & 3 \\ 5 & 3 & 4 \end{pmatrix}.$$

From (4), we can calculate the squared free distance, D^2_{free}, of this TCM. D^2_{free} is at least 6.34 if $\lambda \geq 11$ for case (a) and at least 8.93 if $\lambda \geq 23$ for case (b). □

Example 2 : Let $r = 2$, $m = 4$ and $\Omega = \{1,0\}^4$ with the mapping described in (2) and $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\}$=$\{1,2,2,4\}$. The resultant code is a rate 1/2 convolutional code. Consider two cases : (a) $\nu = 2$ and (b) $\nu = 4$. The generator matrices are respectively given by

$$G_E = \begin{pmatrix} 0 & 3 & 3 & 2 \\ 3 & 3 & 2 & 1 \end{pmatrix}, and \quad G_E = \begin{pmatrix} 0 & 2 & 7 & 7 \\ 7 & 5 & 7 & 2 \end{pmatrix}.$$

The free distance, d_{free}, is at least 12 if $\lambda \geq 5$ for case (a) and at least 16 if $\lambda \geq 9$ for case (b). □

A suboptimum decoding which only needs the trellis of C can be designed [1,3]. Let $y(t)$ be the received symbol which is the possibly error-corrupted form of $z(t)$. For the t-th time unit, we calculate the bit metric for $v_j(t)$ based on the received $y(t + \tau_j)$ and the previously recovered code bit, $v_i(t - (j - i)\lambda)$, $1 \leq i < j$. Then, the bit metrics for $v_1(t), \cdots, v_m(t)$ are summed up to form the branch metric for $\hat{v}(t)$. With the branch metrics for all the possible $\hat{v}(t - i)$, $i \geq 0$, the Viterbi decoder of C can recover $\hat{u}(t - \lambda + 1)$ and $\hat{v}(t - \lambda + 1)$, where λ is also used as the truncation length of decoding. The decoding delay is $m\lambda - 1$ time units.

III Trellis Coding Using Multiple Pairs of Delay Processors and Signal Mappers

As indicated in [1, 3], we observe that a delay processor and a signal mapper following the encoder of a convolutional code C can result in a convolutional code C' of a large free distance.

It is natural to consider once again applying a delay processor and a signal mapper to the output of the convolutional code C' to achieve a possibly larger free distance.

Consider a trellis code with its encoder given in Fig. 2. The input message sequence $\bar{u}$ is first converted to a sequence $\bar{v}^{(1)}$ through an encoder of a convolutional code C. The sequence $\bar{v}^{(1)}$ is then repeatedly processed by L pairs of delay processors (denoted as $Q^{(1)}, \cdots, Q^{(L)}$ respectively in Fig. 2) and signal mappers with mapping functions of $\omega^{(1)}, \cdots, \omega^{(L)}$ respectively to produce the output symbol sequence $\bar{z}$. Let $\bar{v}^{(\ell)} = (\cdots, \hat{v}^{(\ell)}(0), \hat{v}^{(\ell)}(1), \cdots)$ be the input sequence for $Q^{(\ell)}$ and $\bar{s}^{(\ell)} = (\cdots, \hat{s}^{(\ell)}(0), \hat{s}^{(\ell)}(1), \cdots)$ be the output sequence for $Q^{(\ell)}$, where $1 \leq \ell \leq L$. The ℓ-th signal mapper maps the binary m-tuple $\hat{s}^{(\ell)}(t) = (s_1^{(\ell)}(t), \cdots, s_m^{(\ell)}(t))$ to

$$\hat{v}^{(\ell+1)}(t) = \omega^{(\ell)}(\hat{s}^{(\ell)}(t)) \quad \text{for } 1 \leq \ell \leq L. \tag{5}$$

For $1 \leq \ell \leq L$, $\hat{v}^{(\ell)}(t) = (v_1^{(\ell)}(t), \cdots, v_m^{(\ell)}(t))$ is a binary m-tuple, while $\hat{v}^{(L+1)}(t) = z(t) = \omega^{(L)}(\hat{s}^{(L)}(t)) \in \Omega$ is the output symbol which may be a binary m-tuple or a signal point of a signal constellation. The relation between $s_j^{(\ell)}(t)$ and $v_j^{(\ell)}(t)$ is given by

$$s_j^{(\ell)}(t) = v_j^{(\ell)}(t - (m-j)\lambda^{(\ell)}) = v_j^{(\ell)}(t - \tau_j^{(\ell)}) \quad \text{for } 1 \leq j \leq m, 1 \leq \ell \leq L, \tag{6}$$

where $\tau_j^{(\ell)} = (m-j)\lambda^{(\ell)}$. Suppose that the mapping function $\omega^{(\ell)}$ is linear and invertible for $1 \leq \ell \leq L-1$. Then $\omega^{(\ell)}$ can be represented by an $m \times m$ nonsingular matrix $K^{(\ell)}$. That is

$$\hat{v}^{(\ell+1)}(t) = \hat{s}^{(\ell)}(t)K^{(\ell)} \quad \text{for } 1 \leq \ell \leq L-1. \tag{7}$$

The mapping function $\omega^{(L)}$ can also be represented by a nonsingular matrix $K^{(L)}$ if $\Omega = \{0,1\}^m$ and $\omega^{(L)}$ is linear and invertible.

Let $C^{(L)}$ be the collection of $\bar{v}^{(L)}$ resultant from all the possible $\bar{u}$. By Theorem 1, Δ_{free} of the trellis code shown in Fig. 2 is lower bounded by

$$\Delta_{LB}(C^{(L)}, \lambda^{(L)}, \Delta_j) = \min_{\bar{v}^{(L)} \in C^{(L)}, \hat{v}^{(L)}(0) \neq \hat{0}} \sum_{j=1}^{m} \sum_{t=0}^{\lambda^{(L)}-1} v_j^{(L)}(t)\Delta_j. \tag{8}$$

It is difficult to calculate the lower bound of the Δ_{free} based on (8). Hence, we will resort to another approach to calculate the lower bound. In the following, we will assume $\lambda^{(\ell)} \geq m\lambda^{(\ell-1)}$

for $2 \le \ell \le L$. Let $\delta_j^{(L)} = \Delta_j$, $1 \le j \le m$. Define

$$d_w^{(\ell)}(\hat{v}^{(\ell)}) = \sum_{j=1}^{m} v_j^{(\ell)} \delta_j^{(\ell)} \quad \text{for } 1 \le \ell \le L, \tag{9}$$

and

$$\delta_j^{(\ell)} = \min d_w^{(\ell+1)}(\omega^{(\ell)}(\hat{s}^{(\ell)})) \quad \text{for } 1 \le \ell \le L-1, \tag{10}$$

where the minimization is over all the possible $\hat{s}^{(\ell)}$ with $\hat{s}_j^{(\ell)} \neq \hat{0}$ and $s_i^{(\ell)} = 0$ for $i < j$. With the initially given $\{\delta_1^{(L)}, \cdots, \delta_m^{(L)}\}$, we can recursively calculate $\{\delta_1^{(\ell)}, \cdots, \delta_m^{(\ell)}\}$ for $\ell = L-1, \cdots, 1$. From Appendix A, we have the following theorem.

Theorem 2 *If $\lambda^{(\ell+1)} \ge m\lambda^{(\ell)}$ for $1 \le \ell < L$, then Δ_{free} of the trellis code shown in Fig. 2 is lower bounded by*

$$\Delta_{LB}(C, \lambda^{(1)}, \delta_j^{(1)}) = \min_{\hat{v}^{(1)} \in C, \hat{v}^{(1)}(0) \neq \hat{0}} \sum_{t=0}^{\lambda^{(1)}-1} d_w^{(1)}(\hat{v}^{(1)}(t)) = \min_{\hat{v}^{(1)} \in C, \hat{v}^{(1)}(0) \neq \hat{0}} \sum_{t=0}^{\lambda^{(1)}-1} \sum_{j=1}^{m} v_j^{(1)}(t) \delta_j^{(1)}. \tag{11}$$

□

For arbitrarily chosen nonsingular matrices $K^{(\ell)}$, $1 \le \ell \le L-1$, it is not necessarily true that $\delta_j^{(1)} \ge \Delta_j$ for all j. In this project, we use the following design procedure to choose $K^{(\ell)}$.

Step 1 Let $I = \{1, 2, \cdots, m\}$ be the index set. Suppose that $\delta_{i_1}^{(L)} = \delta_{i_1+1}^{(L)} = \cdots = \delta_{i_2-1}^{(L)} < \delta_{i_2}^{(L)} = \delta_{i_2+1}^{(L)} = \cdots = \delta_{i_3-1}^{(L)} < \delta_{i_3}^{(L)} = \delta_{i_3+1}^{(L)} = \cdots = \delta_{i_p-1}^{(L)} < \delta_{i_p}^{(L)} = \delta_{i_p+1}^{(L)} = \cdots = \delta_{i_q-1}^{(L)} < \delta_{i_q}^{(L)} = \delta_{i_q+1}^{(L)} = \cdots = \delta_m^{(L)}$, where $J \equiv \{i_1, \cdots, i_q\} \subseteq I$ and $1 = i_1 < i_2 < \cdots < i_q \le m$. For a nonzero m-tuple, $\hat{v} = (v_1, \cdots, v_m)$, define $R(\hat{v})$ to be the largest index i such that $v_i \neq 0$. Let $C_{j,m}^{(L-1)}$ be the code generated by the last $(m-j+1)$ rows of a nonsingular matrix $K^{(L-1)}$. We choose $K^{(L-1)}$ such that (a) for $i_p \le j < i_{p+1}$, $p = 1, 2, \cdots, q$,

$$\min_{\hat{v} \in C_{j,m}^{(L-1)}, \hat{v} \neq \hat{0}} R(\hat{v}) \ge i_p,$$

where $i_p \in J$ and $i_{q+1} = m+1$; (b) there is at least one j such that the minimum Hamming distance of $C_{j,m}^{(L-1)}$ is at least 2. Note that (9) and (10) imply

$$\delta_j^{(L-1)} = \min_{\hat{v} \in C_{j,m}^{(L-1)}, \hat{v} \neq \hat{0}} \sum_{i=1}^{m} v_i \delta_i^{(L)}, \tag{12}$$

From condition (a), we have $\delta_j^{(L-1)} \geq \delta_j^{(L)}$ for $1 \leq j \leq m$ and $\delta_1^{(L-1)} \leq \delta_2^{(L-1)} \cdots \leq \delta_m^{(L-1)}$. With the additional condition (b), we further have $\delta_j^{(L-1)} > \delta_j^{(L)}$ for at least one j.

Step 2 Choose $K^{(\ell)}$, $\ell = L-2, \cdots, 1$ in a way similar to Step 1. Then we have $\delta_j^{(\ell)} \geq \delta_j^{(\ell+1)}$ for $\ell = L-2, \cdots, 1$ and $1 \leq j \leq m$. Hence $\delta_j^{(1)} \geq \Delta_j$ for $1 \leq j \leq m$ and $\delta_j^{(1)} > \Delta_j$ for at least one j.

Step 3 For a given C, we compute the lower bound of Δ_{free} according to (11). This can be done in a way similar to the computation of the free distance of C except that the weighting factors, $\delta_1^{(1)}, \cdots, \delta_{m-1}^{(1)}$ and $\delta_m^{(1)}$ must be considered.

Using the above procedure, we can construct trellis codes with large free distances.

Example 3 : Let C be a rate 2/3 binary convolutional code, $L = 2$ and Ω be the 8PSK signal set [2] with $\{\Delta_1, \Delta_2, \Delta_3\}$=$\{0.586, 2, 4\}$. Choose

$$K^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

The resultant code is a TCM for which its coding rate is 2 bits per 8PSK signal point. We have $\{\delta_1^{(2)}, \delta_2^{(2)}, \delta_3^{(2)}\} = \{0.586, 2, 4\}$. From (12), we have $\{\delta_1^{(1)}, \delta_2^{(1)}, \delta_3^{(1)}\} = \{0.586, 2.586, 6.586\}$. We consider two cases : (a) $\nu = 2$ and (b) $\nu = 4$. The associated generator matrices are the same as those used in Example 1. From (11), we can calculate the squared free distance, D_{free}^2, of this TCM. For case (a), D_{free}^2 is at least 7.52 if $\lambda^{(1)} \geq 12$ and $\lambda^{(2)} \geq 36$. For case (b), D_{free}^2 is at least 10.69 if $\lambda^{(1)} \geq 25$ and $\lambda^{(2)} \geq 75$. □

Example 4 : Let C be a rate 2/4 binary convolutional code, $L = 2$ and $\Omega = \{0, 1\}^4$. Let $K^{(2)}$ be the same as the matrix K given in (2). Then, $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\}$=$\{1, 2, 2, 4\}$. Choose

$$K^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

The resultant code is a rate 1/2 binary convolutional code. We have $\{\delta_1^{(2)}, \delta_2^{(2)}, \delta_3^{(2)}, \delta_4^{(2)}\} = \{1, 2, 2, 4\}$. From (12), we have $\{\delta_1^{(1)}, \delta_2^{(1)}, \delta_3^{(1)}, \delta_4^{(1)}\} = \{1, 3, 3, 9\}$. Consider two cases : (a) $\nu = 2$ and (b) $\nu = 4$. The associated generator matrices are the same as those used in Example 2. The free distance, d_{free}, of the constructed binary trellis code is at least 17 if $\lambda^{(1)} \geq 5$ and $\lambda^{(2)} \geq 20$ for case (a) and is at least 24 if $\lambda^{(1)} \geq 10$ and $\lambda^{(2)} \geq 40$ for case (b). □

A suboptimum decoding for the trellis code can be implemented as follows. In the following, we illustrate the decoding method in the AWGN channel. The decoding method in the fading channel is similar to that in the AWGN channel. Let $||y - z||^2$ represent the squared Euclidean distance between symbol y and symbol z. If y or z is a binary m-tuple, then a bit "0" must be replaced by "-1" and a bit "1" remains to be "1". Let $y(t) = \hat{v}^{*(L+1)}(t)$ be the received symbol that is the possibly error-corrupted form of $z(t) = \hat{v}^{(L+1)}(t)$. For $\ell = L, \cdots, 1$, $1 \leq j \leq m$, define the log-likelihood-ratio [1] for bit $v_j^{(\ell)}(t)$ by

$$\begin{aligned} \mu_j^{(\ell)}(t) &= \ln \frac{p(\hat{v}^{*(\ell+1)}(t + \tau_j^{(\ell)}) | v_j^{(\ell)}(t) = 1)}{p(\hat{v}^{*(\ell+1)}(t + \tau_j^{(\ell)}) | v_j^{(\ell)}(t) = 0)} \\ &\approx \min_{x \in A_0} \frac{||\hat{v}^{*(\ell+1)}(t + \tau_j^{(\ell)}) - \omega^{(\ell)}(x)||^2}{N_0} - \min_{x \in A_1} \frac{||\hat{v}^{*(\ell+1)}(t + \tau_j^{(\ell)}) - \omega^{(\ell)}(x)||^2}{N_0}, \end{aligned} \quad (13)$$

where $\hat{v}^{*(\ell+1)}(p) = (\mu_1^{(\ell+1)}(p), \cdots, \mu_m^{(\ell+1)}(p))$ for $\ell = L-1, \cdots, 1$ and $A_i = \{\omega^{(\ell)}(s_1^{(\ell)}(t + \tau_j^{(\ell)}), \cdots, s_{j-1}^{(\ell)}(t + \tau_j^{(\ell)}), i, x_{j+1}, \cdots, x_m) \ | x_q \in \{0, 1\}$ for $j < q \leq m\}$ for $i = 0, 1$. Let $\eta^{(1)} = 0$, $\eta^{(\ell)} = \sum_{i=1}^{\ell-1} \tau_1^{(i)}$ and $\lambda^{(\ell)} \geq m\lambda^{(\ell-1)}$ for $\ell \geq 2$. Note that $\lambda^{(2)} \geq \lambda^{(1)} + \eta^{(2)}$. Since $\lambda^{(\ell)} \geq \lambda^{(1)} + \eta^{(\ell)}$ implies that $\lambda^{(\ell+1)} \geq \lambda^{(\ell)} + \tau_1^{(\ell)} \geq \lambda^{(1)} + \eta^{(\ell)} + \tau_1^{(\ell)} = \lambda^{(1)} + \eta^{(\ell+1)}$, then $\lambda^{(\ell)} \geq \lambda^{(1)} + \eta^{(\ell)}$ for $\ell \geq 2$. Assume that $\hat{v}^{(\ell)}(t - l)$ has been correctly recovered for $l \geq \lambda^{(1)}$ and $1 \leq \ell \leq L$. The suboptimum decoding can be implemented as follows.

Step 1 For $\ell = L, \cdots, 1$, we use (13) to calculate $\mu_j^{(\ell)}(t + \eta^{(\ell)})$ for $v_j^{(\ell)}(t + \eta^{(\ell)})$. Note that for $1 \leq p < j$, $s_p^{(\ell)}(t + \tau_j^{(\ell)} + \eta^{(\ell)}) = v_p^{(\ell)}(t - (j-p)\lambda^{(\ell)} + \eta^{(\ell)})$ has already been recovered since $(j-p)\lambda^{(\ell)} \geq \lambda^{(1)} + \eta^{(\ell)}$.

Step 2 Using $\mu_j^{(1)}(t) = \mu_j^{(1)}(t + \eta^{(1)})$, $j = 1, \cdots, m$, we calculate the branch metrics for all the possible $\hat{v}(t)$, which are fed to the Viterbi decoder of C to recover $\hat{u}(t - \lambda^{(1)} + 1)$ and

$\hat{v}^{(1)}(t-\lambda^{(1)}+1)$, where $\lambda^{(1)}$ is used as the truncation length of the Viterbi decoder of C.

Step 3 The recovered $\hat{v}^{(1)}(t-\lambda^{(1)}+1)$ is used to recover $\hat{s}^{(1)}(t-\lambda^{(1)}+1)$ which is then used to recover $\hat{v}^{(\ell)}(t-\lambda^{(1)}+1)$ and $\hat{s}^{(\ell)}(t-\lambda^{(1)}+1)$ for $\ell=2,\cdots,L$. Then we increase t by 1 and go to step 1.

The error performance of the suboptimum decoding can be further improved by using SOVA [5], since the assumption of correct recovery of $\hat{v}^{(\ell)}(t-l)$ is not always true. Let the log-likelihood-ratio obtained by SOVA for $s_j^{(\ell)}(t)$ be denoted by $\Lambda(s_j^{(\ell)}(t))$. Then, the parameter $\mu_j^{(\ell)}(t)$ given in (13) is modified to be

$$\begin{aligned}\mu_j^{(\ell)}(t) \approx & \min_{x\in B_0}\{\frac{||\hat{v}^{*(\ell+1)}(t+\tau_j^{(\ell)})-\omega^{(\ell)}(x)||^2}{N_0}-\frac{1}{2}\sum_{i=1}^{j-1}[2s_i^{(\ell)}(t+\tau_j^{(\ell)})-1]\Lambda(s_i^{(\ell)}(t+\tau_j^{(\ell)}))\} \quad (14)\\ - & \min_{x\in B_1}\{\frac{||\hat{v}^{*(\ell+1)}(t+\tau_j^{(\ell)})-\omega^{(\ell)}(x)||^2}{N_0}-\frac{1}{2}\sum_{i=1}^{j-1}[2s_i^{(\ell)}(t+\tau_j^{(\ell)})-1]\Lambda(s_i^{(\ell)}(t+\tau_j^{(\ell)}))\},\end{aligned}$$

where $B_i=\{\omega^{(\ell)}(s_1^{(\ell)}(t+\tau_j^{(\ell)}),\cdots,s_{j-1}^{(\ell)}(t+\tau_j^{(\ell)}),i,x_{j+1},\cdots,x_m) \mid s_q^{(\ell)}(t+\tau_j^{(\ell)})\in\{0,1\}$ for $1\le q<j, x_q\in\{0,1\}$ for $j<q\le m\}$ for $i=0,1$. The total decoding delay is $\lambda^{(1)}+\eta^{(L+1)}-1$ time units. If $\lambda^{(\ell)}=m\lambda^{(\ell-1)}$ for $\ell\ge 2$, then $\lambda^{(1)}+\eta^{(L+1)}-1=m^L\lambda^{(1)}-1$.

IV Trellis Coding Using a Convolutional Processor and a Signal Mapper

The trellis coding shown in Fig. 2 can be described in a different way. Let D represent the operator of one unit time delay. Define an $m\times m$ diagonal matrix $Q_M^{(\ell)}$ for $1\le\ell\le L$, for which the entry at the i-th row and the i-th column is $D^{\tau_i^{(\ell)}}$. The function of the delay processor $Q^{(\ell)}$ can be represented by $\hat{s}^{(\ell)}(t)=\hat{v}^{(\ell)}(t)Q_M^{(\ell)}$ for $1\le\ell\le L$. Then, we have

$$\begin{aligned}\hat{s}^{(L)}(t) &= \hat{v}^{(L)}(t)Q_M^{(L)}=\hat{s}^{(L-1)}(t)K^{(L-1)}Q_M^{(L)}=\cdots\\ &= \hat{v}^{(1)}(t)Q_M^{(1)}K^{(1)}Q_M^{(2)}\cdots K^{(L-1)}Q_M^{(L)}=\hat{v}^{(1)}(t)P, \quad (15)\end{aligned}$$

where $P = Q_M^{(1)} K^{(1)} Q_M^{(2)} \cdots K^{(L-1)} Q_M^{(L)}$. We may regard P as the transfer function matrix for a rate 1 convolutional code with input sequence $\bar{v}^{(1)}$ and output sequence $\bar{s}^{(L)}$. This suggests a new class of trellis coding with encoding configuration shown in Fig. 3, in which an input message sequence $\bar{u}$ is fed into the encoder of a convolutional code C followed by a convolutional processor and a signal mapper to produce the output symbol sequence $\bar{z}$. The convolutional processor is the encoder of a rate 1 convolutional code with transfer function matrix P.

In the following, we will consider a special design of the convolutional processor P such that the $(j+1)$-th level input bit will affect only the $(j+1)$-th and the j-th level output bits. Then $P_{i,j}$, the entry at the i-th row and the j-th column of the $m \times m$ matrix P, will be zero except for the diagonal elements and $P_{j+1,j}$, where $j = 1, 2, \cdots, m-1$. For $j = 1, 2, \cdots, m$, set

$$P_{j,j} = D^{\beta_j \lambda}, \tag{16}$$

and for $j = 1, 2, \cdots, m-1$, set

$$P_{j+1,j} = \begin{cases} 0 & \text{if } \ell_{j+1} = 0, \\ (D^{(\beta_j - 1)\lambda} + \cdots + D^{(\beta_j - \ell_{j+1})\lambda}) & \text{if } \ell_{j+1} > 0. \end{cases} \tag{17}$$

Then, we have

$$s_m(t) = v_m(t - \beta_m \lambda), \tag{18}$$

and for $j = 1, 2, \cdots, m-1$,

$$s_j(t) = \begin{cases} v_j(t - \beta_j \lambda) & \text{if } \ell_{j+1} = 0, \\ v_j(t - \beta_j \lambda) \oplus v_{j+1}(t - (\beta_j - 1)\lambda) \oplus \cdots \oplus v_{j+1}(t - (\beta_j - \ell_{j+1})\lambda) & \text{if } \ell_{j+1} > 0. \end{cases} \tag{19}$$

To insure that $v_{j+1}(t)$, $j < m$, can affect $\ell_{j+1} + 1$ different output symbols, we must have $\beta_j - \beta_{j+1} \geq \ell_{j+1} + 1$. If it is desired that the pairwise distance measure between sequences $\bar{z}$ and $\bar{z}'$ is lower bounded by

$$\Delta_{LB}((\bar{v}, \bar{v}'), \lambda, (\Delta_j + \ell_j \Delta_{j-1})) = \sum_{j=1}^{m} \sum_{t=0}^{\lambda-1} (v_j(t) \oplus v_j'(t))(\Delta_j + \ell_j \Delta_{j-1}), \tag{20}$$

we require that β_m=0 and for $j = m-1, \cdots, 1$,

$$\beta_j - \beta_{j+1} = \ell_{j+1} + \zeta_{j+1} + 1, \tag{21}$$

where $\zeta_{j+1} \geq 0$ is an integer for which the constraint will be given in Theorem 3. The relation between $\bar{s}$ and $\bar{v} = (\hat{0}, \cdots, \hat{0}, \hat{v}(t), \hat{0}, \cdots, \hat{0})$ is illustrated in Fig. 4.

Example 5 : Let $\ell_1 = 0$, $\ell_2 = 1$, $\ell_3 = 0$, $\zeta_3 = 0, \zeta_2 = 1$ and $\zeta_1 = 0$. It follows from (21) that $\beta_3 = 0, \beta_2 = 1, \beta_1 = 4$. Hence by (16) and (17), we have

$$P = \begin{pmatrix} D^{4\lambda} & 0 & 0 \\ D^{3\lambda} & D^{\lambda} & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

We have $s_1(t) = v_1(t - 4\lambda) \oplus v_2(t - 3\lambda)$, $s_2(t) = v_2(t - \lambda)$, and $s_3(t) = v_3(t)$ by (18) and (19). The relation between $\bar{s}$ and $\bar{v}$ is shown in Fig. 5. Suppose that $\hat{v}(t) = \hat{v}'(t)$ for $t < 0$ and $\hat{v}(0) \neq \hat{v}'(0)$. We see that $\Delta(z(0), z'(0)) \geq (v_3(0) \oplus v_3'(0))\Delta_3$, $\Delta(z(\lambda), z'(\lambda)) \geq (v_2(0) \oplus v_2'(0))\Delta_2$, $\Delta(z(3\lambda), z'(3\lambda)) \geq (v_2(0) \oplus v_2'(0))\Delta_1$ and $\Delta(z(2\lambda), z'(2\lambda)) + \Delta(z(4\lambda), z'(4\lambda)) \geq (v_1(0) \oplus v_1'(0))\Delta_1$ if $\Delta_2 \geq \Delta_1$. Hence (20) is satisfied. □

From Appendix B, we have the following theorem.

Theorem 3 *Let $\ell_j \leq \lfloor \Delta_j / \Delta_{j-1} \rfloor$. If $\zeta_p \geq \zeta_{p-1} + \ell_p$ for $\ell_p > 0$ (and $\zeta_p = 0$ for $\ell_p = 0$), then Δ_{free} of the proposed trellis code is lower bounded by*

$$\Delta_{LB}(C, \lambda, (\Delta_j + \ell_j \Delta_{j-1})) = \min_{\bar{v} \in C, \hat{v}(0) \neq \hat{0}} \sum_{j=1}^{m} \sum_{t=0}^{\lambda-1} v_j(t)(\Delta_j + \ell_j \Delta_{j-1}), \tag{22}$$

where $\Delta_0 = 0$ and $\ell_1 = 0$. □

For this scheme, we use the following design procedure to construct trellis codes.

Step 1 Select $\ell_1, \cdots, \ell_m$ and $\zeta_1, \cdots, \zeta_m$ which are subject to the constraint given in Theorem 3. Calculate $\beta_1, \cdots, \beta_m$ by (21). Then we have P with entries described by (16) and (17).

Step 2 For a given C, we compute the lower bound of Δ_{free} according to (22). This can be done in a way similar to the computation of free distance of C except that the weighting factors, $(\Delta_1 + \ell_1 \Delta_0) = \Delta_1, \cdots, (\Delta_m + \ell_m \Delta_{m-1})$ must be considered.

For this scheme, increasing ℓ_j may result in increased Δ_{free} as indicated in (22). However, increasing ℓ_j may not necessarily yield improved error performance. This may result from the increased error coefficient. Moreover, decoding delay is also increased. Hence, a large ℓ_j is not necessarily desired.

Example 6 : Let $r = 2$, $m = 3$ and Ω be the 8PSK signal set [2] with $\{\Delta_1, \Delta_2, \Delta_3\}$= $\{0.586, 2, 4\}$. Let P be the matrix used in Example 5. Let d_i denote $\sum_{t=0}^{\lambda-1} v_i(t)$ for $i = 1, 2, 3$. The squared free distance of the constructed TCM is $D^2_{free} \geq \min_{\underline{v} \in C, \underline{v}(0) \neq \underline{0}} \{d_1 \cdot \Delta_1 + d_2 \cdot (\Delta_2 + \Delta_1) + d_3 \cdot \Delta_3\}$. Consider two cases : (a) $\nu = 2$ and (b) $\nu = 4$. The associated generator matrices are the same as those used in Example 1. We have $D^2_{free} \geq 7.17$ if $\lambda \geq 11$ for case (a) and $D^2_{free} \geq 9.52$ if $\lambda \geq 23$ for case (b). □

We can modify Example 6 by using ℓ_3=1 instead of ℓ_3=0. In this way, a larger lower bound on free distance can be achieved. However, the error performance remains similar.

Example 7 : Let $r = 2$, $m = 4$ and $\Omega = \{1, 0\}^4$ with the mapping described in (2) and $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\}$= $\{1, 2, 2, 4\}$. Let $\ell_4 = 0$, $\ell_3 = 1$, $\ell_2 = 1$, $\ell_1 = 0$, $\zeta_4 = 0$, $\zeta_3 = 2$, $\zeta_2 = 1$, $\zeta_1 = 0$. We have

$$P = \begin{pmatrix} D^{8\lambda} & 0 & 0 & 0 \\ D^{7\lambda} & D^{5\lambda} & 0 & 0 \\ 0 & D^{4\lambda} & D^{\lambda} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Then, we have a rate 1/2 binary trellis code with free distance $d_{free} \geq \min_{\underline{v} \in C, \underline{v}(0) \neq \underline{0}} \{d_1 \cdot \Delta_1 + d_2 \cdot (\Delta_2 + \Delta_1) + d_3 \cdot (\Delta_3 + \Delta_2) + d_4 \cdot \Delta_4\}$. Consider two cases (a) $\nu = 2$ and (b) $\nu = 4$. The associated generator matrices are the same as those used in Example 2. Then, we have $d_{free} \geq 14$ if $\lambda \geq 5$ for case (a) and $d_{free} \geq 20$ if $\lambda \geq 7$ for case (b). □

With the special design of P, the analyzed trellis code can be suboptimally decoded by using the trellis of C. As an illustration, we describe the decoding procedure for Example 6 as follows. In the following, we illustrate the decoding method in the AWGN channel. The decoding method in the fading channel is similar to that in the AWGN channel. We assume

that $\hat{v}(t-l)$ has been correctly recovered for $l \geq \lambda$.

Step 1 For $j \in \{0,1\}$, we calculate the bit metric for $v_1(t) = j$. From Fig. 5, we see that $s_1(t+4\lambda) = v_1(t) \oplus v_2(t+\lambda)$ contains the information of $v_1(t)$. The bit $v_2(t+\lambda)$ is not yet recovered and also appears in $\hat{s}(t+2\lambda)$, where $s_1(t+2\lambda) = v_1(t-2\lambda) \oplus v_2(t-\lambda)$ is already recovered at earlier time units. Hence, $v_1(t)$ can be estimated from the received symbols $y(t+4\lambda)$ and $y(t+2\lambda)$. The bit metric for $v_1(t) = j$ is calculated to be

$$\min_{\hat{s}(t+4\lambda),\hat{s}(t+2\lambda)} \{||y(t+4\lambda) - \omega(\hat{s}(t+4\lambda))||^2 + ||y(t+2\lambda) - \omega(\hat{s}(t+2\lambda))||^2\}$$

where $\hat{s}(t+4\lambda) = (j \oplus x_1, x_3, x_4)$, $\hat{s}(t+2\lambda) = (v_1(t-2\lambda) \oplus v_2(t-\lambda), x_1, x_2)$, and $x_1 = v_2(t+\lambda) \in \{0,1\}$, $x_2 = v_3(t+2\lambda) \in \{0,1\}$, $x_3 = v_2(t+3\lambda) \in \{0,1\}$, $x_4 = v_3(t+4\lambda) \in \{0,1\}$ (since $v_2(t+\lambda)$, $v_3(t+2\lambda)$, $v_2(t+3\lambda)$ and $v_3(t+4\lambda)$ are not yet recovered).

Step 2 For $j \in \{0,1\}$, we calculate the bit metric for $v_2(t) = j$. We see that $s_2(t+\lambda) = v_2(t)$ and $s_1(t+3\lambda) = v_1(t-\lambda) \oplus v_2(t)$ contain the information of $v_2(t)$, where $v_1(t-\lambda)$ is already recovered. Hence, $v_2(t)$ can be estimated by received symbols $y(t+\lambda)$ and $y(t+3\lambda)$. The bit metric for $v_2(t) = j$ is calculated to be

$$\min_{\hat{s}(t+\lambda)} \{||y(t+\lambda) - \omega(\hat{s}(t+\lambda))||^2\} + \min_{\hat{s}(t+3\lambda)} ||y(t+3\lambda) - \omega(\hat{s}(t+3\lambda))||^2\},$$

where $\hat{s}(t+\lambda) = (v_1(t-3\lambda) \oplus v_2(t-2\lambda), j, x_1)$, $\hat{s}(t+3\lambda) = (v_1(t-\lambda) \oplus j, x_2, x_3)$, and $x_1 = v_3(t+\lambda) \in \{0,1\}$, $x_2 = v_2(t+2\lambda) \in \{0,1\}$, $x_3 = v_3(t+3\lambda) \in \{0,1\}$.

Step 3 For $j \in \{0,1\}$, the bit metric for $v_3(t) = j$ is calculated to be $\min_{\hat{s}(t)}\{||y(t) - \omega(\hat{s}(t))||^2\}$, where $\hat{s}(t) = (v_1(t-4\lambda) \oplus v_2(t-3\lambda), v_2(t-\lambda), j)$.

Step 4 Calculate the metric for $\hat{v}(t)$ as the sum of bit metrics for $v_1(t)$, $v_2(t)$ and $v_3(t)$.

Step 5 With the branch metric for $\hat{v}(t-i)$, $i \geq 0$, we use the Viterbi decoder with truncation length λ for the convolutional code C to recover $\hat{u}(t-\lambda+1)$ and $\hat{v}(t-\lambda+1)$. Then we increase t by 1 and proceed to step 1. The decoding delay is $5\lambda - 1$ time units.

In the general case, the decoding is similar to that of Example 6. Suppose that for a nonnegative integer i, $s_p(t+i\lambda)$ contains the information of $v_j(t)$. That means $s_p(t+i\lambda)$ is the sum of $v_j(t)$, $v_{p'}(t+i'\lambda)$ and some other bits. Then, the received symbol $y(t+i\lambda)$ needs to be used in the estimation of $v_j(t)$. If $i' > 0$, and $s_{p''}(t+i''\lambda)$ contains the information of $v_{p'}(t+i'\lambda)$, then the received symbol $y(t+i''\lambda)$ also needs to be used in the estimation of $v_j(t)$. If $s_{p''}(t+i''\lambda)$ is the sum of $v_{p'}(t+i'\lambda)$, $v_{p'''}(t+i'''\lambda)$ and some other bits, where $i''' > 0$, then more received symbols need to be used in the estimation of $v_j(t)$. The bit metrics used in the above mentioned decoding can be modified by introducing log-likelihood ratios obtained by SOVA in a way similar to that described in section III.

V Performance Evaluation

Using trellis coding schemes with additional processing units, we can construct trellis codes with large free distances. However, the suboptimum decoding for each scheme may yield a large error coefficient. Therefore, simulation is needed to verify the error performance. In [1], interleaving is used. However, we do not use interleaving here, since large interleaving size requires large decoding delay. In the following, we provide simulation results of trellis codes with additional processing units in the AWGN channel, independent Rayleigh fading channel and correlated Rayleigh fading channel. The data related to the simulation for Examples 1, 2, 3, 4, 6 and 7 are listed in Table 1.

V.1 Performance over the AWGN channel

Simulation results over the AWGN channel for Examples 3, 4, 6 and 7 using the suboptimum decoding (with and without SOVA) are given in Fig. 6, 7, 8 and 9 respectively. Simulation results for trellis codes constructed from Hellstern's scheme (Examples 1 and 2) and the 64-state conventional binary convolutional code and the 16-state Ungerboeck TCM are also given in these figures.

The numbers of additions and comparisons per symbol (add,comp) can be used as one measure of decoding complexity. In Table 1, (add, comp) are computed based on the suboptimal decoding using hard feedback, i.e., without SOVA. If SOVA is used, the sum of add and comp will be about twice of that of not using SOVA. From Fig. 8 and Table 1, we see that Example 6b has similar complexity and better error performance as compared to Example 1b. In addition to (add, comp), the number of trellis states can serve as another measure of decoding complexity. For a high-speed Viterbi decoder using many parallel processors, the number of trellis states will be a dominant factor of complexity[6]. In this case, either Example 3b or 6b has similar complexity and better error performance as compared to Example 1b. Similar comparison can be made between other examples such as Examples 3a and 1a, Examples 2a and 4a, $\cdots$, etc. We can conclude that either of the analyzed schemes yields error performance better than Hellstern's scheme based on the same C (or similar decoding complexity). We note that either the 4-state Example 3a or 4-state Example 6a requires lower E_b/N_o (to achieve BER=10^{-6}) than that required by the 16-state Ungerboeck's TCM. Moreover, the 4-state Example 4a or 4-state Example 7a requires lower E_b/N_o (to achieve BER=10^{-6}) than that required by the 64-state conventional binary convolutional code. Hence, we can conclude that either of the analyzed schemes has better error performance and lower decoding complexity as compared to the conventional trellis code. According to Table 1, we see that the decoding delay of either of the analyzed schemes is longer than that of Hellstern's scheme.

Also included in Fig. 7 and Fig. 9 are the simulation results of a conventional convolutional code with a 256-states trellis. If a BER of 10^{-5} is needed in the communication design, 16-state Examples 4b and 7b require a similar E_b/N_o and a longer decoding delay but are with a much lower decoding complexity as compared to the conventional 256-state convolutional code.

Also included in Fig. 7 and Fig. 9 are the simulation results of a rate 1/2 turbo code using Log-MAP decoding with interleaving size of 512 message bits and $\nu = 2$ [7]. The number of iteration is 6 and a random interleaver is considered. Although the error performance of 4-states Examples 4a and 7a is worse than that of this turbo code, the decoding complexity

and decoding delay of 4-states Examples 4a and 7a are much lower than those of this 4-states turbo code.

Finally, by comparing Examples 6 and 7 with Examples 3 and 4, we see that if the same C is used, using the second scheme we can achieve similar error performance with less decoding delay as compared to using the first scheme, even though using the first scheme can achieve a larger lower bound on free distance.

V.2 Performance over independent Rayleigh fading

For a coding system in the independent Rayleigh fading channel, a large symbol distance is highly desired [10]. For a binary convolutional code using BPSK modulation, its free distance is the same as its symbol distance. We have already demonstrated that using trellis coding scheme with additional processing units, we can construct binary convolutional codes with large free distances. For a TCM, its symbol distance is different from its free distance. In the conventional TCM, one branch, $\hat{v}(t) = (v_1(t), \cdots, v_m(t)$ of C will affect one output symbol of the signal mapper. In Hellstern's scheme, the delay processor enables each code bit $v_j(t), j = 1, \cdots, m$, of C to affect one output symbol of the signal mapper. Hence, the introduction of the delay processor in Hellstern's scheme can increase the symbol distance of the trellis code. In the analyzed schemes, the multiple delay processors or the convolutional processor enable each code bit $v_j(t), j = 1, \cdots, m$, of C to affect more than one output symbol in some occasions. Hence, the introduction of multiple delay processors or the convolutional processor can further increase the symbol distance of the trellis code. However, the error coefficient can significantly affect the BER of a coding system in the independent Rayleigh fading channel. Hence, it is interesting to see the error performances of codes given in Examples 1-4 and 6-7 over the independent Rayleigh fading channel. Simulation results over the independent Rayleigh fading channel with perfect channel state information for Examples 1, 3, 6 and Examples 2, 4, 7 using the suboptimum decoding (with SOVA) are given in Fig. 10 and 11 respectively.

Also included in Fig. 11 are the simulation results of a conventional convolutional code with

a 256-states trellis. If a BER of 10^{-5} is needed in the communication design, 16-state Example 4b requires a similar E_b/N_o and a longer decoding delay but with a much lower decoding complexity as compared to the conventional 256-state convolutional code.

Also included in Fig. 11 are the simulation results of a rate 1/2 turbo code using Log-MAP decoding with interleaving size of 512 message bits and $\nu = 2$ [7]. The number of iteration is 6 and a random interleaver is considered. Although the error performance of Examples 4 and 7 is worse than that of this turbo code, the decoding complexity and decoding delay of Examples 4 and 7 are lower than that of this turbo code.

Finally, by comparing Examples 6 and 7 with Examples 3 and 4, we see that if the same C is used, using the first scheme we can achieve better error performance at low BER with longer decoding delay as compared to using the second scheme.

V.3 Performance over correlated Rayleigh fading

Often, successive fading amplitudes are correlated in the narrowband cellular mobile radio [13]. In this section, we evaluate the error performance of trellis coding with additional processing units when correlated fading is assumed and no interleaving is employed. We assume the power spectral density of the faded amplitude due to the Doppler shift is

$$V(f) = \frac{1}{2\pi\sqrt{f_D^2 - f^2}} \tag{23}$$

if $|f| < |f_D|$ and $V(f) = 0$, otherwise. The autocorrelation function $R_H(0; \Delta t)$ can be written as

$$R_H(0; \Delta t) = J_0(2\pi f_D \Delta t), \tag{24}$$

where $J_0(x)$ is the zero-order Bessel function of the first kind. In our simulations, we have assumed a vehicle speed of 60mi/h, a carrier frequency $f_c = 900$ MHz and a symbol rate of 8000 symbols/s. This results in a fade rate $f_D T_s = 0.01$. We also assume that we have perfect channel state information.

Simulation results over the correlated Rayleigh fading channel for Examples 1, 3, 6, and Examples 2, 4, 7 using the suboptimum decoding (with SOVA) are given in Fig. 12 and 13 respectively. Compared to the simulation results of Examples 1, 3 and 6 (TCM) in the independent Rayleigh fading channel, the correlation between successive received symbols results in severe degradation in the error performance of Examples 1, 3 and 6 (TCM). Hence a channel interleaver is still needed for Examples 1, 3 and 6 (TCM). Compared to the simulation results of Examples 2, 4 and 7 (binary codes) in the independent Rayleigh fading channel, the correlation between successive received symbols does not result in severe degradation in the error performance of Examples 2, 4 and 7. Hence a channel interleaver is not needed for Examples 2, 4 and 7 (binary codes) if the decoding delay is an important consideration.

Also included in Fig. 13 are the simulation results of a conventional convolutional code with a 256-states trellis. The superior error performance of Examples 2, 4 and 7 over the conventional convolutional code may be due to that the delay processor and the convolutional processor can serve a kind of interleaver. Hence the effect of correlation between successive received symbols can be reduced by the delay processor or the convolutional processor.

Also included in Fig. 13 are the simulation results of a rate 1/2 turbo code using Log-MAP decoding with interleaving size of 512 message bits and $\nu = 2$ [7]. The number of iteration is 6 and a random interleaver is considered. 4-state Example 4a can achieve a coding gain of 2.2 dB at a BER of 10^{-5} over this 4-state turbo code. Note that the decoding complexity of 4-state Example 4a is much lower than that of this 4-state turbo code.

Finally, by comparing Examples 6 and 7 with Examples 3 and 4, we see that if the same C is used, using the first scheme we can achieve better error performance at low BER with longer decoding delay as compared to using the second scheme.

VI Remarks

In this part, we concentrate on analyzing two kind of trellis codes with large constraint lengths. We apply these trellis coding schemes with proposed suboptimum decoding in the additive white Gaussian noise (AWGN) channel, independent fading channel and correlated fading channel and good error performance is observed. We also analyze the decoding delay and decoding complexity for the used decoding. With these results, trellis coding with additional processing units can achieve comparable and sometimes better error performance in the considered channels with moderate decoding complexity and moderate decoding delay as compared to the conventional trellis code and turbo code.

Appendix— Proofs of Theorem 2 and 3

A. Proof of Theorem 2

Consider the righthand side of (8). The condition of $\lambda^{(L)} - 1 \geq m\lambda^{(L-1)} - 1$ implies that

$$\sum_{t=0}^{\lambda^{(L)}-1} \sum_{j=1}^{m} v_j^{(L)}(t)\Delta_j = \sum_{t=0}^{\lambda^{(L)}-1} d_w^{(L)}(\hat{v}^{(L)}(t)) \geq \sum_{t=0}^{m\lambda^{(L-1)}-1} d_w^{(L)}(\hat{v}^{(L)}(t)). \tag{A-1}$$

Let $t = \tau_j^{(L-1)} + \rho = (m-j)\lambda^{(L-1)} + \rho$. Then the summation index t for $0 \leq t \leq m\lambda^{(L-1)} - 1$ can be replaced by (ρ, j) for $0 \leq \rho < \lambda^{(L-1)}$, $1 \leq j \leq m$. For $i < j$, we have $t - \tau_i^{(L-1)} = \rho - (j-i)\lambda^{(L-1)} < 0$. Since $\hat{v}^{(L-1)}(p) = \hat{0}$ for $p < 0$, then $v_i^{(L-1)}(t - \tau_i^{(L-1)}) = 0$. Hence $s_i^{(L-1)}(t) = 0$. It follows from (10) that

$$d_w^{(L)}(\hat{v}^{(L)}(t)) = d_w^{(L)}(\omega^{(L-1)}(\hat{s}^{(L-1)}(t))) \geq s_j^{(L-1)}(t)\delta_j^{(L-1)} = v_j^{(L-1)}(\rho)\delta_j^{(L-1)}. \tag{A-2}$$

Then,

$$\sum_{t=0}^{\lambda^{(L)}-1} \sum_{j=1}^{m} v_j^{(L)}(t)\Delta_j \geq \sum_{\rho=0}^{\lambda^{(L-1)}-1} \sum_{j=1}^{m} v_j^{(L-1)}(\rho)\delta_j^{(L-1)} = \sum_{t=0}^{\lambda^{(L-1)}-1} d_w^{(L-1)}(\hat{v}^{(L-1)}(t)). \tag{A-3}$$

We can similarly derive that $\sum_{t=0}^{\lambda^{(L)}-1} \sum_{j=1}^{m} v_j^{(L)}(t)\Delta_j \geq \sum_{t=0}^{\lambda^{(\ell)}-1} d_w^{(\ell)}(\hat{v}^{(\ell)}(t))$ for $\ell = L-2, \cdots, 1$. Then, $\Delta_{free} \geq \Delta_{LB}(C^{(L)}, \lambda^{(L)}, \Delta_j) \geq \min_{\bar{v}^{(1)} \in C, \hat{v}^{(1)}(0) \neq \hat{0}} \sum_{t=0}^{\lambda^{(1)}-1} d_w^{(1)}(\hat{v}^{(1)}(t))$. □

B. Proof of Theorem 3

Consider the sequence $\bar{v}$ with $\hat{v}(0) \neq \hat{0}$ and $\hat{v}(t) = \hat{0}$ for $t < 0$. We may divide the sequence $\bar{v}$ into λ disjoint subsequences, $\bar{v}(0)$, $\bar{v}(1)$,$\cdots$, $\bar{v}(\lambda-1)$, where $\bar{v}(h) = \{\cdots, \hat{v}(-\lambda+h), \hat{v}(h), \hat{v}(\lambda+h), \cdots\}$, $h = 0, 1, \cdots, \lambda-1$. Let $\bar{s}$ and $\bar{z}$ be the corresponding sequences for $\bar{v}$. We then divide each of $\bar{s}$ and $\bar{z}$ into λ subsequences similarly. The subsequence $\bar{z}(h)$ will only depend on $\bar{v}(h)$ among all the λ subsequences of $\bar{v}$. Now we will consider the distance property related to subsequences $\bar{v}(0), \bar{s}(0)$ and $\bar{z}(0)$. Let $z_0 = \omega(\hat{0})$ and $\bar{z}_0 = \{\cdots, z_0, z_0, \cdots\}$. We will prove that $\Delta(\bar{z}(0), \bar{z}_0)$, which is the distance measure between $\bar{z}(0)$ and $\bar{z}_0$, is lower bounded by $\sum_{j=1}^{m} v_j(0)(\Delta_j + \ell_j \Delta_{j-1})$. Then, we can similarly show that the distance measure between $\bar{z}(h)$ and $\bar{z}_0$ is lower bounded by $\sum_{j=1}^{m} v_j(h)(\Delta_j + \ell_j \Delta_{j-1})$. Then, it is easy to see that the bound given in (22) is true. In the following, we first assume $\ell_j > 0$ for $2 \leq j \leq m$. The case for $\ell_j = 0$ will be considered later. Let $\alpha_j = \beta_j - \ell_{j+1}$ for $1 \leq j \leq m-1$, $\alpha_0 = \beta_1 + 1$ and $\alpha_m = 0$. It can be checked that $\alpha_{j-1} = \beta_j + \zeta_j + 1$ for $1 \leq j \leq m$. It is evident that

$$\begin{aligned} \Delta(\bar{z}(0), \bar{z}_0) &\geq \sum_{t=0}^{\beta_1} \Delta(z(t\lambda), z_0) \\ &= \sum_{t=\alpha_m}^{\alpha_{m-1}-1} \Delta(z(t\lambda), z_0) + \sum_{t=\alpha_{m-1}}^{\alpha_{m-2}-1} \Delta(z(t\lambda), z_0) + \cdots + \sum_{t=\alpha_1}^{\alpha_0-1} \Delta(z(t\lambda), z_0). \quad \text{(B-1)} \end{aligned}$$

Consider $\alpha_j \leq t < \alpha_{j-1}$ for $2 \leq j \leq m$. Since $t < \alpha_{j-1}$, then $t - \beta_i < t - (\beta_i - 1) < \cdots < t - (\beta_i - \ell_{i+1}) = t - \alpha_i < 0$ for $i < j$. Since $\hat{v}(p) = 0$ for $p < 0$, from (19), we have $s_i(t\lambda) = v_i((t-\beta_i)\lambda) \oplus v_{i+1}((t-(\beta_i-1))\lambda) \oplus \cdots \oplus v_{i+1}((t-\alpha_i)\lambda) = 0$ for $1 \leq i < j$. Hence $\Delta(z(t\lambda), z_0) \geq s_j(t\lambda)\Delta_j$ for $2 \leq j \leq m$. Moreover, it is clear that $\Delta(z(t\lambda), z_0) \geq s_1(t\lambda)\Delta_1$ for $\alpha_1 \leq t < \alpha_0$. Let

$$Y_j = \sum_{t=\alpha_j}^{\alpha_{j-1}-1} s_j(t\lambda)\Delta_j \quad \text{for } 1 \leq j \leq m. \quad \text{(B-2)}$$

Then, (B-1) implies $\Delta(\bar{z}(0), \bar{z}_0) \geq Y_m + Y_{m-1} + \cdots + Y_1$.

For $p \in \{0,1\}$, and $q_1, q_2, \cdots, q_k \in \{0,1\}$, if $k \leq \ell_j$, the condition of $\Delta_j \geq \ell_j \Delta_{j-1}$ implies $p\Delta_j + q_1\Delta_{j-1} + q_2\Delta_{j-1} + \cdots + q_k\Delta_{j-1} \geq (p \oplus q_1)\Delta_{j-1} + \cdots + (p \oplus q_k)\Delta_{j-1}$. Moreover, it can

be checked that for $p_i, q_j \in \{0,1\}$, if $k \le h$, then

$$\sum_{i=1}^{h} p_i \Delta_j + \sum_{i=1}^{k} q_i \Delta_{j-1} \ge \sum_{i=1}^{k} (q_i \oplus p_i \oplus p_{i-1} \oplus \cdots \oplus p_l)\Delta_{j-1}, \tag{B-3}$$

where $l = \max\{i - (\ell_j - 1), 1\}$. Note that each p_i appears in the summation of the right-hand side of (B-3) at most ℓ_j times.

Let $W_j = \sum_{i=j}^{m} Y_i$ and $Y'_j = \sum_{i=1}^{\zeta_j} v_j(i\lambda)\Delta_j$ for $1 \le j \le m$. From (18) and $\beta_m = 0$, we have

$$W_m = Y_m = \sum_{t=\alpha_m}^{\alpha_{m-1}-1} s_m(t\lambda)\Delta_m = \sum_{t=0}^{\zeta_m} v_m(t\lambda)\Delta_m = v_m(0)\Delta_m + Y'_m. \tag{B-4}$$

Then, $W_{m-1} = Y_m + Y_{m-1} = v_m(0)\Delta_m + Y'_m + Y_{m-1}$. From (B-3) and the condition of $\zeta_m \ge \zeta_{m-1} + \ell_m = \zeta_{m-1} + \beta_{m-1} - \alpha_{m-1} = (\alpha_{m-2} - 1) - \alpha_{m-1}$, we have

$$\begin{aligned} Y'_m + Y_{m-1} &= \sum_{t=1}^{\zeta_m} v_m(t\lambda)\Delta_m + \sum_{t=\alpha_{m-1}}^{\alpha_{m-2}-1} s_{m-1}(t\lambda)\Delta_{m-1} \\ &= s_{m-1}(\alpha_{m-1}\lambda)\Delta_{m-1} + \sum_{t=1}^{\zeta_m} v_m(t\lambda)\Delta_m + \sum_{t=1}^{\zeta_{m-1}+\ell_m} s_{m-1}((\alpha_{m-1}+t)\lambda)\Delta_{m-1} \\ &\ge h_0\Delta_{m-1} + h_1\Delta_{m-1} + \cdots + h_{\ell_m+\zeta_{m-1}}\Delta_{m-1}, \end{aligned} \tag{B-5}$$

where

$$h_0 = s_{m-1}(\alpha_{m-1}\lambda), \tag{B-6}$$

and for $1 \le i \le \ell_m + \zeta_{m-1}$,

$$h_i = s_{m-1}((\alpha_{m-1}+i)\lambda) \oplus v_m(i\lambda) \oplus v_m((i-1)\lambda) \oplus \cdots \oplus v_m(l\lambda), \tag{B-7}$$

where $l = \max\{i - (\ell_m - 1), 1\}$. It follows from (19) that

$$\begin{aligned} s_{m-1}((\alpha_{m-1}+i)\lambda) &= v_{m-1}((\alpha_{m-1}+i-\beta_{m-1})\lambda) \oplus \\ &\quad v_m((\alpha_{m-1}+i-(\beta_{m-1}-1))\lambda) \oplus \cdots \oplus v_m((\alpha_{m-1}+i-\alpha_{m-1})\lambda) \\ &= v_m(i\lambda) \oplus \cdots \oplus v_m((i-(\ell_m-1))\lambda) \oplus v_{m-1}((i-\ell_m)\lambda). \end{aligned} \tag{B-8}$$

Since $\hat{v}(p) = 0$ for $p < 0$, from (B-6), (B-7) and (B-8) we have

$$h_i = \begin{cases} v_m(0) & \text{for } i = 0, 1, \cdots, \ell_m - 1, \\ v_{m-1}((i-\ell_m)\lambda) & \text{for } i = \ell_m, \cdots, \ell_m + \zeta_{m-1}. \end{cases} \tag{B-9}$$

Therefore,

$$\begin{aligned} Y'_m + Y_{m-1} &\geq (\ell_m v_m(0) + v_{m-1}(0))\Delta_{m-1} + \sum_{t=1}^{\zeta_{m-1}} v_{m-1}(t\lambda)\Delta_{m-1} \\ &= (\ell_m v_m(0) + v_{m-1}(0))\Delta_{m-1} + Y'_{m-1}. \end{aligned} \tag{B-10}$$

In general, we can replace m in equations (B-5)–(B-9) by j for $2 \leq j \leq m$ to yield

$$Y'_j + Y_{j-1} \geq (\ell_j v_j(0) + v_{j-1}(0))\Delta_{j-1} + Y'_{j-1}. \tag{B-11}$$

Thus,

$$\begin{aligned} W_1 &= Y_m + Y_{m-1} + \cdots + Y_1 \geq v_m(0)\Delta_m + Y'_m + Y_{m-1} + \cdots + Y_1 \\ &\geq v_m(0)\Delta_m + (\ell_m v_m(0) + v_{m-1}(0))\Delta_{m-1} + Y'_{m-1} + Y_{m-2} + \cdots + Y_1 \\ &\geq v_m(0)\Delta_m + (\ell_m v_m(0) + v_{m-1}(0))\Delta_{m-1} + \cdots + (\ell_2 v_2(0) + v_1(0))\Delta_1 \\ &= v_m(0)(\Delta_m + \ell_m\Delta_{m-1}) + \cdots + v_2(0)(\Delta_2 + \ell_2\Delta_1) + v_1(0)\Delta_1. \end{aligned} \tag{B-12}$$

Note that in the above proof, we assume $\ell_j > 0$ for $2 \leq j \leq m$. If $\ell_j = 0$ for some j, $2 \leq j \leq m$, then $\zeta_j = 0$ and $s_{j-1}(t) = v_{j-1}(t - \beta_{j-1}\lambda)$. We have $\alpha_{j-1} = \beta_{j-1} - \ell_j = \beta_{j-1}$ and $\alpha_{j-2} = \beta_{j-2} - \ell_{j-1} = \beta_{j-1} + \zeta_{j-1} + 1$. Hence,

$$\begin{aligned} Y'_j + Y_{j-1} &= \sum_{i=1}^{\zeta_j} v_j(i\lambda)\Delta_j + \sum_{i=\beta_{j-1}}^{\beta_{j-1}+\zeta_{j-1}} s_{j-1}(i\lambda)\Delta_{j-1} \\ &= 0 + \sum_{i=0}^{\zeta_{j-1}} v_{j-1}(i\lambda)\Delta_{j-1} = v_{j-1}(0)\Delta_{j-1} + Y'_{j-1}. \end{aligned} \tag{B-13}$$

Hence equations (B-11), (B-12) still hold. Therefore, $\Delta(\bar{z}(0), \bar{z}_0)$ is lower bounded by $\sum_{j=1}^{m} v_j(0) \cdot (\Delta_j + \ell_j\Delta_{j-1})$. Then, Δ_{free} is lower bounded by (22). □

References

[1] G. Hellstern, "Coded Modulation with Feedback Decoding Trellis Codes," in *Proc. IEEE Int. Conf. on Communications* (Geneva, Switzerland, May 1993), pp. 1071–1075.

[2] G. Ungerboeck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory,* vol. 28, pp. 55–66, Jan. 1982.

[3] J.Y. Wang and M.C. Lin, "On Constructing Trellis Codes with Large Free Distances and Low Decoding Complexities," *IEEE Trans. Commun.,* vol. 45, no. 9, pp. 1017–1020, Sept. 1997.

[4] G. Pottie and D. Taylor, "Multilevel codes based on partitioning," *IEEE Trans. Inform. Theory,* vol. 35, pp. 87–98, Jan. 1989.

[5] J. Hagenauer, "Source-controlled channel decoding," *IEEE Trans. Commun.,* vol. 43, no. 9, pp. 2449–2457, Sept. 1995.

[6] C. Chang and K. Yao, "Systolic array processing of the Viterbi algorithm," *IEEE Trans. Inform. Theory,* vol. 35, pp. 76–86, Mar. 1989.

[7] 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.

[8] L. C. Perez, J. Segher, and D. J. Costello, "A distance spectrum interpretation of turbo codes," *IEEE Trans. Inform. Theory,* vol. 42, no. 6, pp. 409–428, Nov. 1996.

[9] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. Inform. Theory,* vol. 42, pp. 409–428, Mar. 1996.

[10] D. Divsalar and M.K. Simon, "The design of trellis coded MPSK for fading channels: Performance criteria," *IEEE Trans. Commun.,* vol. 36, pp. 1004-1012, Sept. 1988.

[11] E. Zehavi, "8-PSK trellis codes for a Rayleigh channel," *IEEE Trans. Commun.,* vol. 40, no 5, pp. 873–884, May 1992.

[12] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Trans. Inform. Theory,* vol. 44, no. 3, pp. 927–946, May 1998.

[13] W. Jakes, Microwave Mobile Communications. New York: Wiley, 1974.

[14] J-Y Wang and M-C Lin, "A trellis coded modulation scheme with a convolutional processor," 1997 IEEE International symposium on Information Theory, Ulm, Germany, June 29-July 4, 1997

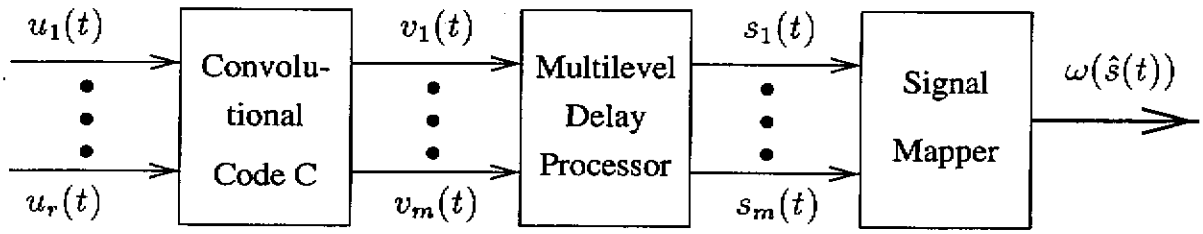


Figure 1: Encoding structure of trellis codes in [1] and [3].

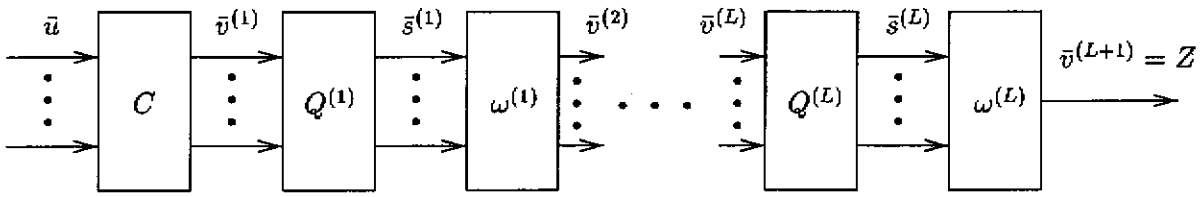


Figure 2: Encoding of a trellis code using multiple pairs of delay processors and signal mappers.

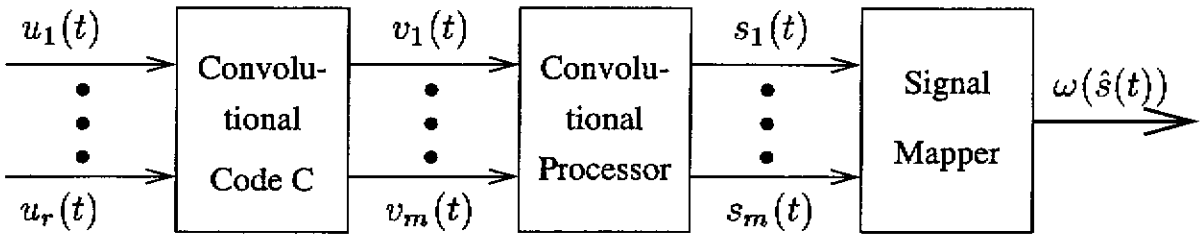


Figure 3: Encoding structure of a trellis code with a convolutional processor.

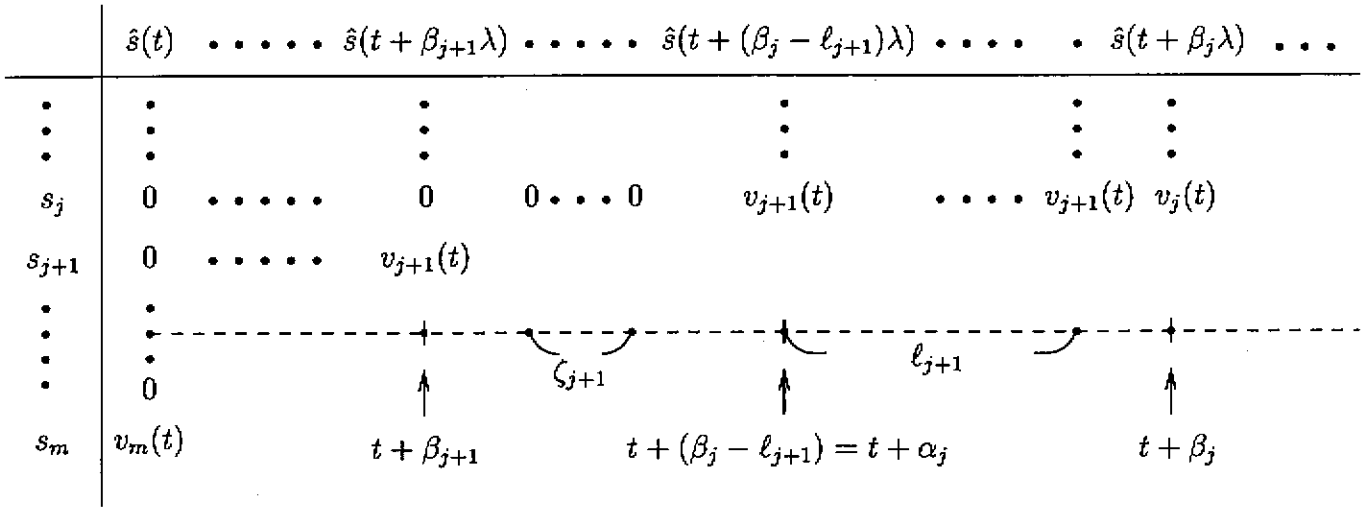


Figure 4: Relation between sequences $\bar{s}$ and $\bar{v} = (\cdots, \hat{0}, \hat{0}, \hat{v}(t), \hat{0}, \hat{0}, \cdots)$ described by (19), where $\alpha_j = \beta_j - \ell_{j+1}$.

	$\hat{s}(t)$	$\hat{s}(t+1\lambda)$	$\hat{s}(t+2\lambda)$	$\hat{s}(t+3\lambda)$	$\hat{s}(t+4\lambda)$
s_1	$v_1(t-4\lambda)$ $\oplus$ $v_2(t-3\lambda)$	$v_1(t-3\lambda)$ $\oplus$ $v_2(t-2\lambda)$	$v_1(t-2\lambda)$ $\oplus$ $v_2(t-1\lambda)$	$v_1(t-1\lambda)$ $\oplus$ $v_2(t)$	$v_1(t)$ $\oplus$ $v_2(t+1\lambda)$
s_2	$v_2(t-1\lambda)$	$v_2(t)$	$v_2(t+1\lambda)$	$v_2(t+2\lambda)$	$v_2(t+3\lambda)$
s_3	$v_3(t)$	$v_3(t+1\lambda)$	$v_3(t+2\lambda)$	$v_3(t+3\lambda)$	$v_3(t+4\lambda)$

Figure 5: Relation between sequences $\bar{s}$ and $\bar{v}$ for Example 5.

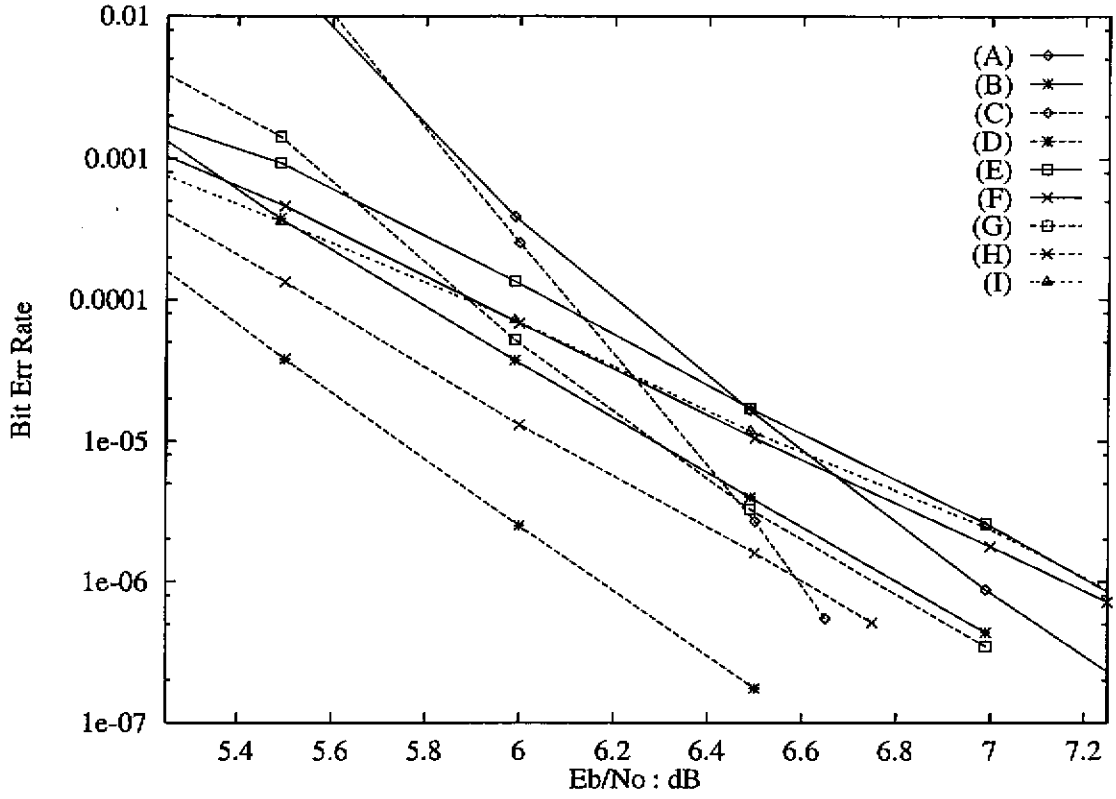


Figure 6: Simulation results for Examples 1 and 3 in the AWGN channel. (A): 3a, without SOVA, $\nu = 2$, $\lambda^{(1)} = 30$, $\lambda^{(2)} = 90$. (B): 3a, with SOVA, $\nu = 2$, $\lambda^{(1)} = 30$, $\lambda^{(2)} = 90$. (C): 3b, without SOVA, $\nu = 4$, $\lambda^{(1)} = 60$, $\lambda^{(2)} = 180$. (D): 3b, with SOVA, $\nu = 4$, $\lambda^{(1)} = 60$, $\lambda^{(2)} = 180$. (E): 1a, without SOVA, $\nu = 2$, $\lambda = 20$. (F): 1a, with SOVA, $\nu = 2$, $\lambda = 20$. (G): 1b, without SOVA, $\nu = 4$, $\lambda = 40$. (H): 1b, with SOVA, $\nu = 4$, $\lambda = 40$. (I): Ungerboeck's TCM, $\nu = 4$, $\lambda = 24$.

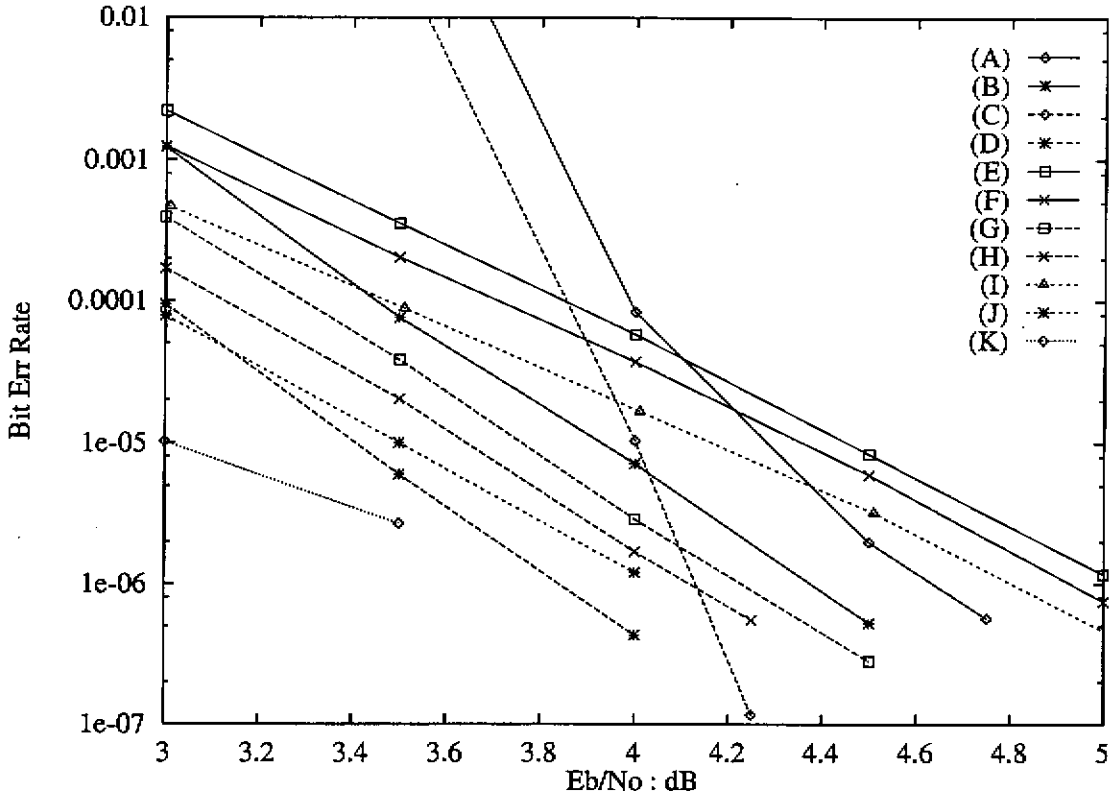


Figure 7: Simulation results for Examples 2 and 4 in the AWGN channel. (A): 4a, without SOVA, $\nu = 2$, $\lambda^{(1)} = 20$, $\lambda^{(2)} = 80$. (B): 4a, with SOVA, $\nu = 2$, $\lambda^{(1)} = 20$, $\lambda^{(2)} = 80$. (C): 4b, without SOVA, $\nu = 4$, $\lambda^{(1)} = 40$, $\lambda^{(2)} = 160$. (D): 4b, with SOVA, $\nu = 4$, $\lambda^{(1)} = 40$, $\lambda^{(2)} = 160$. (E): 2a, without SOVA, $\nu = 2$, $\lambda = 20$. (F): 2a, with SOVA, $\nu = 2$, $\lambda = 20$. (G): 2b, without SOVA, $\nu = 4$, $\lambda = 40$. (H): 2b, with SOVA, $\nu = 4$, $\lambda = 40$. (I): rate 1/2 conventional convolutional code, $\nu = 6$, $\lambda = 36$. (J): rate 1/2 conventional convolutional code, $\nu = 8$, $\lambda = 45$. (K): rate 1/2 turbo code, with Log-MAP, $\nu = 2$, interleaver size: 512, iteration number: 6.

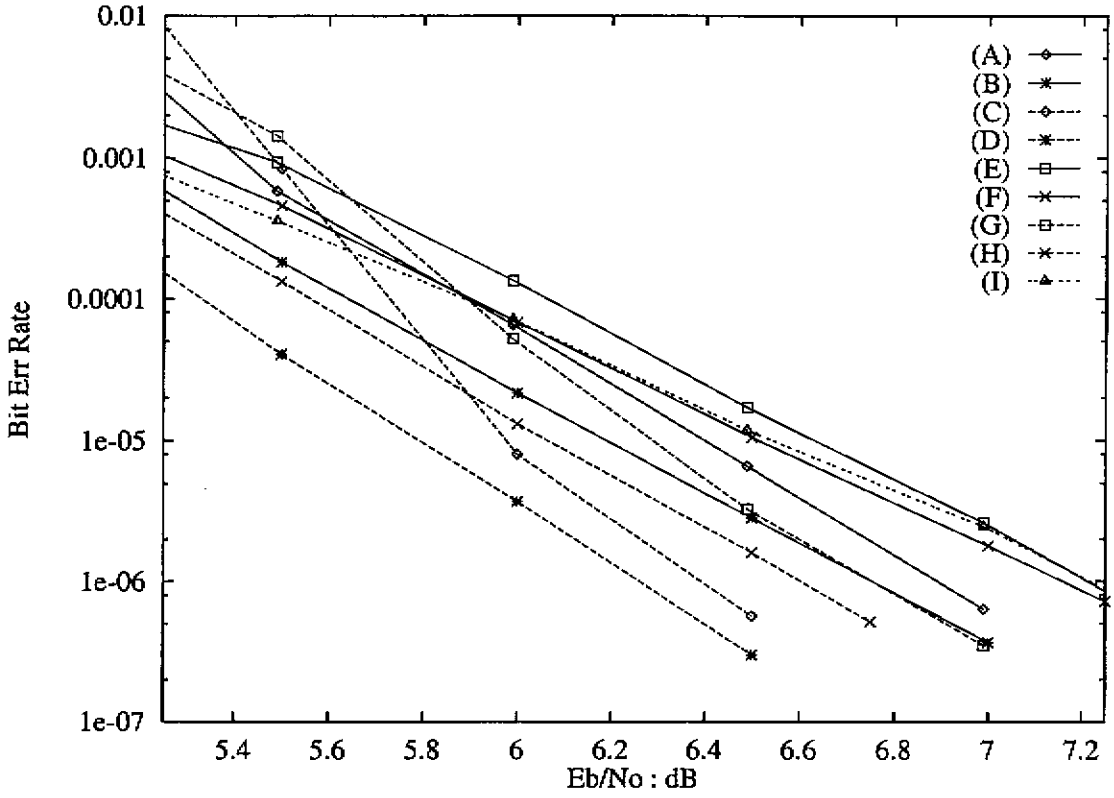


Figure 8: Simulation results for Examples 1 and 6 in the AWGN channel. (A): 6a, without SOVA, $\nu = 2$, $\lambda = 40$. (B): 6a, with SOVA, $\nu = 2$, $\lambda = 40$. (C): 6b, without SOVA, $\nu = 4$, $\lambda = 80$. (D): 6b, with SOVA, $\nu = 4$, $\lambda = 80$. (E): 1a, without SOVA, $\nu = 2$, $\lambda = 20$. (F): 1a, with SOVA, $\nu = 2$, $\lambda = 20$. (G): 1b, without SOVA, $\nu = 4$, $\lambda = 40$. (H): 1b, with SOVA, $\nu = 4$, $\lambda = 40$. (I): Ungerboeck's TCM, $\nu = 4$, $\lambda = 24$.

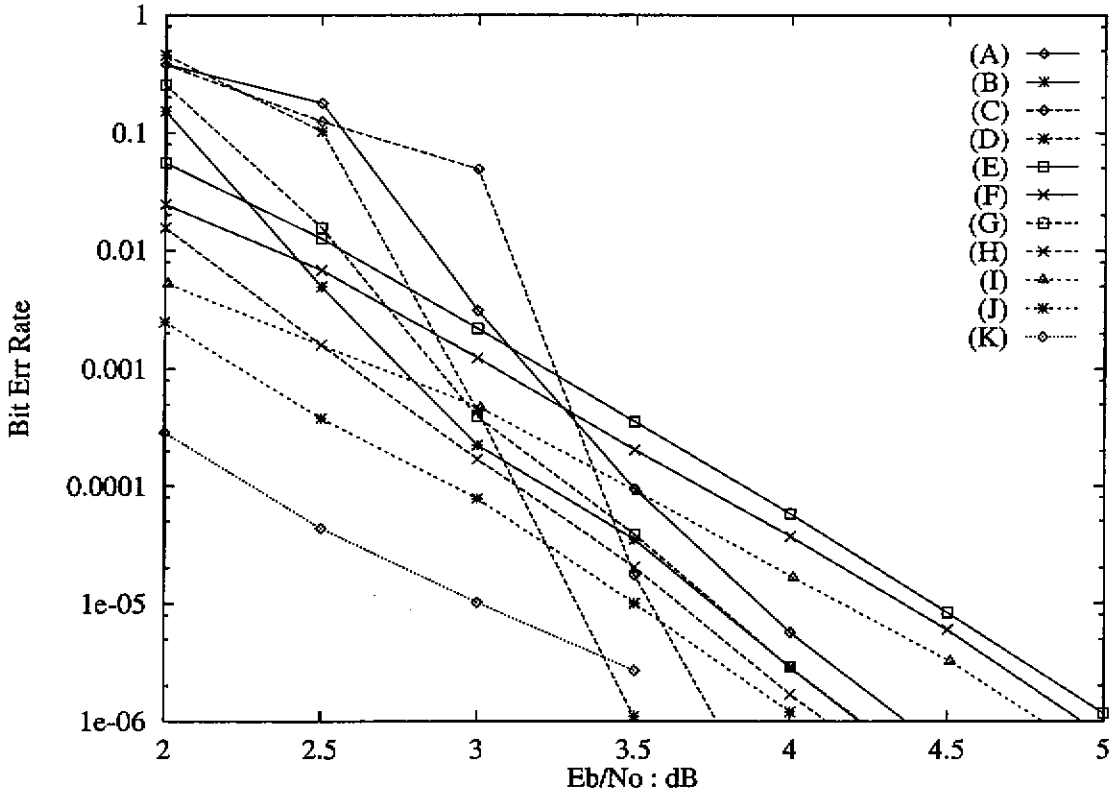


Figure 9: Simulation results for Examples 2 and 7 in the AWGN channel. (A): 7a, without SOVA, $\nu = 2$, $\lambda = 20$. (B): 7a, with SOVA, $\nu = 2$, $\lambda = 20$. (C): 7b, without SOVA, $\nu = 4$, $\lambda = 40$. (D): 7b, with SOVA, $\nu = 4$, $\lambda = 40$. (E): 2a, without SOVA, $\nu = 2$, $\lambda = 20$. (F): 2a, with SOVA, $\nu = 2$, $\lambda = 20$. (G): 2b, without SOVA, $\nu = 4$, $\lambda = 40$. (H): 2b, with SOVA, $\nu = 4$, $\lambda = 40$. (I): rate 1/2 conventional convolutional code, $\nu = 6$, $\lambda = 36$. (J): rate 1/2 conventional convolutional code, $\nu = 8$, $\lambda = 45$. (K): rate 1/2 turbo code, with Log-MAP, $\nu = 2$, interleaver size: 512, iteration number: 6.

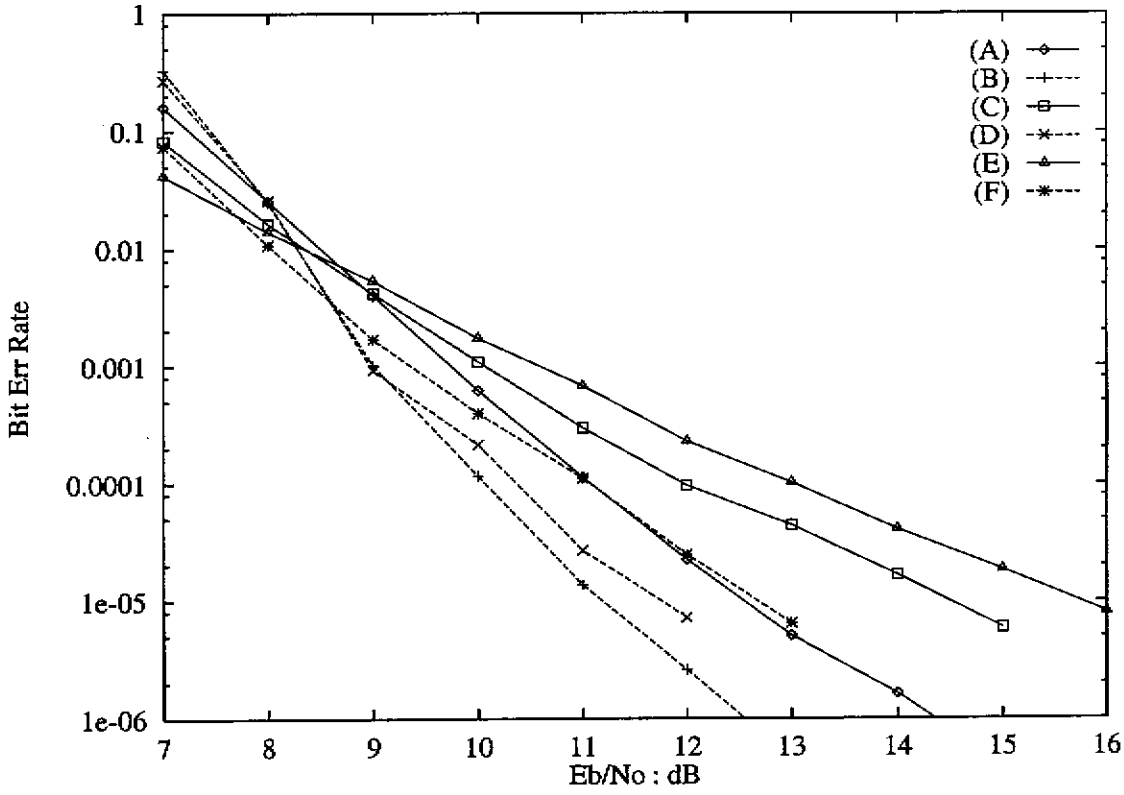


Figure 10: Simulation results for Examples 1, 3 and 6 in the independent Rayleigh fading channel. (A): 3a, with SOVA, $\nu = 2$, $\lambda^{(1)} = 30, \lambda^{(2)} = 90$. (B): 3b, with SOVA, $\nu = 4$, $\lambda^{(1)} = 60, \lambda^{(2)} = 180$. (C): 6a, with SOVA, $\nu = 2$, $\lambda = 40$. (D): 6b, with SOVA, $\nu = 4$, $\lambda = 80$. (E): 1a, with SOVA, $\nu = 2$, $\lambda = 20$. (F): 1b, with SOVA, $\nu = 4$, $\lambda = 40$.

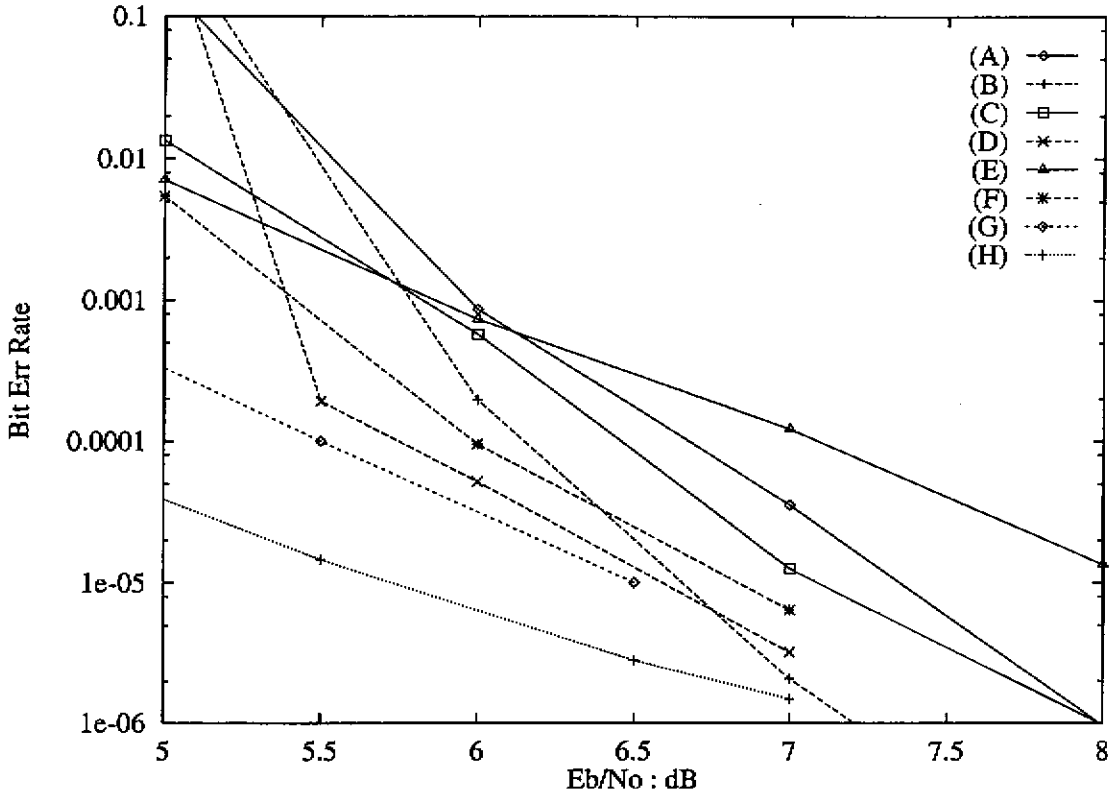


Figure 11: Simulation results for Examples 2, 4 and 7 in the independent Rayleigh fading channel. (A): 4a, with SOVA, $\nu = 2$, $\lambda^{(1)} = 20, \lambda^{(2)} = 80$. (B): 4b, with SOVA, $\nu = 4$, $\lambda^{(1)} = 40, \lambda^{(2)} = 160$. (C): 7a, with SOVA, $\nu = 2$, $\lambda = 20$. (D): 7b, with SOVA, $\nu = 4$, $\lambda = 40$. (E): 2a, with SOVA, $\nu = 2$, $\lambda = 20$. (F): 2b, with SOVA, $\nu = 4$, $\lambda = 40$. (G): rate $1/2$ conventional convolutional code, $\nu = 8$, $\lambda = 45$. (H): rate $1/2$ turbo code, with Log-MAP, $\nu = 2$, interleaver size: 512, iteration number: 6.

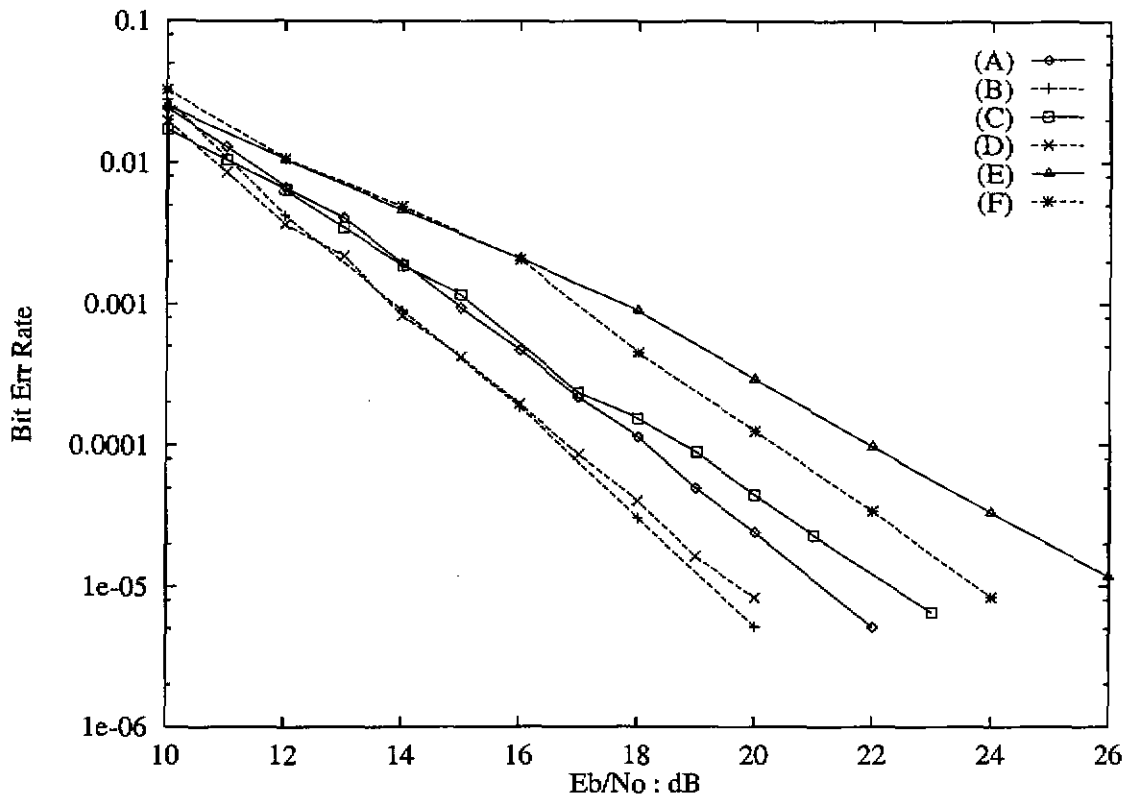


Figure 12: Simulation results for Examples 1, 3 and 6 in the correlated Rayleigh fading channel. (A): 3a, with SOVA, $\nu = 2$, $\lambda^{(1)} = 30, \lambda^{(2)} = 90$. (B): 3b, with SOVA, $\nu = 4$, $\lambda^{(1)} = 60, \lambda^{(2)} = 180$. (C): 6a, with SOVA, $\nu = 2$, $\lambda = 40$. (D): 6b, with SOVA, $\nu = 4$, $\lambda = 80$. (E): 1a, with SOVA, $\nu = 2$, $\lambda = 20$. (F): 1b, with SOVA, $\nu = 4$, $\lambda = 40$.

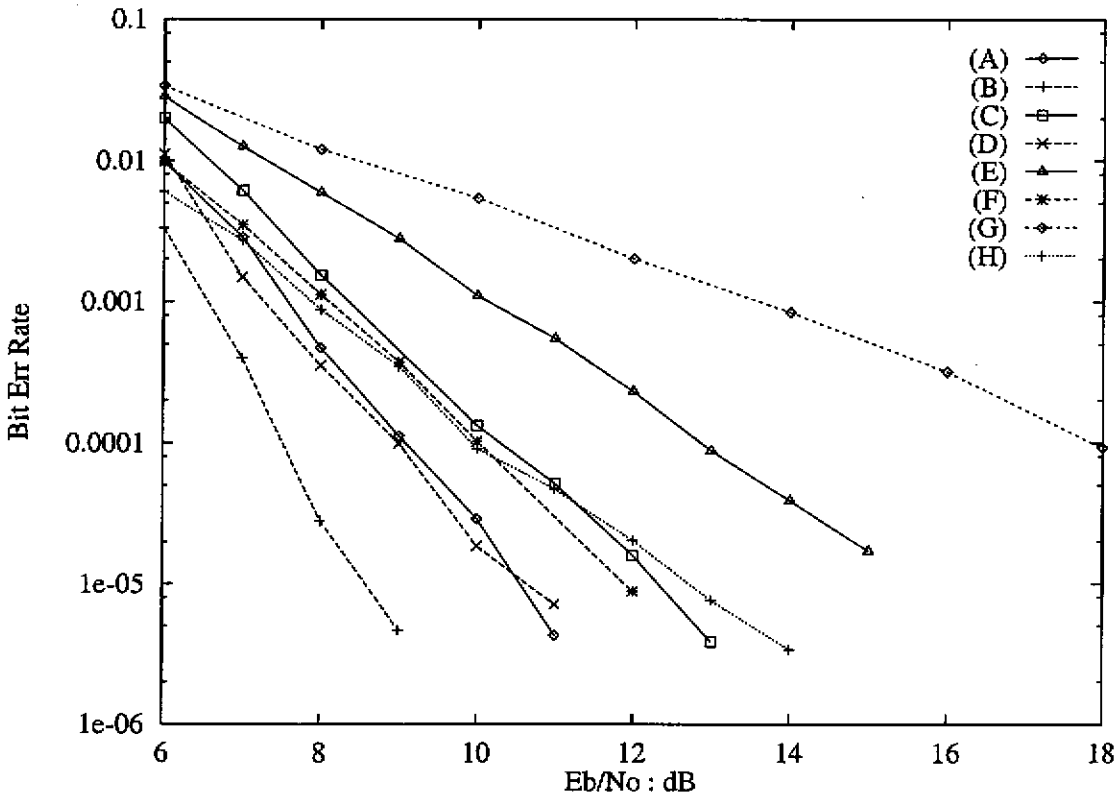


Figure 13: Simulation results for Examples 2, 4 and 7 in the correlated Rayleigh fading channel. (A): 4a, with SOVA, $\nu = 2$, $\lambda^{(1)} = 20, \lambda^{(2)} = 80$. (B): 4b, with SOVA, $\nu = 4$, $\lambda^{(1)} = 40, \lambda^{(2)} = 160$. (C): 7a, with SOVA, $\nu = 2$, $\lambda = 20$. (D): 7b, with SOVA, $\nu = 4$, $\lambda = 40$. (E): 2a, with SOVA, $\nu = 2$, $\lambda = 20$. (F): 2b, with SOVA, $\nu = 4$, $\lambda = 40$. (G): rate 1/2 conventional convolutional code, $\nu = 8$, $\lambda = 45$. (H): rate 1/2 turbo code, with Log-MAP, $\nu = 2$, interleaver size: 512, iteration number: 6.

Table 1: Data of the trellis codes given in Examples 1, 2, 3, 4, 6 and 7.

Example	Ω	$\frac{r}{m}$	ν	Δ_{LB}	(add, comp) (without SOVA)	delay (time units)	λ ($\lambda^{(1)}$)	delay (message b
1a	8PSK	$\frac{2}{3}$	2	6.34	(32, 20)	$3\lambda - 1$	20	118
3a	8PSK	$\frac{2}{3}$	2	7.52	(48, 28)	$9\lambda^{(1)} - 1$	30	538
6a	8PSK	$\frac{2}{3}$	2	7.17	(38, 30)	$5\lambda - 1$	40	398
1b	8PSK	$\frac{2}{3}$	4	8.93	(80, 56)	$3\lambda - 1$	40	238
3b	8PSK	$\frac{2}{3}$	4	10.69	(96, 64)	$9\lambda^{(1)} - 1$	60	1078
6b	8PSK	$\frac{2}{3}$	4	9.52	(86, 66)	$5\lambda - 1$	80	798
2a	$\{0,1\}^4$	$\frac{2}{4}$	2	12	(64, 34)	$4\lambda - 1$	20	158
4a	$\{0,1\}^4$	$\frac{2}{4}$	2	17	(112, 56)	$16\lambda^{(1)} - 1$	20	638
7a	$\{0,1\}^4$	$\frac{2}{4}$	2	14	(78, 70)	$9\lambda - 1$	20	358
2b	$\{0,1\}^4$	$\frac{2}{4}$	4	16	(112, 70)	$4\lambda - 1$	40	318
4b	$\{0,1\}^4$	$\frac{2}{4}$	4	24	(160, 92)	$16\lambda^{(1)} - 1$	40	1278
7b	$\{0,1\}^4$	$\frac{2}{4}$	4	20	(126, 106)	$9\lambda - 1$	40	718

第一節

緒論

在第三代行動通訊中之編碼技術，曾有人考慮使用傳統之256個狀態點之迴旋碼及渦輪碼(turbo code)來增加其錯誤更正的能力。不過使用256個狀態點之迴旋碼，因狀態點過多，因此其電路面積會因此大大地增加；而渦輪碼則採用疊帶(iterative)方式來解碼，硬體複雜度及解碼時間皆會大大地增加。

在部份A，我們已經仔細分析具有迴旋處理器之籬柵編碼架構以及其解碼方式，並且已用軟體分析證明具有迴旋處理器之籬柵碼應用於AWGN通道，獨立衰減通道(Independent fading channel)及相關衰減通道(correlated fading channel)中比傳統之迴旋碼及渦輪碼具有可匹配之錯誤更正能力。具有迴旋處理器之籬柵碼結構有低解碼狀態點數之優點，適合於多處理器方式之設計，而且不需要做到疊帶解碼步驟。而在此部份，我們將利用FPGA（Field -Programmable Gate Array）來驗證、設計此籬柵碼之腓特比解碼電路。

在第一年計劃裡，我們使用FPGA（Field-Programmable Gate Array）來實現一個碼率 $R=2/4$ 之具有迴旋處理器和信號對應器，且狀態數為4。在第二年計劃裡，我們使用FPGA來實現一個碼率 $R=2/4$ 之具有迴旋處理器和信號對應器，且狀態數為16，在第一年和第二年計畫中其迴授方式則是採用硬式迴授方式。在第三年計劃中，我們採用軟式回授(soft feedback)的方式來完成其碼率 $R=2/4$ 之具有迴旋處理器和信號對應器且狀態數為16的腓特比解碼電路。其軟式回授之值得到方式，我們將採用軟式-輸出腓特比演算法(SOVA:soft-output Viterb algorithm) [2-5]。

在本文之第二節裡，我們將介紹 4 個狀態點及 16 個狀態點之具有迴旋處理器的籬柵碼電路設計方式，其兩個籬柵碼之迴授方式則為硬式迴授方式。在第三節裡，我們將描述採用 SOVA 解碼器的設計架構，數位硬體設計之參數。第四節將描述硬體之測試結果與比較。在第五節，我們將分析具有迴旋處理器之籬柵碼與傳統迴旋碼及渦輪碼之硬體複雜度之比較。最後，在第六節裡我們會對此部份作個總結。

第二節

具迴旋處理器之籬柵碼之腓特比解碼器的電路設計

在此節，我們將介紹 4 個和 16 個狀態點之具有迴旋處理器的籬柵碼的腓特比解碼器電路設計之方法，而其迴授方式則是採用硬式迴授方式。

2.1 4 個狀態點之籬柵碼 T 之解碼器的電路設計

藉由使用迴旋碼 C 之解碼籬柵和一些迴授資訊，這個籬柵碼 T 可以被次佳化地解碼。因此，這個籬柵碼 T 之解碼器與傳統之腓特比解碼器的差異在於分支計量的計算需要作些修正且需加上額外之電路來產生迴授資訊。在這一小節裡，我們將詳細地描述 4 個狀態點之籬柵碼 T 之解碼器的電路設計，其中籬柵碼 T 由一個 $R=2/4$，$m=2$ 之傳統迴旋碼 C，後面接著一個迴旋處理器和一個信號對應器所組成。

2.1.1 4 個狀態點之籬柵碼 T 之解碼器的描述

圖 2.1 顯示了具迴旋處理器和 4 個狀態點之籬柵碼 T 的次佳化解碼器之方塊圖，其中迴旋碼 C 之編碼器是一個 $R=2/4$，$m=2$ 之迴旋編碼器。假設現在的時間點是 t。$\hat{r}_a$，$\hat{r}_b$，$\hat{r}_c$，$\hat{r}_d$，$\hat{r}_e$，$\hat{r}_f$，$\hat{r}_g$，$\hat{r}_h$和$\hat{r}_i$分別表示$\hat{r}(t)$，$\hat{r}(t-\lambda)$，$\hat{r}(t-2\lambda)$，$\hat{r}(t-3\lambda)$，$\hat{r}(t-4\lambda)$，$\hat{r}(t-5\lambda)$，$\hat{r}(t-6\lambda)$，$\hat{r}(t-7\lambda)$。另外，b_1^*，c_1^*，d_1^*，e_1^*，e_2^*，f_1^*，f_2^*，g_1^*，g_2^*，h_1^*，h_2^*，i_1^*，i_2^*和i_3^*分別表示 $v_1^*(t-9\lambda)$，$v_1^*(t-10\lambda)\oplus v_2^*(t-9\lambda)$，$v_1^*(t-11\lambda)\oplus v_2^*(t-10\lambda)$，$v_1^*(t-12\lambda)\oplus v_2^*(t-11\lambda)$，$v_2^*(t-9\lambda)$，$v_1^*(t-13\lambda)\oplus v_2^*(t-12\lambda)$，$v_2^*(t-10\lambda)\oplus v_3^*(t-9\lambda)$，$v_1^*(t-14\lambda)\oplus v_2^*(t-13\lambda)$，$v_2^*(t-11\lambda)\oplus v_3^*(t-10\lambda)$，$v_1^*(t-15\lambda)\oplus v_2^*(t-14\lambda)$，$v_2^*(t-12\lambda)\oplus v_3^*(t-11\lambda)$，$v_1^*(t-16\lambda)\oplus v_2^*(t-15\lambda)$，$v_2^*(t-13\lambda)\oplus v_3^*(t-12\lambda)$和$v_3^*(t-9\lambda)$。由於在解碼程序之步驟一到步驟四中，我們要算出分別對應v_1，v_2，v_3和v_4之位元 0 與位元 1 的最小位元計量，因此需將這些已收到的信號，$\hat{r}_a$，$\hat{r}_b$，$\hat{r}_c$，$\hat{r}_d$，$\hat{r}_e$，$\hat{r}_f$，$\hat{r}_g$，$\hat{r}_h$和$\hat{r}_i$，及已還原的信號，b_1^*，c_1^*，d_1^*，e_1^*，e_2^*，f_1^*，f_2^*，g_1^*，g_2^*，h_1^*，h_2^*，i_1^*，i_2^*和i_3^*，送進迴旋碼 C 之腓特比解碼器。其中，b_1^*，c_1^*，d_1^*，e_1^*，e_2^*，f_1^*，f_2^*，g_1^*，g_2^*，h_1^*，h_2^*，i_1^*，i_2^*和i_3^*這些資料是經由將先前由迴旋碼 C 之腓特

比解碼器所解碼出來的訊息位元u_1^*，u_2^*再次編碼並送入迴旋處理器後所獲得，而$u_1^*=u_1^*(t-9\lambda)$，$u_2^*=u_2^*(t-9\lambda)$。

從圖 2.1 我們可以看到，籬柵碼 T 之解碼器的設計重點在於迴旋碼 C 之腓特比解碼器。圖 2.2 顯示了這個迴旋碼 C 之腓特比解碼器的架構，它包含了三個部分：BMU，ACSU 和 SMU。在 BMU 這個部分裡，其包含了用來計算位元計量的四個小單元，分別是V_1，V_2，V_3和V_4，以及用來計算分支計量的單元，$\sum$。V_1，V_2，V_3和V_4四個小單元是用來計算分別對應迴旋碼 C 之編碼器所輸出之四個位元的位元計量 M_1，M_2，M_3和 M_4。我們將現在所收到的信號$\hat{r}_a$和先前所收到的信號$\hat{r}_b$，$\hat{r}_c$，$\hat{r}_d$，$\hat{r}_e$，$\hat{r}_f$，$\hat{r}_g$，$\hat{r}_h$和$\hat{r}_i$，以及先前所還原之信號b_1^*，c_1^*，d_1^*，e_1^*，e_2^*，f_1^*，f_2^*，g_1^*，g_2^*，h_1^*，h_2^*，i_1^*，i_2^*和i_3^*一併送進 BMU 來產生 M_{11}，M_{12}，M_{21}，M_{21}，M_{31}，M_{31}，M_{41}和 M_{41}，也就是 M_1 對應於$v_1(t-8\lambda)$等於 0 和 1，M_2 對應於$v_2(t-8\lambda)$等於 0 和 1，M_3 對應於$v_3(t-8\lambda)$等於 0 和 1，M_4 對應於$v_4(t-8\lambda)$等於 0 和 1。透過由加法器所組成之小單元 $\sum$，我們可以獲得在解碼步驟五裡所需要之 16 種可能的分支計量。將這些 16 種可能的分支計量表示成bm_0，…，bm_{15}，並且送進 ACSU 以獲得 4 個狀態點計量及相對應之存活路徑，p_0，p_1，p_2和p_3。這 4 個存活路徑被送進 SMU 並且儲存起來以便作為還原訊息位元的依據。最後，SMU 產生出我們想要的訊息位元，u_1^*，u_2^*。由於籬柵碼 T 的解碼籬柵只有 4 個狀態點，我們可以在 ACSU 裡採用併聯處理的方式以提升解碼速率。事實上，我們的解碼器也僅需要 4 個 ACS 處理器。

對於圖 2.2 之解碼器的電路設計，我們將在稍後作詳細的描述。

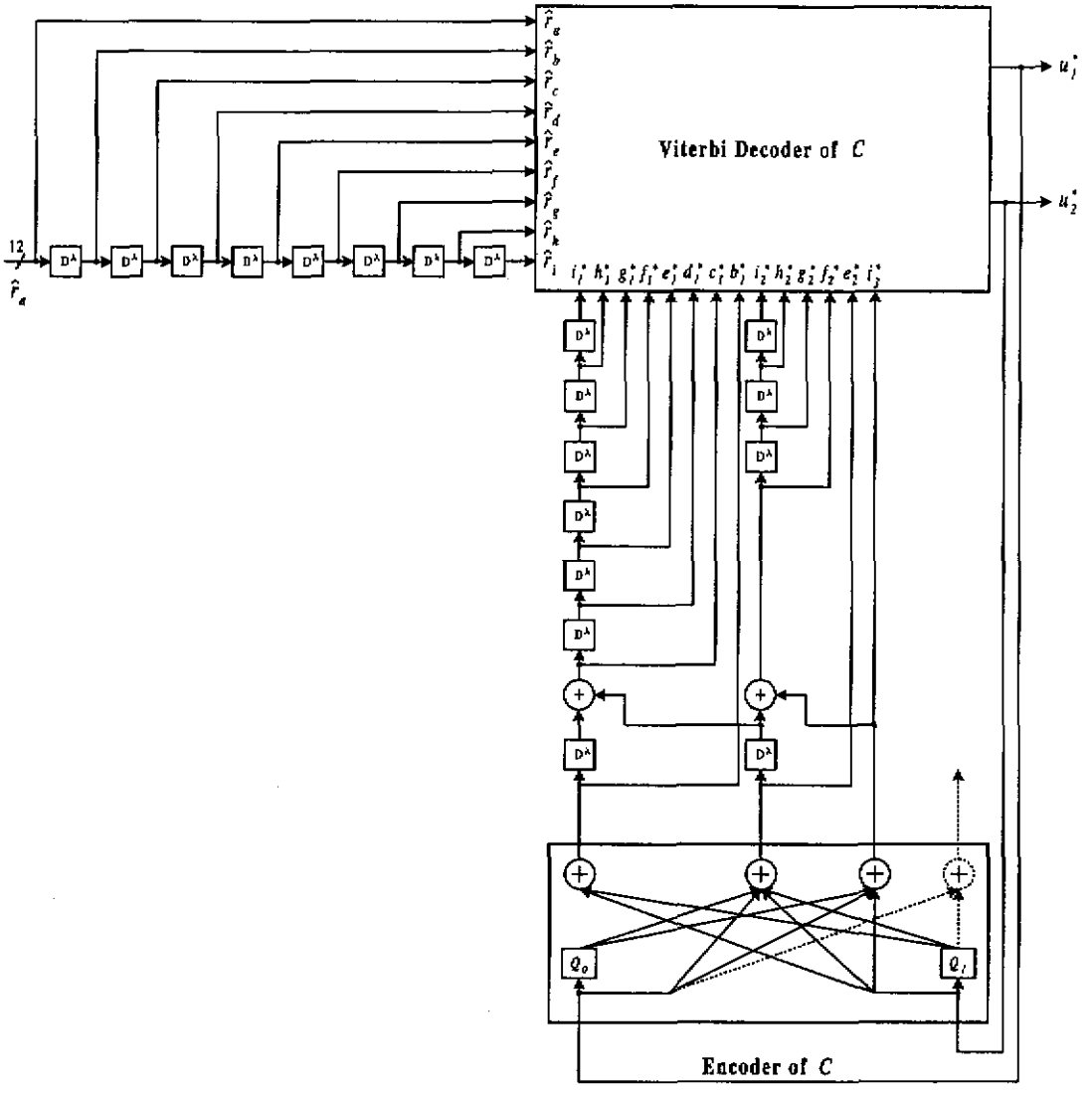


圖 2.1　4 個狀態點之籬柵碼 T 的解碼器方塊圖

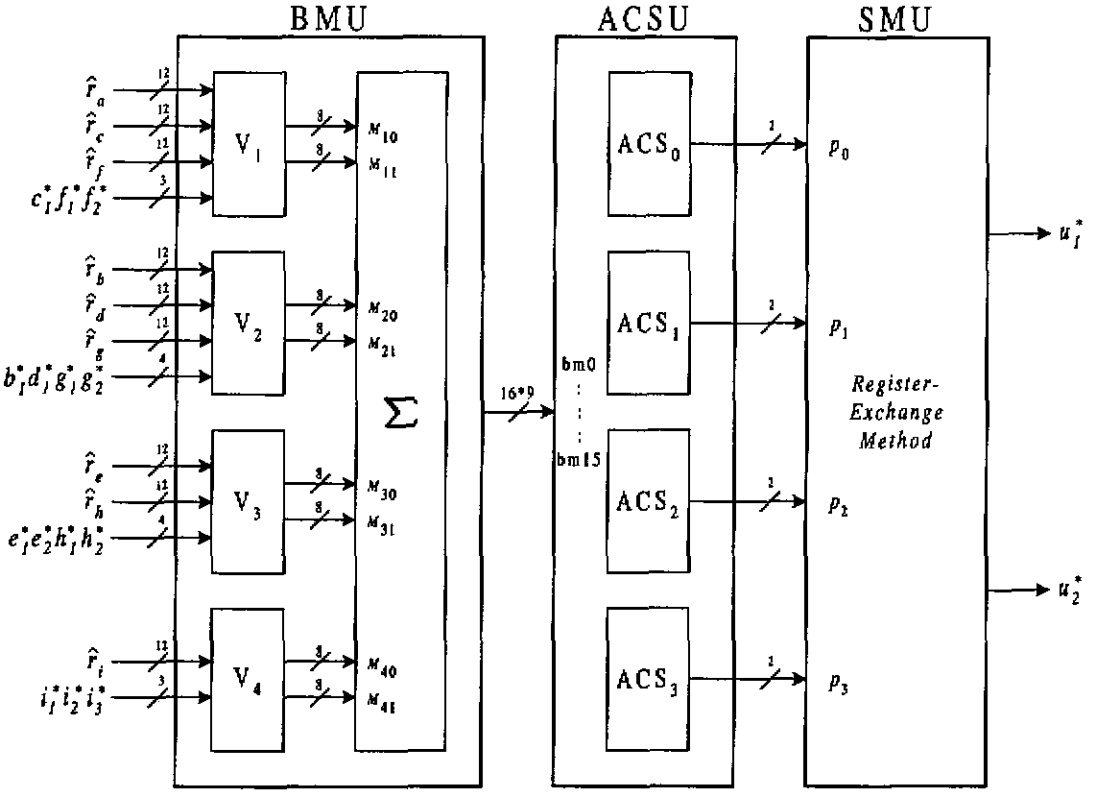


圖 2.2 迴旋碼 C 之腓特比解碼器

2.1.2 量化

就硬體實現而言，為了幫助解碼器之數位化處理，我們必須將所收到的連續信號進行量化。量化的方法分為兩種[6]：1) 均勻的量化 (uniform quantization)，和 2) 使用羅意德—魅斯理論 (Lloyd - Max algorithm) 之非均勻的量化 (non - uniform quantization)。前者是將信號變動的範圍內分解成數個等分的子區間來判斷出量化層級 (quantization level)，而後者之羅意德—魅斯理論是用來計算最佳化之量化器以使誤差失真之平方達到最小。針對這兩種量化方案 (quantization scheme)而言，兩者均獲得相似之性能，但因為均勻的量化理論比羅意德—魅斯理論較

顯著地低複雜，所以我們選擇前者之量化方法。

均勻的量化理論中最簡單的量化方式是硬式決策 (hard deC1ision)，如圖 2.3 (a)所示，也就是如果收到之連續信號大於 0 時則具有 0 為輸出，否則 1 為輸出。因此，所收到的信號被表示成每個碼符號（code symbol）為 1 個位元。但採用硬式決策時，與無限大的量化相比較，其在 E_b/N_o 通常伴隨著大約 2dB 的損失[7]。藉由將收到的信號量化成 4 或 8 個層級而非僅僅只有 2 個層級，則這些大部分的損失可以被償還。增加額外的層級將使得每一個收到信號之 2 個或 3 個位元的表示成為必要。圖 2.3 (b)、(c)和(d)顯示了三種量化方案，分別是 4 個層級、8 個層級和 16 個層級，而間距分別是 1.0、0.5 和 0.25。

在脒特比解碼器裡，由於符號計量 (symbol metric)必須被表示成數位型態，於是計量量化之效應便衍生出一個問題。模擬結果顯示了解碼器的性能對於符號計量量化相當地不敏感。事實上，相對於使用對數可能性 (log likelihood)而言，使用整數作為符號計量在 2 個，4 個，8 個層級之量化下僅造成可忽略的性能衰減[8]。圖 2.4 顯示了針對 8 層量化通道的一組計量。使用這些符號計量顯示了符號計量和所收到的符號本身一樣對於 2，4 或 8 個層級之量化可以分別地被表示成 1，2 或 3 個位元。

考慮 8 個層級之量化，在發射端將籬柵碼 T 之輸出經過 BPSK (Binary Phase Shift Keying)調變後送進 AWGN（Additive White Gaussian Noise）通道。在接收端，假設受雜訊干擾之收到的位元信號 $r_i(t)$ 被量化在第 r 個層級，其中 $r = 0, 1, \cdots, 7$，那麼由圖 3.4 可知，對於位元 0 和 1 之位元計量分別是 7-r 和 r。為了化簡電路設計，我們將位元計量作定量平移，於是可以得到位元 0 和 1 的位元計量分別是 7 和 2r。通常，對於任意 2 的次方之量化層級 Q 而言，位元 0 和 1 之位元計量分別是 Q-1 和 2r。

經由模擬結果顯示，如圖 2.5，使用量化層級 $Q = 8$ 之解碼錯誤率相

對於 Q =∞之解碼錯誤率僅僅損失不到 0.25dB。其中圖 2.5 裡之曲線（E）是傳統的迴旋碼，其 $R = 1/2$，記憶位元數 $m = 6$，且生成矩陣之八進制表示式為 $G = (155, 113)_8$。我們可以看到使用量化層級 $Q = 8$ 之籬柵碼 T 在 E_b/N_o = 3.5dB 以上其位元錯誤率比傳統之迴旋碼要好，所以為了減少計算量及存儲器之用量，我們採用量化層級 $Q = 8$。

另外，經由軟體模擬，如圖 2.6，我們發現在 $Q = 8$ 的條件下選擇常數 λ 大於 24 時在降低錯誤率方面只能獲得些微的好處。

對於 Q = 8，$2r$ 這個量之最大值為 2 ×7 = 14，因此我們需要用 4 個位元來表示 7 或 $2r$。也就是說，每一個計量 $M(r_i(t), \sigma_i)$ 被表示成 4 個位元。而層級計量 (level metric)是將 4 個子層級 (sublevel) 計量 $M(r_i(t), \sigma_i)$ 加總起來，也就是最大值為 14 ×4 = 56，因此我們需要用 6 個位元來表示之。由於 v_1，v_2，v_3 和 v_4 之位元計量分別包含 3，3，2，和 1 個層級計量，所以位元計量中之最大值為 56 ×3 = 168，也就是我們需要用 8 個位元來表示 1 個位元計量。在步驟 5 中，每一種可能之分支計量是將 4 個位元計量加總起來。由於分支計量之最大值為 56 ×(3 + 3 + 2 + 1) = 504，因此我們需要用 9 個位元來表示一個可能之分支計量。經由軟體模擬結果，如圖 9，使用 9 個位元來儲存存活路徑（survivor path）之路徑計量（path metric）是相當足夠的。當然，這個最好的路徑計量必須被解碼器儲存為狀態計量（state metric）。

綜合上述，對於 $R=2/4$、$m=2$ 之具有迴旋處理器的二元迴旋碼之電路設計，我們列出規格如下：

- 狀態點數：4 個，
- 量化層級的數目：$Q=8$，
- 截斷長度（及延遲常數）：$\lambda=24$，
- 碼率：$R=2/4$，
- 生成矩陣：$G_C=\begin{pmatrix}0 & 3 & 3 & 2\\ 3 & 3 & 2 & 1\end{pmatrix}$
- 信號對應器：$W=\begin{pmatrix}0 & 0 & 0 & 1\\ 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1\\ 1 & 1 & 1 & 1\end{pmatrix}$

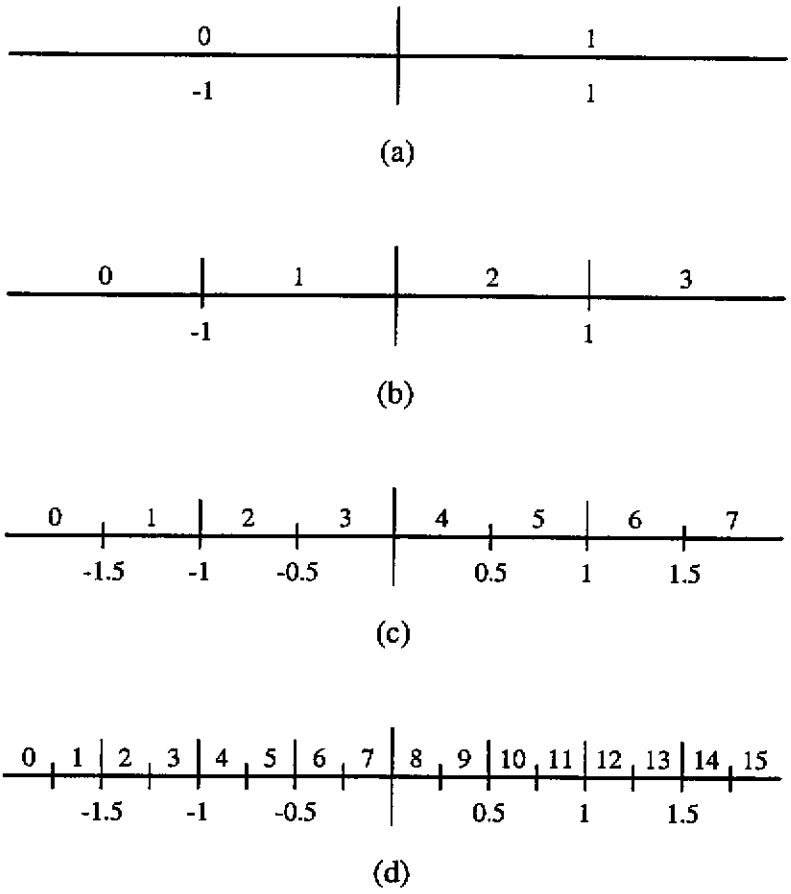


圖 2.3　量化臨界和區間：(a) 2 層之量化、(b) 4 層之量化、(c) 8 層之量化、(d) 16 層之量化

Quantization Level

Code Symbol	0	1	2	3	4	5	6	7
0	7	6	5	4	3	2	1	0
1	0	1	2	3	4	5	6	7

圖 2.4　8 層量化的符號計量

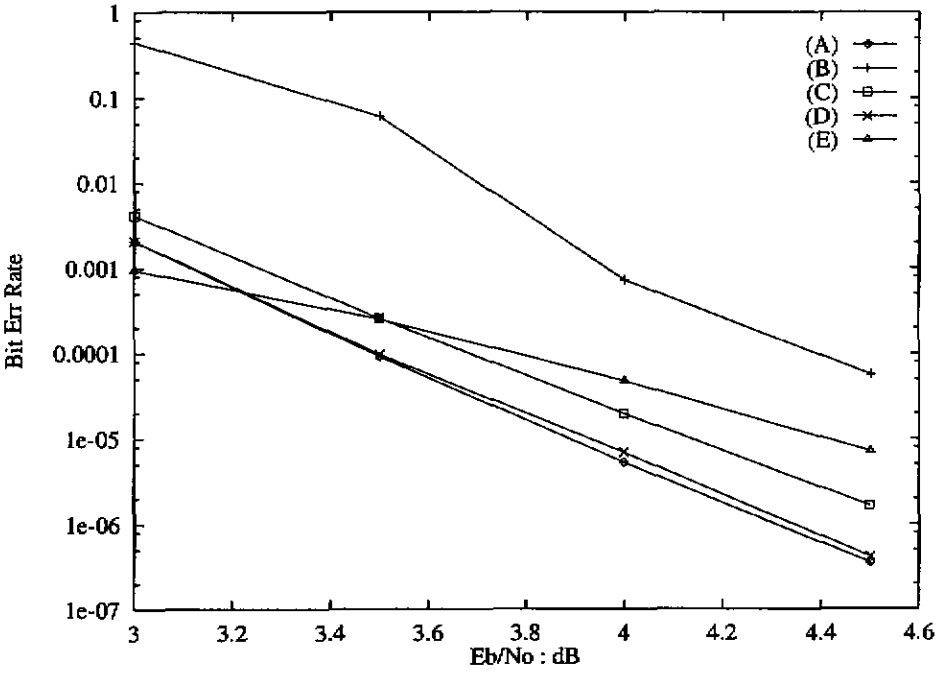


圖 2.5　籬柵碼 T 在不同之量化層級下的模擬結果($m=2$，$\lambda=20$)
(A) Q = ∞，(B) Q = 4，(C) Q = 8，(D) Q = 16，
(E) 具 64 個狀態點之傳統的迴旋碼

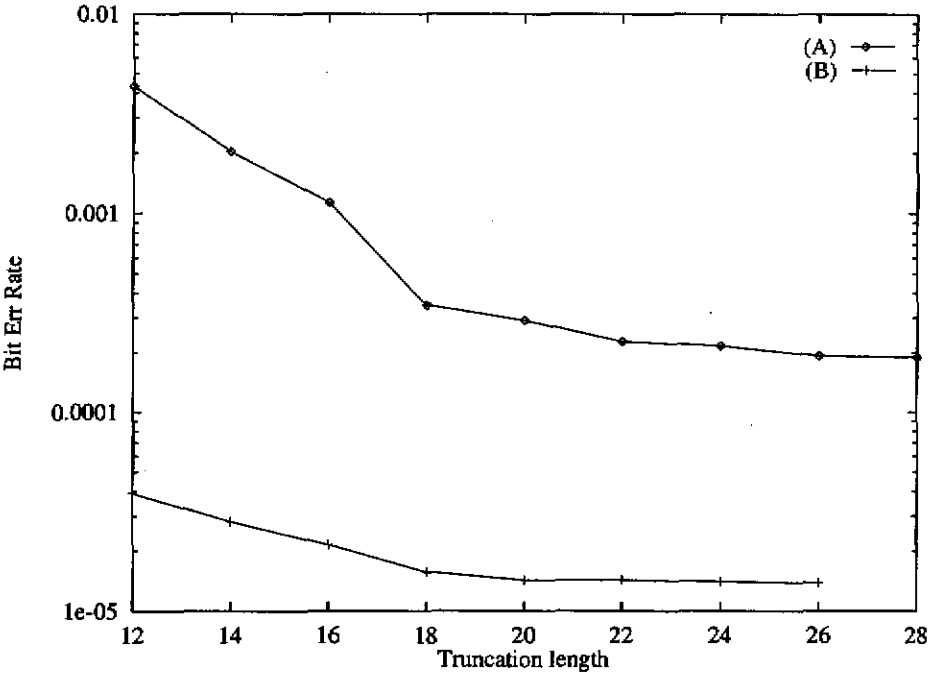


圖 2.6　籬柵碼 T 在不同截斷長度下的模擬結果($m=2$)
(A) E_b/N_o = 3.5dB　(B) E_b/N_o = 4.0dB

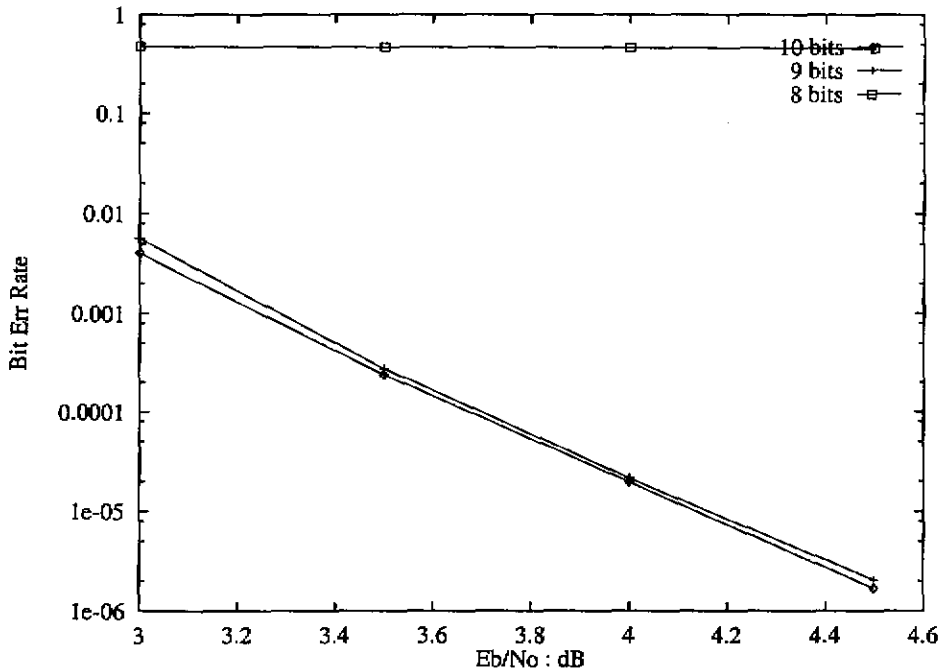


圖 2.7　籬柵碼 T 在不同記憶位元下的模擬結果（$m=2$，$\lambda=24$）

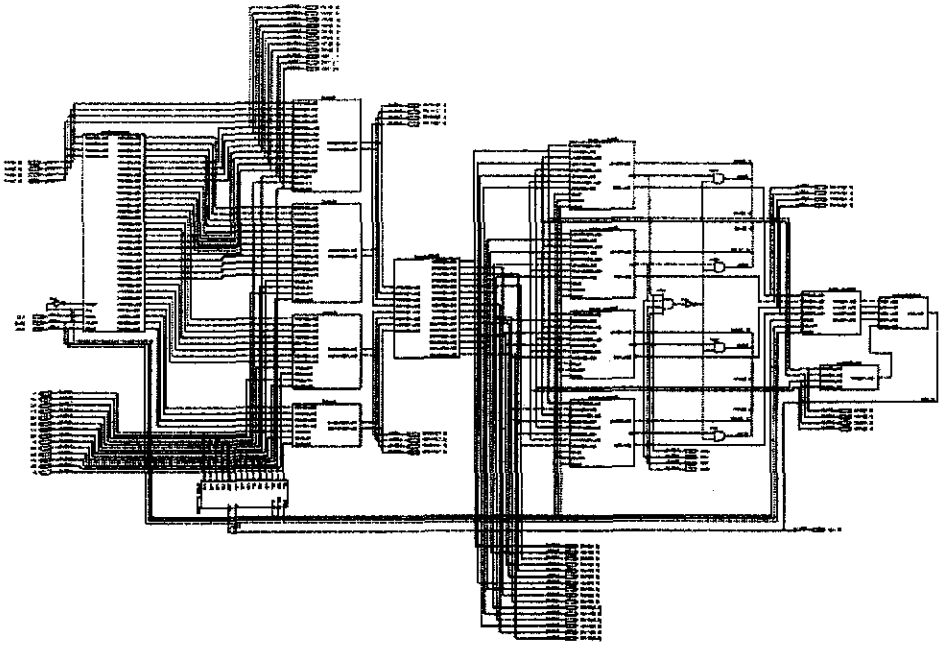


圖 2.8　4 個狀態點之籬柵碼 T 之次佳化解碼器的電路圖

圖 2.8 顯示了 4 個狀態點之籬柵碼 T 之次佳化解碼器的電路圖。在這裡我們顯示此解碼器之最上層電路，也就是如圖 2.1 及圖 2.2 所示之方塊圖。此圖是經由掃描器掃瞄後獲得，故解析度不是很高，但我們仍可以大約看出其與方塊圖之間的關係。最左邊的方塊為用來儲存已量化之資料，接下來的四個方塊分別與圖 2.2 之 $V_1 \sim V_4$ 相對應。中間之方塊是用來計算分之記量，而右邊的四個方塊分別與 $ACS_0 \sim ACS_3$ 相對應。最後最右邊之方塊為 SMU。

2.2 16 個狀態點之籬柵碼 T 之解碼器的電路設計

在這一小節裡，我們將描述 16 個狀態點之籬柵碼 T 之解碼器的電路設計。這個籬柵碼 T 由一個 $R = 2/4$，$m = 4$ 之傳統迴旋碼 C，後面接著一個迴旋處理器和一個信號對應器所組成。

2.2.1 籬柵碼 T 之編碼器的描述

圖 2.9 顯示了具 4 層迴旋處理器之 16 個狀態點的籬柵碼 T 的編碼器方塊圖，其中迴旋碼 C 之編碼器是一個 $R = 2/4$，$m = 4$ 之傳統迴旋碼。迴旋碼 C 將輸入訊息序列 $\bar{u} = \{\cdots, \hat{u}(0), \hat{u}(1), \cdots\}$ 轉換成序列 $\bar{v} = \{\cdots, \hat{v}(0), \hat{v}(1), \cdots\}$，其中 $\hat{u}(t) = \{u_1(t), u_2(t)\}$ 是一個 2 位元之符號且 $\bar{v} = \{v_1(t), \cdots, v_4(t)\}$ 是一個 4 位元之符號。接著序列 $\bar{v}$ 被送進迴旋處理器產生序列 $\bar{q} = \{\cdots, \hat{q}(0), \hat{q}(1), \cdots\}$，其中 $\hat{q}(t) = \{q_1(t), \cdots, q_4(t)\}$。$v_j(t)$ 和 $q_j(t)$ 關係亦如圖 2.3 所示。最後，序列 $\bar{q}$ 被送進信號對應器獲得一個輸出符號序列 $\bar{\sigma} = \{\cdots, \hat{\sigma}(0), \hat{\sigma}(1), \cdots\}$，其中 $\hat{\sigma}(t) = \{\sigma_1(t), \cdots, \sigma_4(t)\}$ 是在 $\{0, 1\}^4$。

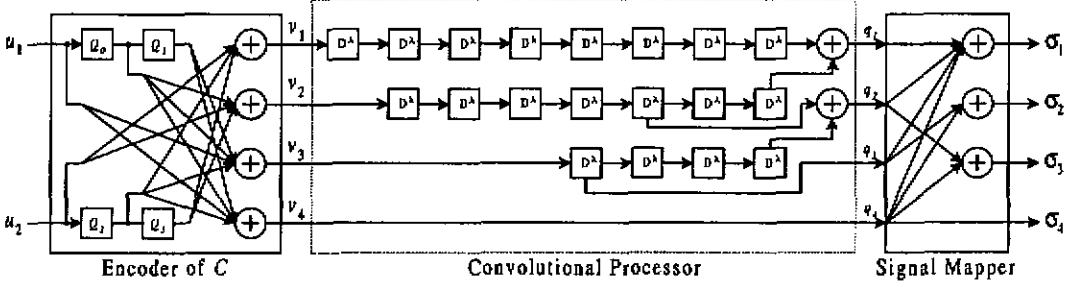


圖 2.9 16 個狀態點之籬柵碼 T 的編碼器方塊圖

2.2.2 籬柵碼 T 之解碼器的描述

圖 2.10 顯示了 16 個狀態點之籬柵碼 T 的次佳化解碼器方塊圖，其中迴旋碼 C 之編碼器是一個 $R=2/4$，$m=4$ 之傳統迴旋編碼器。這個 16 個狀態點之籬柵碼 T 的解碼器架構仍如前所述，由於在解碼程序之步驟一到步驟四中，我們要算出分別對應 v_1，v_2，v_3 和 v_4 之位元 0 與位元 1 的最小分支計量，因此需將這些已收到的信號，$\hat{r}_a$，$\hat{r}_b$，$\hat{r}_c$，$\hat{r}_d$，$\hat{r}_e$，$\hat{r}_f$，$\hat{r}_g$，$\hat{r}_h$ 和 $\hat{r}_i$，及已還原的信號，b_1^*，c_1^*，d_1^*，e_1^*，e_2^*，f_1^*，f_2^*，g_1^*，g_2^*，h_1^*，h_2^*，i_1^*，i_2^* 和 i_3^*，送進迴旋碼 C 之腓特比解碼器。其中，b_1^*，c_1^*，d_1^*，e_1^*，e_2^*，f_1^*，f_2^*，g_1^*，g_2^*，h_1^*，h_2^*，i_1^*，i_2^* 和 i_3^* 這些資料是經由將先前由迴旋碼 C_2 之腓特比解碼器所解碼出來的訊息位元 u_1^*，u_2^* 再次編碼並送入迴旋處理器後所獲得，而 $u_1^*=u_1^*(t-9\lambda)$，$u_2^*=u_2^*(t-9\lambda)$。

經由模擬結果顯示，如圖 2.11，使用量化層級 $Q=16$ 之解碼錯誤率相當接近 Q=∞之解碼錯誤率。其中圖 2.11 裡之曲線（D）是傳統之迴旋碼之位元錯誤率，其 $R=1/2$，$m=8$，且生成矩陣之八進制表示式為 $G=(561,753)_8$。我們可以看到使用量化層級 $Q=16$ 之迴旋碼 T 在 $E_b/N_o=3.5$dB 以上其位元錯誤率比傳統之迴旋碼要好，所以為了保有較優越之錯誤性能，我們採用量化層級 $Q=16$。

另外，經由軟體模擬，如圖 2.12，我們發現在 Q = 16 的條件下選擇常數 λ 大於 40 時在降低錯誤率方面只能獲得些微的好處。

對於 Q = 16，$2r$ 這個量之最大值為 $2\times15=30$，因此我們需要用 5 個位元來表示 15 或 $2r$。也就是說，每一個計量 $M(r_i(t),\sigma_i)$ 被表示成 5 個位元。而層級計量是將 4 個子層級計量 $M(r_i(t),\sigma_i)$ 加總起來，也就是最大值為 $30\times4=120$，因此我們需要用 7 個位元來表示之。由於 v_1，v_2，v_3 和 v_4 之位元計量分別包含 3，3，2，和 1 個層級計量，所以位元計量中之最

大值為 $120\times3=360$，也就是我們需要用 9 個位元來表示 1 個位元計量。在步驟 5 中，每一種可能之分支計量是將 4 個位元計量加總起來。由於分支計量之最大值為 $120\times(3+3+2+1)=1080$，因此我們需要用 11 個位元來表示一個可能之分支計量。經由軟體模擬結果，如圖 3.23，使用 11 個位元來儲存存活路徑之路徑計量是相當足夠的。當然，這個最好的路徑計量必須被解碼器儲存為狀態計量。

綜合上述，對於 $R=2/4$、$m=4$ 之具有迴旋處理器的二元迴旋碼之電路設計，我們列出規格如下：

- 狀態點數：16 個，
- 量化層級的數目：$Q=16$，
- 截斷長度（及延遲常數）：$\lambda=40$，
- 碼率：$R=2/4$，
- 生成矩陣：$G_C=\begin{pmatrix}0 & 2 & 7 & 7\\ 7 & 5 & 7 & 2\end{pmatrix}$
- 信號對應器：$W=\begin{pmatrix}0 & 0 & 0 & 1\\ 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 1\\ 1 & 1 & 1 & 1\end{pmatrix}$

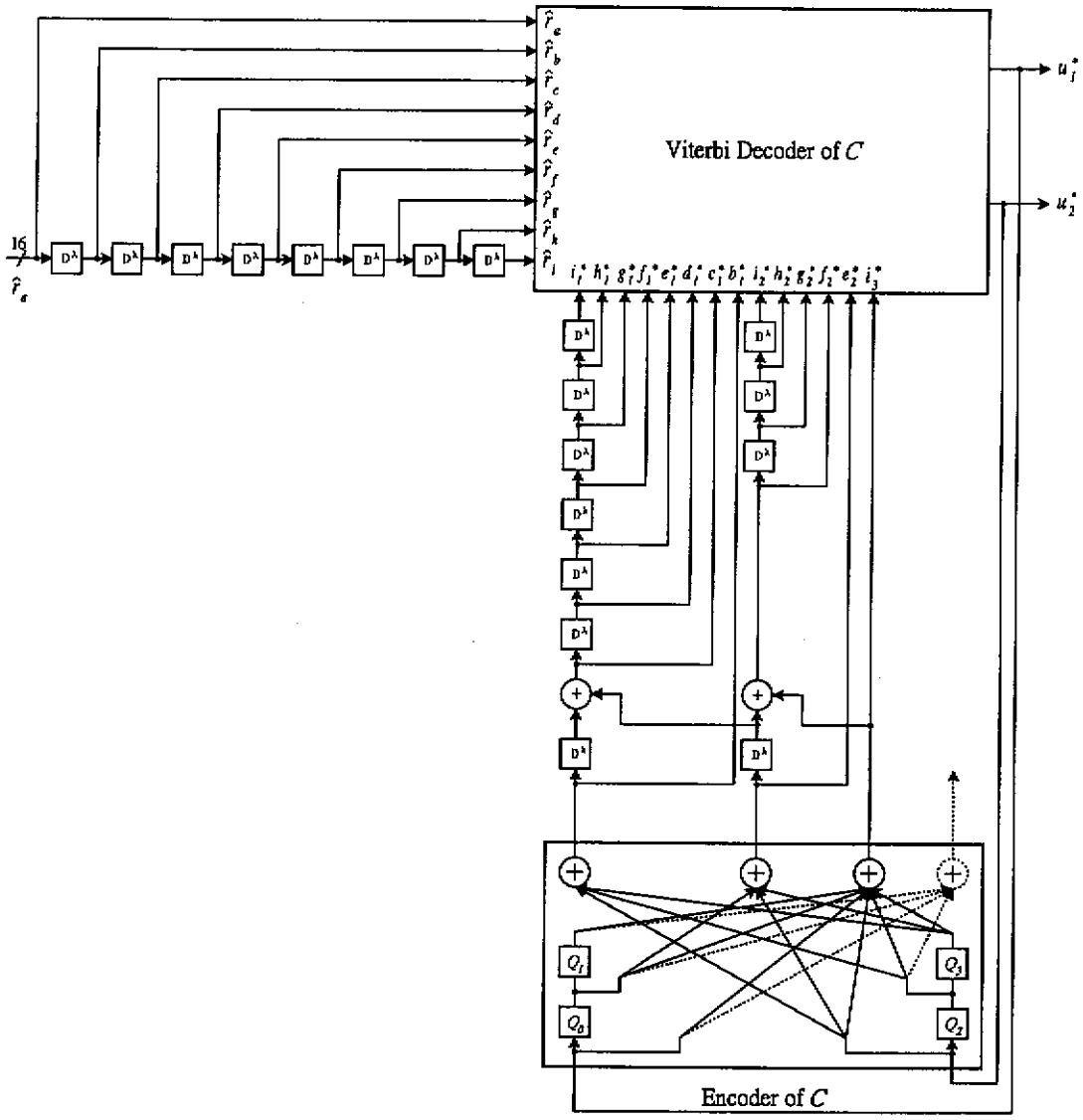


圖 2.10　16 個狀態點之籬柵碼 T 的解碼器方塊圖

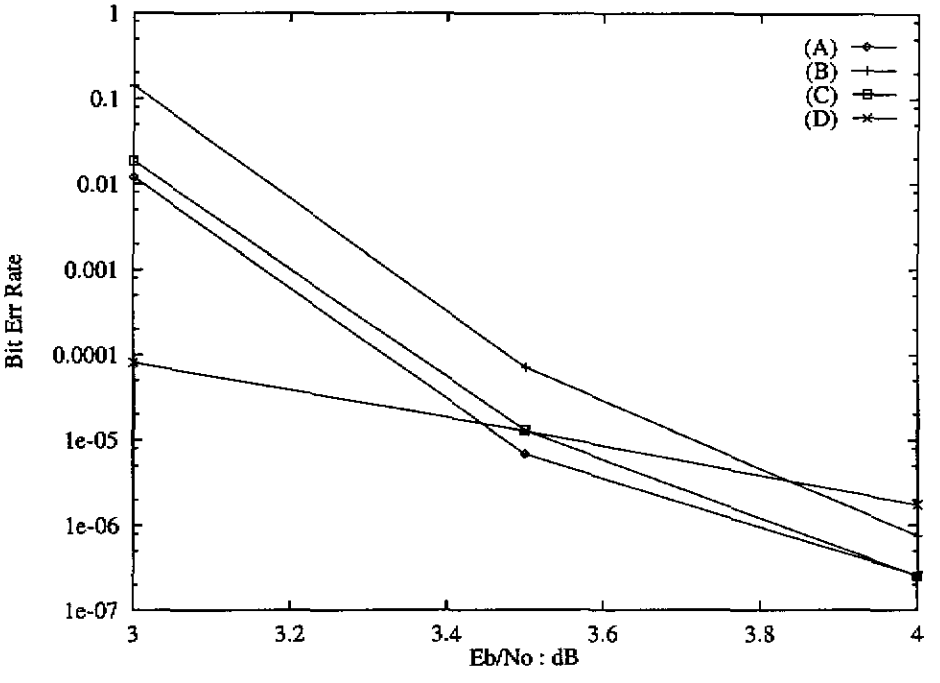


圖 2.11 籬柵碼 T 在不同之量化層級下的模擬結果($m=4$ ， $\lambda=40$)
(A) Q = ∞ ，(B) Q = 8，(C) Q = 16，
(D) 具 256 個狀態點之傳統的迴旋碼

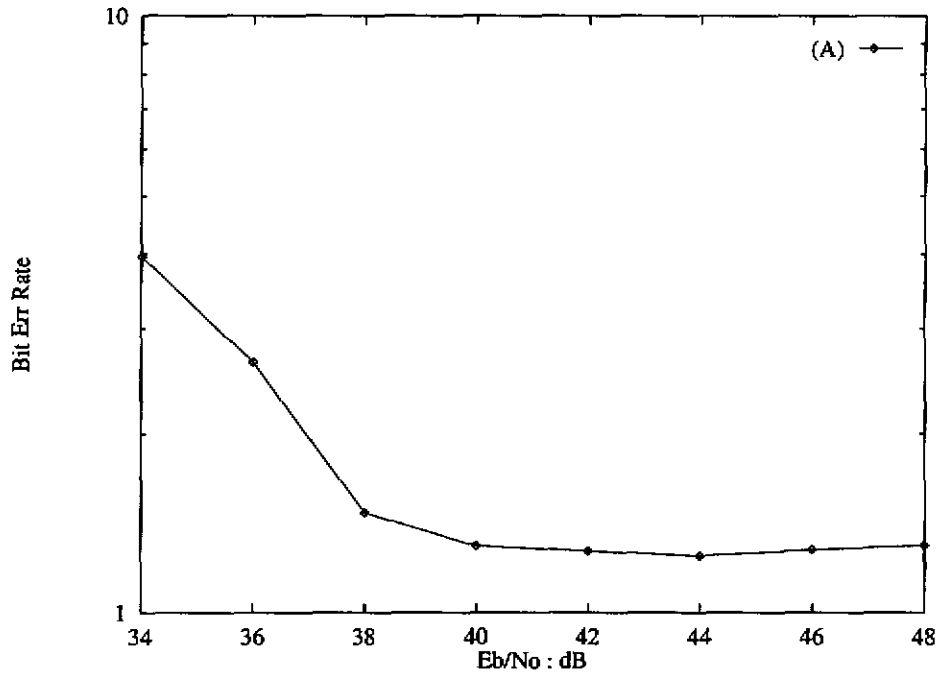


圖 2.12　籬柵碼 T 在不同截斷長度下的模擬結果($m=4$)

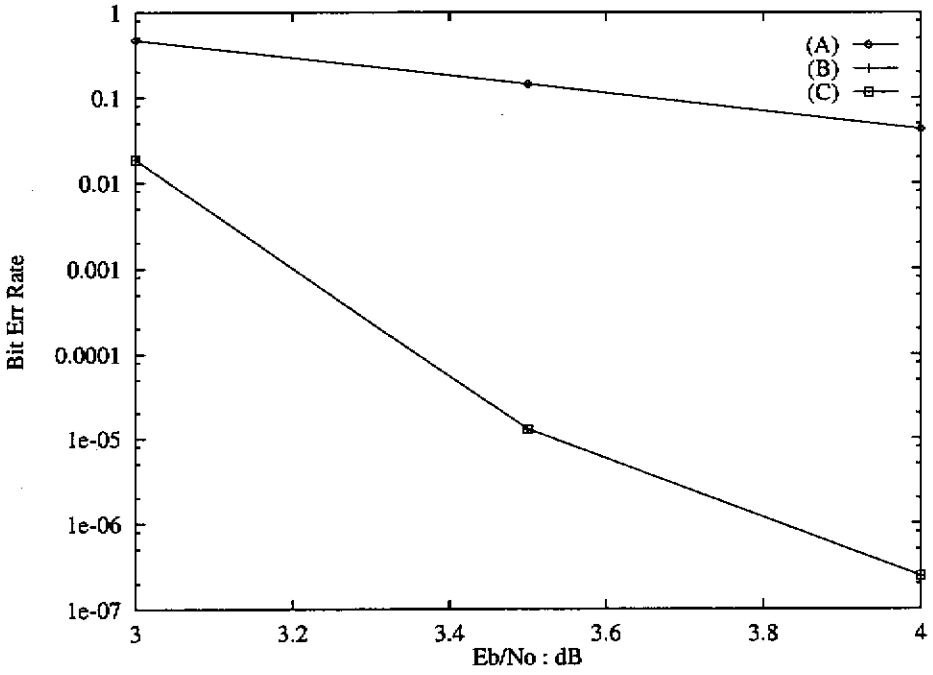


圖 2.13 籬柵 T 碼在不同之記憶位元下之模擬結果（$m=4$，$\lambda=40$）

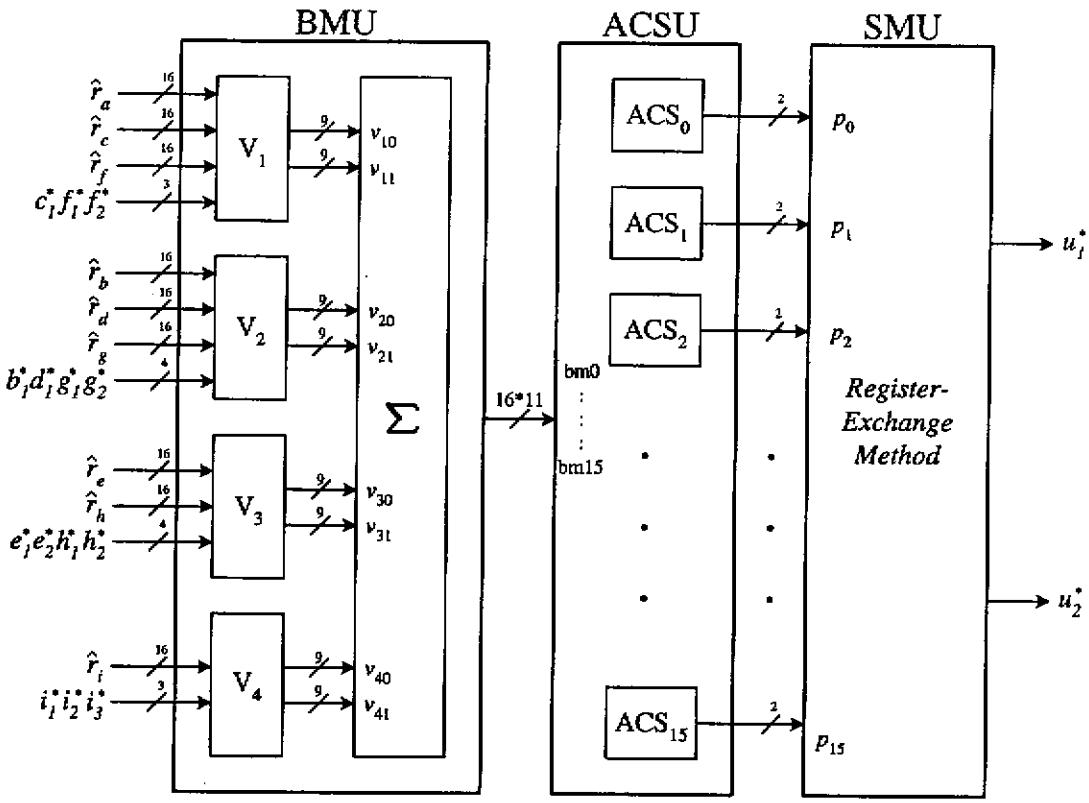


圖 2.14　迴旋碼 C 之腓特比解碼器

圖 2.24 顯示了迴旋碼 C 之腓特比解碼器的方塊圖。由於 $m=4$，也就是狀態點數為 $2^m=16$，因此若採用併聯處理器的架構，則我們需要 16 個 ACS 單元來求得存活路徑計量 sm_0，…，sm_{15} 以及路徑資料 p_0，…，p_{15}。

第三節

具迴旋處理器之籬柵碼之 SOVA 解碼器的電路設計

在此節，我們將使用軟式回授方式，來降低籬柵碼 T 的解碼器的錯誤機率。其回授方式，我們將採用兩個步驟之 SOVA 演算法。採用 SOVA 演算法，雖可降低解碼器之錯誤機率，但其也會增加額外複雜的電路，其性能優越，仍依賴良好的電路設計來驗證。

3.1 16 個狀態點之籬柵碼 T 之解碼器的電路設計

藉由使用迴旋碼 C 之解碼籬柵和一些迴授資訊，這個籬柵碼 T 可以被次佳化地解碼。因此，這個籬柵碼 T 之解碼器與傳統之腓特比解碼器的差異在於分支計量的計算需要作些修正且需加上額外之電路來產生迴授資訊。在這一小節裡，我們將詳細地描述 16 個狀態點之籬柵碼 T 之解碼器的電路設計，其中籬柵碼 T 由一個 $R=2/4$，$m=4$ 之傳統迴旋碼 C，後面接著一個迴旋處理器和一個信號對應器所組成。且位元資料回授方式我們將改為軟式迴授方式，其回授值的求法，我們採用 SOVA 解碼器獲得軟式輸出資訊。

3.1.1 16 個狀態點之籬柵碼 T 之解碼器的描述

圖 3.1 顯示了採用軟式回授方式之具迴旋處理器和 16 個狀態點之籬柵碼 T 的次佳化解碼器之方塊圖，其中迴旋碼 C 之編碼器是一個 $R=2/4$，$m=4$ 之迴旋編碼器。假設現在的時間點是 $t+8\lambda$。$\hat{r}_a$，$\hat{r}_b$，$\hat{r}_c$，$\hat{r}_d$，$\hat{r}_e$，$\hat{r}_f$，$\hat{r}_g$，$\hat{r}_h$ 和 $\hat{r}_i$ 分別表示 $\hat{r}(t+8\lambda)$，$\hat{r}(t+7\lambda)$，$\hat{r}(t+6\lambda)$，$\hat{r}(t+5\lambda)$，$\hat{r}(t+4\lambda)$，$\hat{r}(t+3\lambda)$，$\hat{r}(t+2\lambda)$，$\hat{r}(t+1\lambda)$ 及 $\hat{r}(t)$。另外，Lb_1，Lc_1，Ld_1，Le_1，Le_2，Lf_1，Lf_2，Lg_1，Lg_2，Lh_1，Lh_2，Li_1，Li_2 和 Li_3 分別表示之前時間點已解出的位元的軟式輸出值，$L\{v_1^*(t-1\lambda)\}$，$L\{v_1^*(t-2\lambda) \oplus v_2^*(t-1\lambda)\}$，$L\{v_1^*(t-3\lambda) \oplus v_2^*(t-2\lambda)\}$，$L\{v_1^*(t-4\lambda) \oplus v_2^*(t-3\lambda)\}$，$L\{v_2^*(t-1\lambda)\}$，$L\{v_1^*(t-5\lambda) \oplus v_2^*(t-4\lambda)\}$，$L\{v_2^*(t-2\lambda) \oplus v_3^*(t-1\lambda)\}$，$L\{v_1^*(t-6\lambda) \oplus v_2^*(t-5\lambda)\}$，$L\{v_2^*(t-3\lambda) \oplus v_3^*(t-2\lambda)\}$，$L\{v_1^*(t-7\lambda) \oplus v_2^*(t-6\lambda)\}$，$L\{v_2^*(t-4\lambda) \oplus v_3^*(t-3\lambda)\}$，$L\{v_1^*(t-8\lambda) \oplus v_2^*(t-7\lambda)\}$，$L\{v_2^*(t-5\lambda) \oplus v_3^*(t-4\lambda)\}$ 和 $L\{v_3^*(t-1\lambda)\}$。由第一部份可知在解碼程序之步驟一到步驟四中，我們要算出分別對應 v_1，v_2，v_3 和 v_4 之位元 0 與位元 1 的最大位元計量，因此需將這些已收到的信號，$\hat{r}_a$，$\hat{r}_b$，$\hat{r}_c$，$\hat{r}_d$，$\hat{r}_e$，$\hat{r}_f$，$\hat{r}_g$，$\hat{r}_h$ 和 $\hat{r}_i$，及得到的軟式回授值，Lb_1，Lc_1，Ld_1，Le_1，Le_2，Lf_1，Lf_2，Lg_1，Lg_2，

Lh_1，Lh_2，Li_1，Li_2和Li_3，送進迴旋碼 C 之腓特比解碼器。其中，Lb_1，Lc_1，Ld_1，Le_1，Le_2，Lf_1，Lf_2，Lg_1，Lg_2，Lh_1，Lh_2，Li_1，Li_2和Li_3這些資料是經由先前迴旋碼 C 之腓特比解碼器所解碼出來的訊息位元 u_1^*，u_2^*再次編碼後並送入 SOVA 解碼器後所獲得軟式輸出值 $L(v_1^*(t-1\lambda-D))$，$L(v_2^*(t-1\lambda-D))$和$L(v_3^*(t-1\lambda-D))$，再經過迴旋處理器所獲得的軟式資料。經過 SOVA 解碼器後即得解出位元$u_1'=u_1^*(t-1\lambda-D)$，$u_2'=u_2^*(t-1\lambda-D)$，其中 D 為 SOVA 解碼器中，向後追縱的長度，$D\approx\frac{1}{2}\lambda$。

從圖 3.1 我們可以看到，籬柵碼 T 之解碼器的設計重點在於迴旋碼 C 之腓特比解碼器與軟式輸出解碼器(SOVA)。圖 3.2 顯示了這個迴旋碼 C 之腓特比解碼器的架構，它包含了三個部分：BMU，ACSU 和 SMU。在 BMU 這個部分裡，其包含了用來計算位元計量的四個小單元，分別是V_1，V_2，V_3和V_4，以及用來計算分支計量的單元，$\sum$。V_1，V_2，V_3和V_4四個小單元是用來計算分別對應迴旋碼 C 之編碼器所輸出之四個位元的位元計量 M_1，M_2，M_3和 M_4。我們將現在所收到的信號$\hat{r}_a$和先前所收到的信號$\hat{r}_b$，$\hat{r}_c$，$\hat{r}_d$，$\hat{r}_e$，$\hat{r}_f$，$\hat{r}_g$，$\hat{r}_h$和$\hat{r}_i$，以及先前所得到信號的軟式資料Lb_1，Lc_1，Ld_1，Le_1，Le_2，Lf_1，Lf_2，Lg_1，Lg_2，Lh_1，Lh_2，Li_1，Li_2和Li_3一併送進 BMU 來產生 M_{10}，M_{11}，M_{20}，M_{21}，M_{30}，M_{31}，M_{40}和 M_{41}，也就是 M_1 對應於$v_1(t)$等於 0 和 1 的位元計量，M_2 對應於$v_2(t)$等於 0 和 1 的位元計量，M_3 對應於$v_3(t)$等於 0 和 1 的位元計量，M_4 對應於$v_4(t)$等於 0 和 1 的位元計量。透過由加法器所組成之小單元 $\sum$，我們可以獲得在解碼步驟五裡所需要之 16 種可能的分支計量。將這些 16 種可能的分支計量表示成bm_0，…，bm_{15}，並且送進 ACSU 以獲得 16 個狀態點計量及相對應之存活路徑，p_0，$\cdots p_{14}$和p_{15}。這 16 個存活路徑被送進 SMU 並且儲存起來以便作為還原訊息位元的依據。最後，SMU 產生出我們想要的訊息位元，u_1^*，u_2^*。由於籬柵碼 T 的解碼籬柵只有 16 個狀態點，我們可以在 ACSU 裡採用併聯處理的方式以提升解碼速率。事實上，我們的解碼器也僅需要 16 個 ACS 處理器。

對於圖 3.2 之解碼器的電路設計，我們將在稍後作詳細的描述，而 SOVA 解碼器之電路設計，我們也將在稍後章節作詳細的描述。

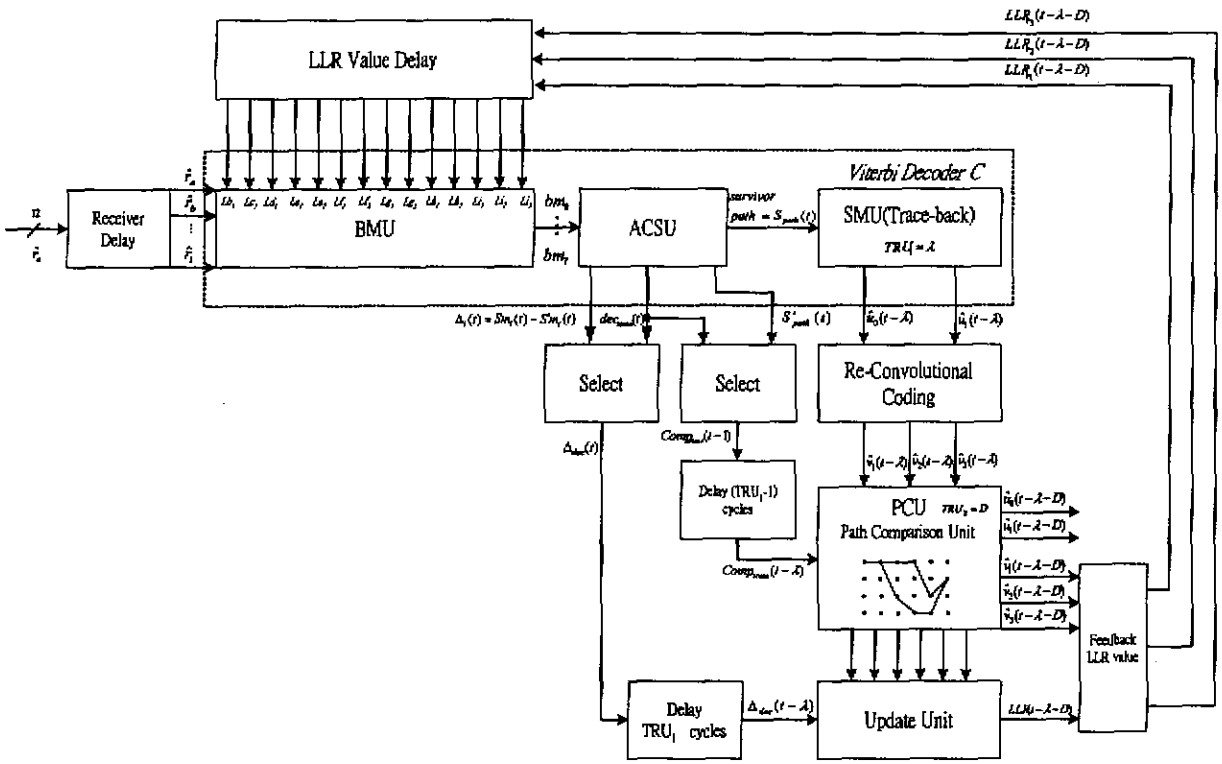


圖 3.1　16 個狀態點之採用軟式回授方式之蘿柵碼 T 的解碼器方塊圖

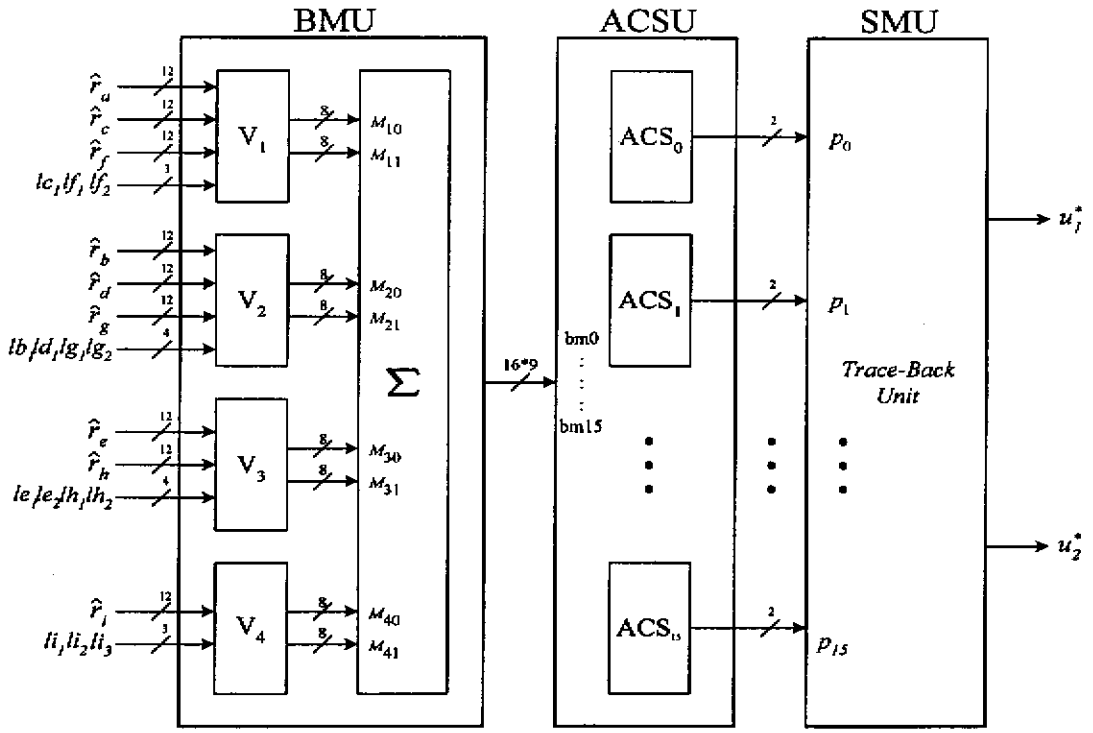


圖 3.2 迴旋碼 C 之腓特比解碼器

3.1.2 量化

圖 4.3 (a)、(b)、(c)和(d)顯示了四種量化方案，分別是 2 個層級、4 個層級、8 個層級和 16 個層級，而間距分別是 2.0、1.0、0.5 和 0.25。

在腓特比解碼器裡，由於符號計量 (symbol metric)必須被表示成數位型態，於是計量量化之效應便衍生出一個問題。模擬結果顯示了解碼器的性能對於符號計量量化相當地不敏感。事實上，相對於使用對數可能性 (log likelihood)而言，使用整數作為符號計量在 2 個，4 個，8 個層級之量化下僅造成可忽略的性能衰減[6]。

考慮 16 個層級之量化，在發射端將籬柵碼 T 之輸出經過 BPSK

(Binary Phase Shift Keying)調變後送進AWGN（Additive White Gaussian Noise）通道。在接收端，假設受雜訊干擾之收到的位元信號$r_i(t)$被量化在第r個層級，其中$r=0,1,\cdots,15$。根據圖3.3(d)16層級之量化圖可知，接收的層級 r 對於位元 1 和 0 位元之間的距離為 $d\{(r\text{-}11.5)^2,(r\text{-}3.5)^2\}=d\{120,16r\}=d\{15,2r\}$，為了化簡電路設計，我們將位元計量作定量平移，於是可以得到位元0和1的位元計量分別是$2r$和15。通常，對於任意2的次方之量化層級Q而言，位元1和0之位元計量分別是Q-1和$2r$。

經由模擬結果顯示，如圖3.4，使用量化層級$Q=16$之解碼錯誤率相對於Q $=\infty$之解碼錯誤率僅僅損失不到0.25dB。其中圖3.4裡CONV之曲線是具有迴旋處理器但尚未加上 SOVA 演算法之籬柵碼。我們可以看到使用量化層級$Q=16$之籬柵碼T在$E_b/N_o=3.5$dB以上其位元錯誤率比未加上 SOVA 演算法之籬柵碼還要好，所以為了減少計算量及存儲器之用量，我們採用量化層級$Q=16$。

另外，經由軟體模擬，如圖3.5，我們發現在$Q=16$的條件下選擇常數λ大於40時在降低錯誤率方面只能獲得些微的好處。

對於Q = 16，$2r$這個量之最大值為$2\times15=30$，因此我們需要用5個位元來表示15或$2r$。但做每個位元計量計算時，還需加上軟式回授資料做增加的調整值，由3.1節可知所得的軟式回授值為$\Delta^m(t)=\frac{M^m(t)-M^n(t)}{2}$，其$\Delta^m(t)$的最大值經程式模擬得到大約為120。於是我們需要用7個位元來表示$\Delta^m(t)$的值。於是每一個計量$M(r_i(t),\sigma_i)$的最大值為15+120=135需用8個位元表示。而層級計量 (level metric)是將4個子層級 (sublevel) 計量$M(r_i(t),\sigma_i)$加總起來，也就是最大值為$135\times4=540$，因此我們需要用10個位元來表示之。由於v_1，v_2，v_3和v_4之位元計量分別包含3，3，2，和1個層級計量，所以位元計量中之最大值為$540\times3=1620$，也就是我們需要用11個位元來表示1個位元計量。在步驟5中，每一種可能之分

支計量是將 4 個位元計量加總起來。由於分支計量之最大值為 $540 \times (3 + 3 + 2 + 1) = 4860$，因此我們需要用 13 個位元來表示一個可能之分支計量。

綜合上述，對於 $R = 2/4$、$m = 4$ 之具有迴旋處理器的二元迴旋碼之電路設計，我們列出規格如下：

- 狀態點數：16 個，
- 量化層級的數目：$Q = 16$，
- 截斷長度（及延遲常數）：$\lambda = 40$，
- 碼率：$R = 2/4$，
- 生成矩陣：$G_C = \begin{pmatrix} 0 & 2 & 7 & 7 \\ 7 & 5 & 7 & 2 \end{pmatrix}$
- 信號對應器：$W = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$

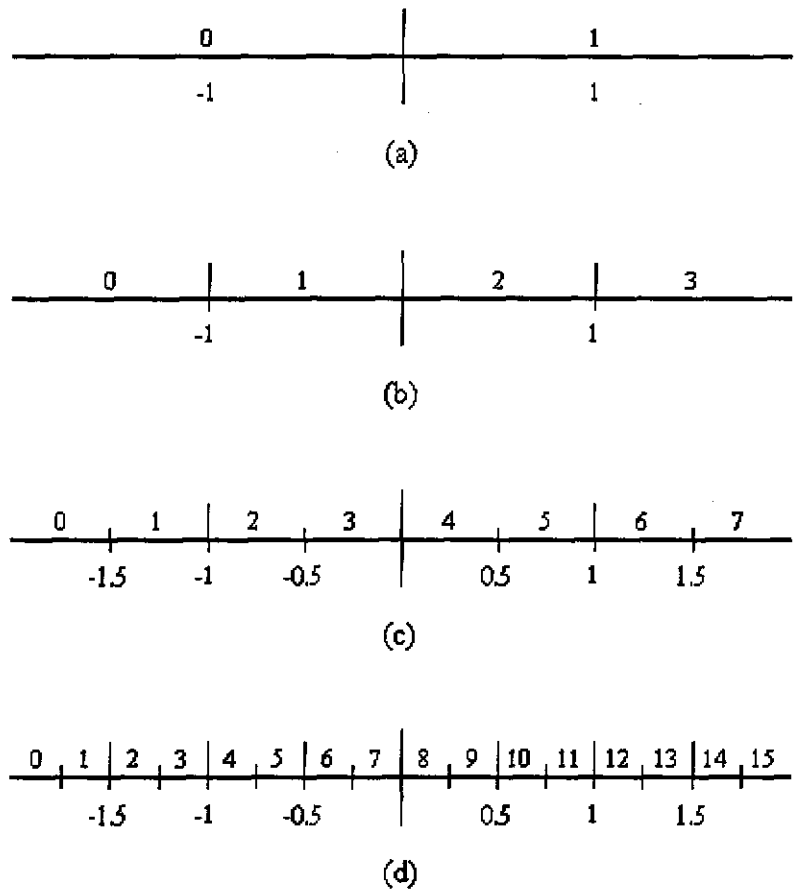


圖 3.3　量化臨界和區間：(a) 2 層之量化、(b) 4 層之量化、(c) 8 層之量化、(d) 16 層之量化

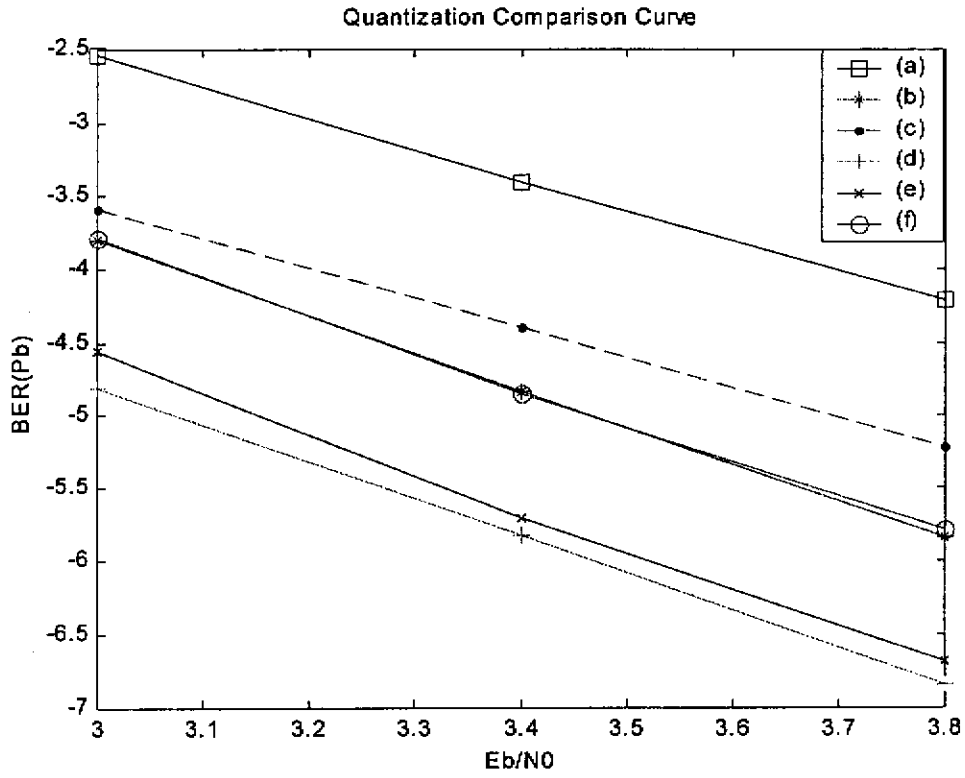


圖 3.4　籬柵碼 T 在不同之量化層級下的模擬結果

(a)採用硬式迴授值之 4 個狀態點之具有迴旋處理器的籬柵碼(b) 採用硬式迴授值之 16 個狀態點之具有迴旋處理器的籬柵碼(c) 採用 SOVA 演算法之 4 個狀態點之具有迴旋處理器的籬柵碼(d) 採用 SOVA 演算法之 16 個狀態點之具有迴旋處理器的籬柵碼(e) 採用 SOVA 演算法之 16 個狀態點之具有迴旋處理器的籬柵碼 16 階量化階層(f) 採用 SOVA 演算法之 16 個狀態點之具有迴旋處理器的籬柵碼 8 階量化階層

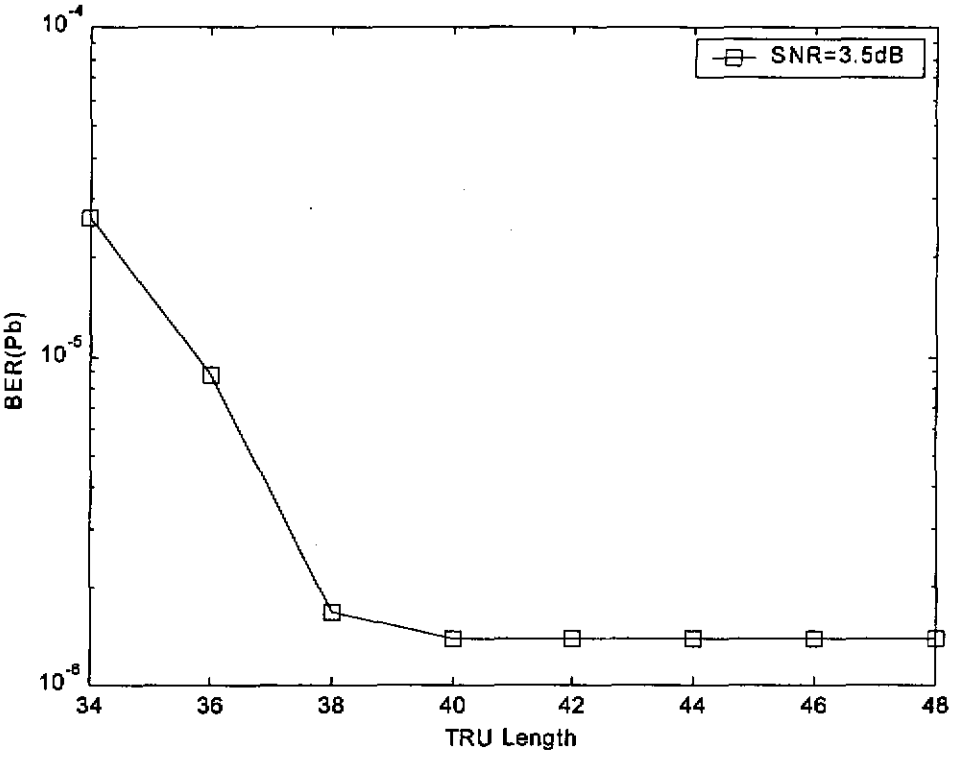


圖 3.5　籬柵碼 T 在不同截斷長度下的模擬結果($m=4$)
$E_b/N_o = 3.5\text{dB}$

3.2 迴旋碼 C 之腓特比解碼器的實現

如先前在圖 3.2 所示，迴旋碼 C 之腓特比解碼器的電路方塊圖由三個部分所組成：分支計量單元（Branch Metric Unit, BMU），加總-比較-選擇單元（Add-Compare-Select Unit, ACSU）和存活記憶單元（Survivor Memory Unit, SMU ）。在這一個小節裡，我們將詳盡地介紹這三個部分的功能。

3.2.1 分支計量 (BM) 單元

分支計量單元由 4 個子單元 V_j，$1 \le j \le 4$，和 1 個 $\sum$ 所組成。其中 V_j 計算出位元計量 M_j，$1 \le j \le 4$，而 $\sum$ 則將這四個位元計量 M_1，M_2，M_3 和 M_4 加總起來得到 16 種可能分支計量。對 M_1 而言，也就是 $v_1(t)$ 的位元計量，它同時考慮了三個接收到的信號 $\hat{r}(t+8\lambda)$，$\hat{r}(t+6\lambda)$ 和 $\hat{r}(t+3\lambda)$，亦即分別在三個時間點 $t+8\lambda$，$t+6\lambda$ 和 $t+3\lambda$ 上。首先，我們將焦點放在 $t+8\lambda$ 這個時間點上。由於 $v_2(t+1\lambda)$，$v_2(t+3\lambda)$，$v_3(t+4\lambda)$，$v_3(t+7\lambda)$ 和 $v_4(t+8\lambda)$ 皆尚未被解碼出來，且值 $v_1(t) \oplus v_2(t+1\lambda)$ 是在多層級編碼（multilevel coding）結構裡的第一子層級，因此我們定義計量 $-\left\|\hat{r}(t+8\lambda)-w(\hat{s}(t+8\lambda))\right\|^2$ 為第一子層級之計量。再來，我們將焦點放在 $t+6\lambda$ 這個時間點上。$L(v_1^*(t-2\lambda))$ 及 $L(v_2^*(t-1\lambda))$ 是被解碼出來位元的軟式回授值，但 $v_3(t+2\lambda)$，$v_3(t+5\lambda)$ 和 $v_4(t+6\lambda)$ 尚未被解碼出來，且值 $v_2(t+1\lambda) \oplus v_3(t+2\lambda)$ 位在多層級編碼結構裡的第二子層級，因此我們定義計量 $-\left\|\hat{r}(t+6\lambda)-w(\hat{s}(t+6\lambda))\right\|^2 + \frac{1}{2}(s_1(t+6\lambda)-1)\cdot L(\hat{s}_1(t+6\lambda))$ 為第二子層級之計量。最後，我們把焦點放在 $t+3\lambda$ 這個時間點上。其中 $L(v_1^*(t-5\lambda))$，$L(v_2^*(t-4\lambda))$，$L(v_2^*(t-2\lambda))$ 和 $L(v_3^*(t-1\lambda))$ 已是被解碼位元的軟式回授值，僅 $v_4^*(t+3\lambda)$ 尚未被解碼，而值 $v_3^*(t+2\lambda)$ 位在多層級編碼架構裡的第三子層級，所以我們定義計量 $-\left\|\hat{r}(t+3\lambda)-w(\hat{s}(t+3\lambda))\right\|^2 + \frac{1}{2}(s_1(t+3\lambda)-1)\cdot L(\hat{s}_1(t+3\lambda))$

$+\frac{1}{2}(s_2(t+3\lambda)-1)\cdot L(\hat{s}_2(t+3\lambda))$為第三子層級之計量。同理，對 M_2 而言，也就是 $v_2^*(t)$ 的位元計量，它亦同時考慮了三個時間點 $t+7\lambda$，$t+5\lambda$ 和 $t+2\lambda$。其中 $-\|\hat{r}(t+7\lambda)-w(\hat{s}(t+7\lambda))\|^2$ 屬於第一子層級之計量，$-\|\hat{r}(t+5\lambda)-w(\hat{s}(t+5\lambda))\|^2+\frac{1}{2}(s_1(t+5\lambda)-1)\cdot L(\hat{s}_1(t+5\lambda))$ 屬於第二子層級之計量，$-\|\hat{r}(t+2\lambda)-w(\hat{s}(t+2\lambda))\|^2+\frac{1}{2}(s_1(t+2\lambda)-1)\cdot L(\hat{s}_1(t+2\lambda))+\frac{1}{2}(s_2(t+2\lambda)-1)\cdot L(\hat{s}_2(t+2\lambda))$ 屬於第三子層級之計量。對 M_3 而言，亦即 $v_3^*(t)$ 的位元計量，它考慮了兩個時間點 $t+4\lambda$ 和 $t+1\lambda$。其中 $-\|\hat{r}(t+4\lambda)-w(\hat{s}(t+4\lambda))\|^2+\frac{1}{2}(s_1(t+4\lambda)-1)\cdot L(\hat{s}_1(t+4\lambda))$ 屬於第二子層級之計量，$-\|\hat{r}(t+1\lambda)-\hat{s}(t+1\lambda)\|^2+\frac{1}{2}(s_1(t+11\lambda)-1)\cdot L(\hat{s}_1(t+1\lambda))+\frac{1}{2}(s_2(t+1\lambda)-1)\cdot L(\hat{s}_2(t+1\lambda))$ 屬於第三子層級之計量。最後，對 M_4 而言，他只考慮一個時間點 t。由於 $L(v_1^*(t-8\lambda))$，$L(v_2^*(t-7\lambda))$，$L(v_2^*(t-5\lambda))$，$L(v_3^*(t-4\lambda))$ 和 $L(v_3^*(t-1\lambda))$ 是已解碼出來位元的軟式回授值，而值 $v_4^*(t)$ 位在多層級編碼架構裡的第四子層級，即圖 2.6 裡的 s_4，因此我們定義計量 $-\|\hat{r}(t)-w(\hat{s}(t))\|^2+\frac{1}{2}(s_1(t)-1)\cdot L(\hat{s}_1(t))+\frac{1}{2}(s_2(t)-1)\cdot L(\hat{s}_2(t))+\frac{1}{2}(s_3(t)-1)\cdot L(\hat{s}_3(t))$ 為第四子層級之計量。

綜合上述，我們可以歸納出一個簡單且低複雜的腓特比解碼器電路設計之方法，也就是分別設計出這些具有低複雜特性之計算四個子層級計量單元，然後將其分別整合至子單元 V_1，V_2，V_3 和 V_4 中並做最佳化處理。如此，我們便可獲得一個具有低複雜特性之迴旋碼 C 的腓特比解碼器，也就保證了籬柵碼 T 之解碼器的性能。

● 第一子層級之計量單元：

經由仔細地檢查圖 2.4 裡的信號對應器，其中 $\oplus$ 表示以 2 為模之加法，我們發現在 $\sigma_1\oplus\sigma_2\oplus\sigma_3\oplus\sigma_4=p$ 的條件下，$p\in\{0,1\}$，$M_{1\alpha}$ 即是 $\sum_{i=1}^{4}M(r_i(t),\sigma_i)$ 之最大值，其中 $\alpha=P$。在這個觀察之下，我們發現 l_{10} 和 l_{11} 可以很容易地藉由將腓特比解碼應用到圖 3.6(a)的籬柵圖而被計算成存

活者之最後計量。圖 3.6(b)顯示了使用這個籬柵圖計算出 l_{10} 和 l_{11} 之方塊圖，其中 ADD，COM 和 MUX 分別表示加法器，比較器和多工器。

● 第二子層級之計量單元：

對於計量 $-\|\hat{r}(t+6\lambda)-w(\hat{s}(t+6\lambda))\|^2+\frac{1}{2}(s_1(t+6\lambda)-1)\cdot L(\hat{s}_1(t+6\lambda))$ 而言，它依據先前所還原之資料的軟式回授值 $Lc_1=L(v_1^*(t-10\lambda)\oplus v_2^*(t-9\lambda))$。而對於計量 $-\|\hat{r}(t+5\lambda)-w(\hat{s}(t+5\lambda))\|^2+\frac{1}{2}(s_1(t+5\lambda)-1)\cdot L(\hat{s}_1(t+5\lambda))$ 而言，它依據先前所還原之資料的軟式回授值 $Ld_1=L(v_1^*(t-3\lambda)\oplus v_2^*(t-2\lambda))$。另外，對於計量 $-\|\hat{r}(t+4\lambda)-w(\hat{s}(t+4\lambda))\|^2+\frac{1}{2}(s_1(t+4\lambda)-1)\cdot L(\hat{s}_1(t+4\lambda))$ 而言，它依據先前所還原之資料的軟式回授值 $Le_1=L(v_1^*(t-4\lambda)\oplus v_2^*(t-3\lambda))$ 及 $Le_2=L(v_2^*(t-1\lambda))$，其中 $v_2(t-1\lambda)$ 與未被解碼之 $v_3^*(t)$ 共同決定在第二子層級之值，也就是決定了此子層級分別對應位元為 0 和 1 之計量。第二子層級之計量的獲得是根據第一子層級位元 s_1，假設我們已經得到被還原之位元 $\hat{s}_1$ 的軟式回授值為 $L(\hat{s}_1)$。假設 $SUM_{l2}=\sum_{i=1}^{4}M(r_i(t),\sigma_i)+\frac{1}{2}(2s_1-1)\cdot L(\hat{s}_1)$，$s_1\in\{0,1\}$ 我們看到如果 $s_1=0$，那麼在 $s_2=\sigma_3\oplus\sigma_4=0$ 的條件下，對於 $(\sigma_1,\sigma_2,\sigma_3,\sigma_4)\in\{(0000),(0011),(1100),(1111)\}$ 而言，l_{20} 即是 $SUM_{l2}=\sum_{i=1}^{4}M(r_i(t),\sigma_i)-\frac{1}{2}\cdot L(\hat{s}_1)$ 的最大值，且在 $s_2=\sigma_3\oplus\sigma_4=1$ 的條件下，對於 $(\sigma_1,\sigma_2,\sigma_3,\sigma_4)\in\{(0110),(0101),(1010),(1001)\}$ 而言，l_{21} 即是 $SUM_{l2}=\sum_{i=1}^{4}M(r_i(t),\sigma_i)+\frac{1}{2}\cdot L(\hat{s}_1)$ 的最大值。另外，如果 $s_1=1$，那麼在 $s_2=\sigma_3\oplus\sigma_4=0$ 的條件下，對於 $(\sigma_1,\sigma_2,\sigma_3,\sigma_4)\in\{(0001),(0010),(1101),(1110)\}$ 而言，l_{20} 即是 $SUM_{l2}=\sum_{i=1}^{4}M(r_i(t),\sigma_i)-\frac{1}{2}\cdot L(\hat{s}_1)$ 的最大值，且在 $s_2=\sigma_3\oplus\sigma_4=1$ 的條件下，對於 $(\sigma_1,\sigma_2,\sigma_3,\sigma_4)\in\{(0111),(0100),(1011),(1000)\}$ 而言，l_{21} 即是 $SUM_{l2}=\sum_{i=1}^{4}M(r_i(t),\sigma_i)+\frac{1}{2}\cdot L(\hat{s}_1)$ 的最大值。因此，l_{20} 和 l_{21} 可以很容易地藉由將腓特比解碼應用到圖 3.7(a)的籬柵圖而被計算成存活者之最後計量，其中圖 3.7(a)裡籬柵的最後一個分支上的符號依據值 s_1 而定。圖 3.7(b)

顯示了使用這個籬柵圖計算出 l_{20} 和 l_{21} 之方塊圖。

● **第三子層級之計量單元：**

對計量 - $\|\hat{r}(t+3\lambda)-w(\hat{s}(t+3\lambda)\|^2 + \frac{1}{2}(s_1(t+3\lambda)-1)\cdot L(\hat{s}_1(t+3\lambda)) + \frac{1}{2}(s_2(t+3\lambda)-1)\cdot L(\hat{s}_2(t+3\lambda))$ 而言，它依據先前所還原之資料的軟式回授值 $Lf_1 = L(v_1^*(t-5\lambda) \oplus v_2^*(t-4\lambda))$ 以及 $Lf_2 = L(v_2^*(t-2\lambda) \oplus v_3^*(t-1\lambda))$。對於計量 - $\|\hat{r}(t+2\lambda)-w(\hat{s}(t+2\lambda)\|^2 + \frac{1}{2}(s_1(t+2\lambda)-1)\cdot L(\hat{s}_1(t+2\lambda)) + \frac{1}{2}(s_2(t+2\lambda)-1)\cdot L(\hat{s}_2(t+2\lambda))$ 而言，它依據先前所還原之資料的軟式回授值 $Lg_1 = L(v_1^*(t-6\lambda) \oplus v_2^*(t-5\lambda))$ 以及 $Lg_2 = L(v_2^*(t-3\lambda) \oplus v_3^*(t-2\lambda))$。另外，對於的計量 - $\|\hat{r}(t+1\lambda)-\hat{s}(t+1\lambda)\|^2 + \frac{1}{2}(s_1(t+1\lambda)-1)\cdot L(\hat{s}_1(t+1\lambda)) + \frac{1}{2}(s_2(t+1\lambda)-1)\cdot L(\hat{s}_2(t+1\lambda))$ 而言，它依據先前所還原之資料的軟式回授值 $Lh_1 = L(v_1^*(t-7\lambda) \oplus v_2^*(t-6\lambda))$ 以及 $Lh_2 = L(v_2^*(t-4\lambda) \oplus v_3^*(t-3\lambda))$。也就是說，第三子層級之計量的獲得是根據第一子層級位元 s_1 以及第二子層級位元 s_2，假設我們已經得到被還原之位元 $\hat{s}_1$ 和 $\hat{s}_2$ 的軟式回授值分別為 $L(\hat{s}_1)$ 和 $L(\hat{s}_2)$。假設 $SUM_{13} \equiv \sum_{i=1}^{4} M(r_i(t),\sigma_i) + \frac{1}{2}(2s_1-1)\cdot L(\hat{s}_1) + \frac{1}{2}(2s_2-1)\cdot L(\hat{s}_2)$。$s_1, s_2 \in \{0,1\}$，我們看到如果 $s_1=0$ 和 $s_2=0$，那麼在 $s_3 = \sigma_2 \oplus \sigma_4 = 0$ 的條件下，對於 $(\sigma_1\sigma_2\sigma_3\sigma_4) \in \{(0000), (1111)\}$ 而言，l_{30} 即是 $SUM_{13} \equiv \sum_{i=1}^{4} M(r_i(t),\sigma_i) - \frac{1}{2}\cdot L(\hat{s}_1) - \frac{1}{2}\cdot L(\hat{s}_2)$ 之最大值，且在 $s_3 = \sigma_2 \oplus \sigma_4 = 1$ 的條件下，對於 $(\sigma_1\sigma_2\sigma_3\sigma_4) \in \{(0011), (1100)\}$ 而言，l_{31} 即是 $\sum_{i=1}^{4} M(r_i(t),\sigma_i) - \frac{1}{2}\cdot L(\hat{s}_1) - \frac{1}{2}\cdot L(\hat{s}_2)$ 之最大值。如果 $s_1=0$ 和 $s_2=1$，那麼在 $s_3=0$ 的條件下，對於 $(\sigma_1\sigma_2\sigma_3\sigma_4) \in \{(0101), (1010)\}$ 而言，l_{30} 即是 $SUM_{13} \equiv \sum_{i=1}^{4} M(r_i(t),\sigma_i) - \frac{1}{2}\cdot L(\hat{s}_1) + \frac{1}{2}\cdot L(\hat{s}_2)$ 之最大值，且在 $s_3=1$ 的條件下，對於 $(\sigma_1\sigma_2\sigma_3\sigma_4) \in \{(0110), (1001)\}$ 而言，l_{31} 即是 $SUM_{13} \equiv \sum_{i=1}^{4} M(r_i(t),\sigma_i) - \frac{1}{2}\cdot L(\hat{s}_1)) + \frac{1}{2}\cdot L(\hat{s}_2)$ 之最大值。另外，如果 $s_1=1$ 和 $s_2=0$，那麼在 $s_3=0$ 的條件下，對於 $(\sigma_1\sigma_2\sigma_3\sigma_4) \in \{(0001), (1110)\}$ 而言，l_{30} 即是 $SUM_{13} \equiv \sum_{i=1}^{4} M(r_i(t),\sigma_i) + \frac{1}{2}\cdot L(\hat{s}_1) - \frac{1}{2}\cdot L(\hat{s}_2)$ 之最大值，且在 $s_3 = 1$ 的條件下，對於 $(\sigma_1\sigma_2\sigma_3\sigma_4) \in$

$\{(0010),(1101)\}$而言，l_{31}即是$SUM_{l3} \equiv \sum_{i=1}^{4} M(r_i(t),\sigma_i) + \frac{1}{2}\cdot L(\hat{s}_1) - \frac{1}{2}\cdot L(\hat{s}_2)$之最大值。如果$s_1 = 1$和$s_2 = 1$，那麼在$s_3 = 0$，對於$(\sigma_1\sigma_2\sigma_3\sigma_4) \in \{(0100),(1011)\}$而言，$l_{30}$即是$SUM_{l3} \equiv \sum_{i=1}^{4} M(r_i(t),\sigma_i) + \frac{1}{2}\cdot L(\hat{s}_1) + \frac{1}{2}\cdot L(\hat{s}_2)$之最大值，且在$s_3 = 1$的條件下，對於$(\sigma_1\sigma_2\sigma_3\sigma_4) \in \{(0111),(1000)\}$而言，$l_{31}$即是$SUM_{l3} \equiv \sum_{i=1}^{4} M(r_i(t),\sigma_i) + \frac{1}{2}\cdot L(\hat{s}_1) + \frac{1}{2}\cdot L(\hat{s}_2)$之最大值。圖3.8顯示了實現以上之描述計算出$l_{30}$和$l_{31}$之方塊圖。

● **第四子層級之計量單元：**

對於計量$-\left\|\hat{r}(t) - w(\hat{s}(t))\right\|^2 + \frac{1}{2}(s_1(t)-1)\cdot L(\hat{s}_1(t)) + \frac{1}{2}(s_2(t)-1)\cdot L(\hat{s}_2(t)) + \frac{1}{2}(s_3(t)-1)\cdot L(\hat{s}_3(t))$而言，它依據先前所還原之資料的軟式回授值$Li_1 = L(v_1^*(t-8\lambda) \oplus v_2^*(t-7\lambda))$，$Li_2 = L(v_2^*(t-5\lambda) \oplus v_3^*(t-4\lambda))$和$Li_3 = L(v_3^*(t-1\lambda))$。假設我們已經得到被還原之位元$\hat{s}_1$、$\hat{s}_2$和$\hat{s}_3$的軟式回授值分別為$L(\hat{s}_1)$、$L(\hat{s}_2)$和$L(\hat{s}_3)$。假設$SUM_{l3} \equiv \sum_{i=1}^{4} M(r_i(t),\sigma_i) + \frac{1}{2}(2s_1-1)\cdot L(\hat{s}_1) + \frac{1}{2}(2s_2-1)\cdot L(\hat{s}_2) + \frac{1}{2}(2s_3-1)\cdot L(\hat{s}$。$s_1, s_2, s_3 \in \{0,1\}$，我們看到對於$s_1 \in \{0,1\}$且$\sigma_1 \in \{0,1\}$，$\sigma_2 = 0 \oplus s_3$，$\sigma_3 = 0 \oplus s_2$和$\sigma_4 = 0$而言，$l_{40}$即$SUM_{l4}$之最大值，且對於$s_1 \in \{0,1\}$且$\sigma_1 \in \{0,1\}$，$\sigma_2 = 1 \oplus s_3$，$\sigma_3 = 1 \oplus s_2$和$\sigma_4 = 1$而言，$l_{41}$即是$SUM_{l4}$之最大值。因此，$l_{40}$和$l_{41}$可以很容易地藉由將脢特比解碼應用到圖3.9(a)的籬柵圖而被計算成存活者之最後計量，其中圖3.9(a)裡籬柵的第二個，第三個和最後一個分支上的符號分別依據值s_2，s_3和s_1而定。圖3.9 (b)顯示了使用這個籬柵圖計算出l_{40}和l_{41}之方塊圖。

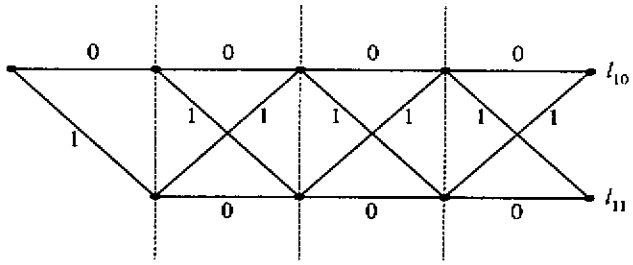
有了這四個計算子層級計量之單元，我們便可將之分別納入計算位元計量之單元。其中V_1和V_2包含了計算第一，第二和第三子層級計量之單元，V_3包含了計算第二和第三子層級計量之單元，而V_4包含了計算第四子層級計量之單元。

● V_1單元：

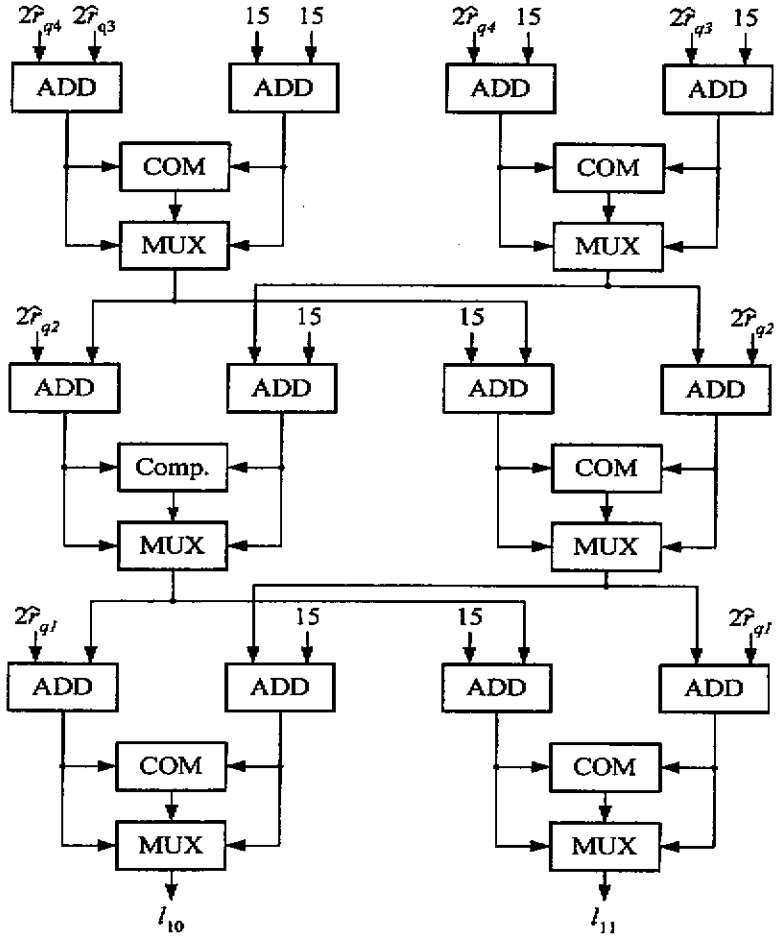
圖 3.10 顯示了計算$v_1=v_1(t)$之位元計量M_1的方塊圖，其包含了第一，第二和第三子層級計量之單元以及一些加法器，比較器和多工器。我們將所收到的資料$\hat{r}_a$送進第一子層級計量之單元以獲得子層級計量$-\|\hat{r}(t+8\lambda)-w(\hat{s}(t+8\lambda))\|^2$，將所收到的資料$\hat{r}_c$和已還原之資料的軟式回授值$Lc_1$送進第二子層級計量之單元以獲得子層級計量$-\|\hat{r}(t+6\lambda)-w(\hat{s}(t+6\lambda))\|^2$，另外將收到的資料$\hat{r}_f$和已還原之資料的軟式回授值$Lf_1$和$Lf_2$送進第三子層級計量之單元以獲得子層級計量$-\|\hat{r}(t+3\lambda)-w(\hat{s}(t+3\lambda))\|^2$。然後，我們將這些子層級計量混合起來以求得$v_1$之共同地最大位元計量。

● V_2單元：

圖 3.11 顯示了計算$v_2=v_2(t)$之位元計量的方塊圖，其包含了第一，第二和第三子層級計量之單元以及一些加法器，比較器和多工器。我們將所收到的資料$\hat{r}_b$和以還原之的資料軟式回授值Lb_1送進第一子層級計量之單元以獲得子層級計量$-\|\hat{r}(t+7\lambda)-w(\hat{s}(t+7\lambda))\|^2$，將所收到的資料$\hat{r}_d$和已還原之資料的軟式回授值$Ld_1$送進第二子層級計量之單元以獲得子層級計量$-\|\hat{r}(t+5\lambda)-w(\hat{s}(t+5\lambda))\|^2$，另外將收到的資料$\hat{r}_g$和已還原之資料的軟式回授值$Lg_1$和$Lg_2$送進第三子層級計量之單元以獲得子層級計量$-\|\hat{r}(t+2\lambda)-w(\hat{s}(t+2\lambda))\|^2$。然後，我們將這些子層級計量混合起來以求得$v_2$之共同地最大位元計量。



(a)



(b)

圖 3.6　第一子層級之計量的計算

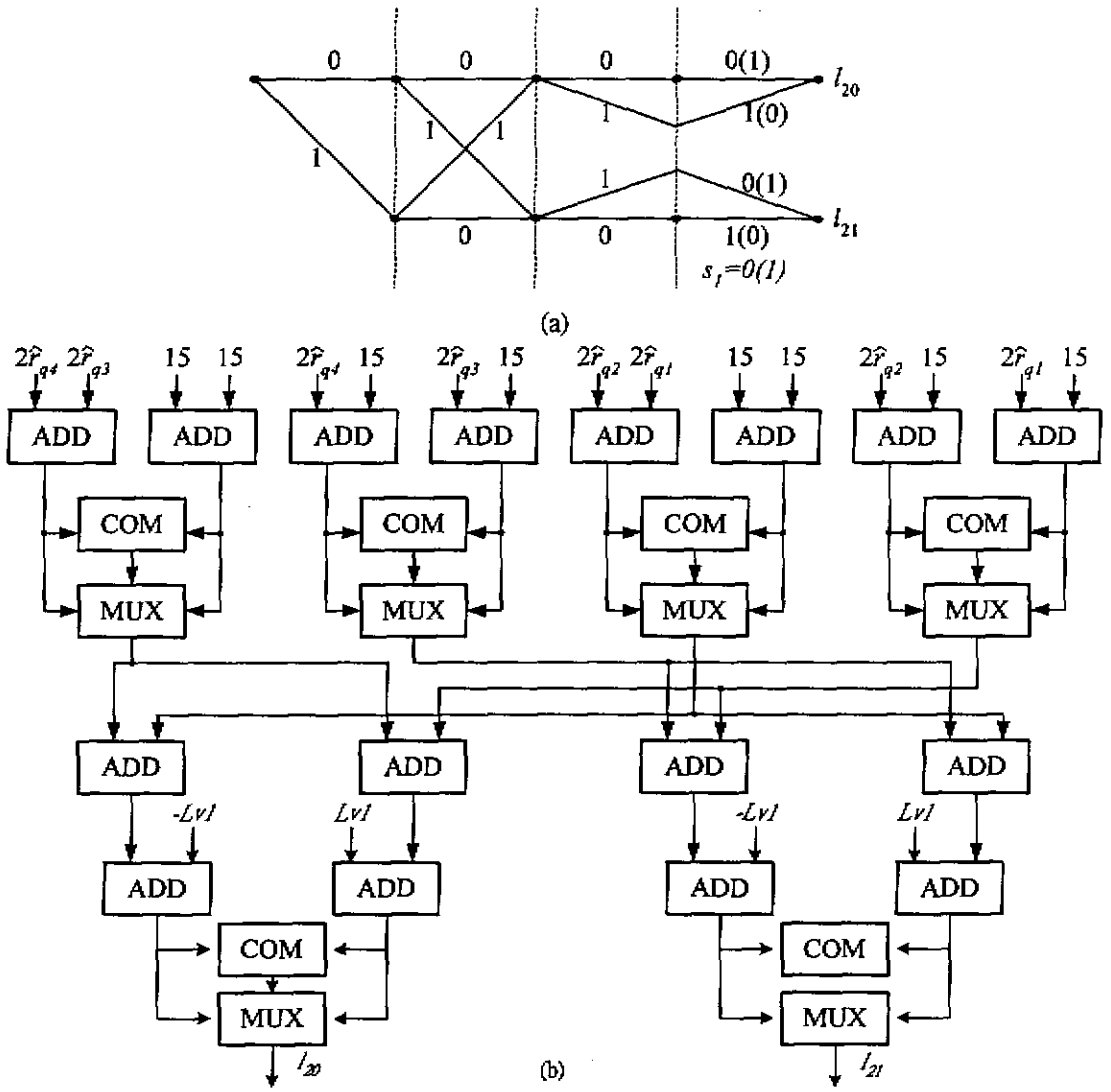


圖 3.7　第二子層級之計量的計算

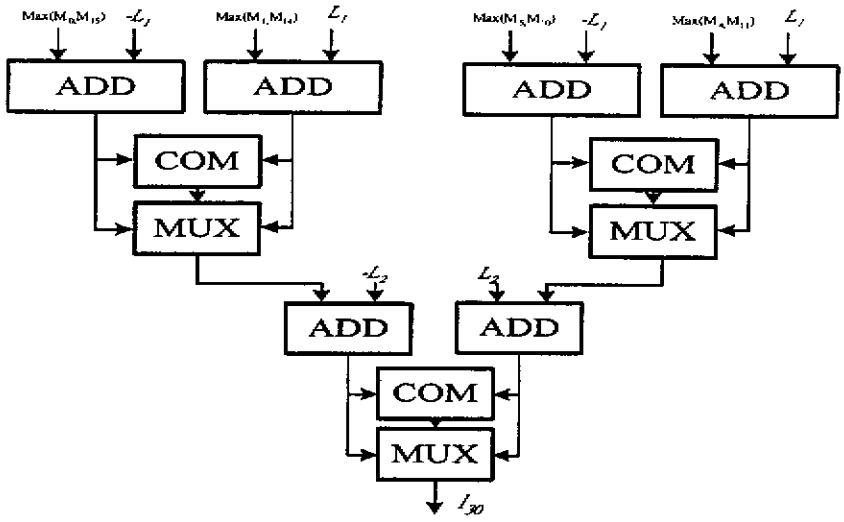


圖 3.8(a)　第三子層級之計量的計算

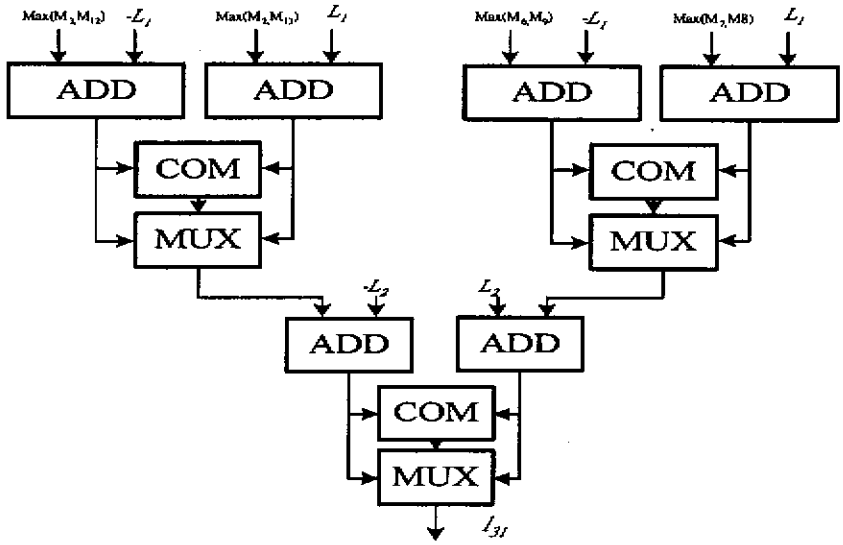


圖 3.8(b)　第三子層級之計量的計算

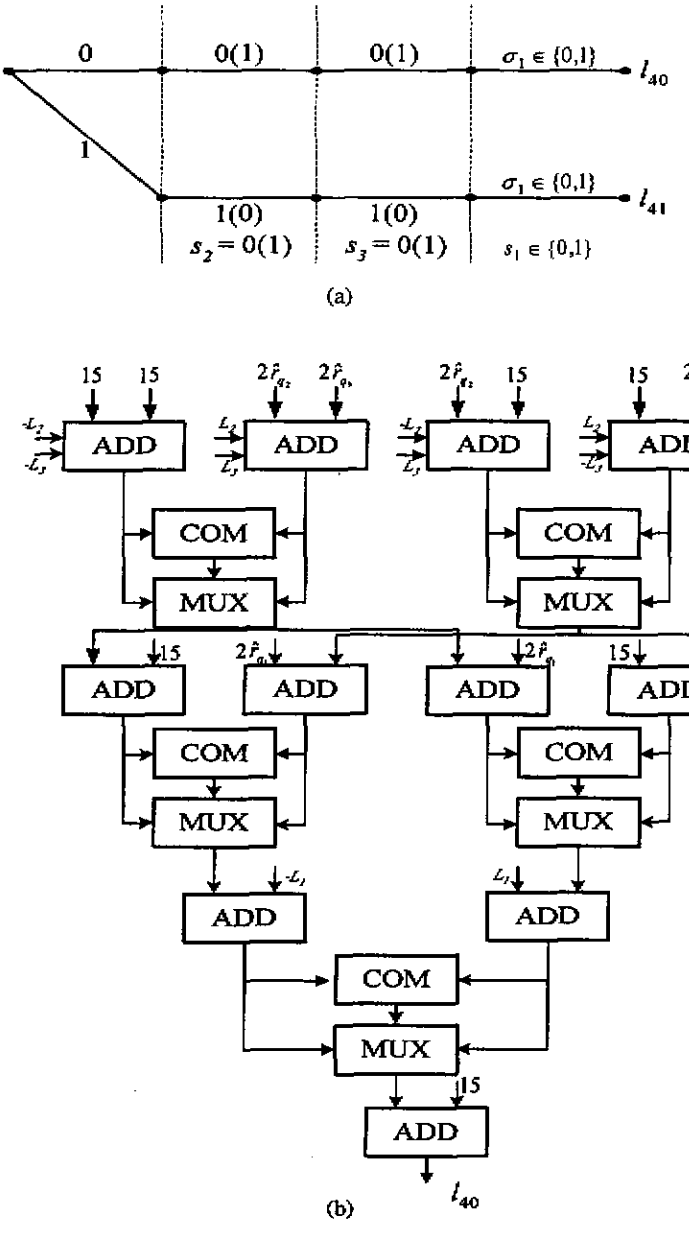


圖 3.9　第四子層級之計量的計算

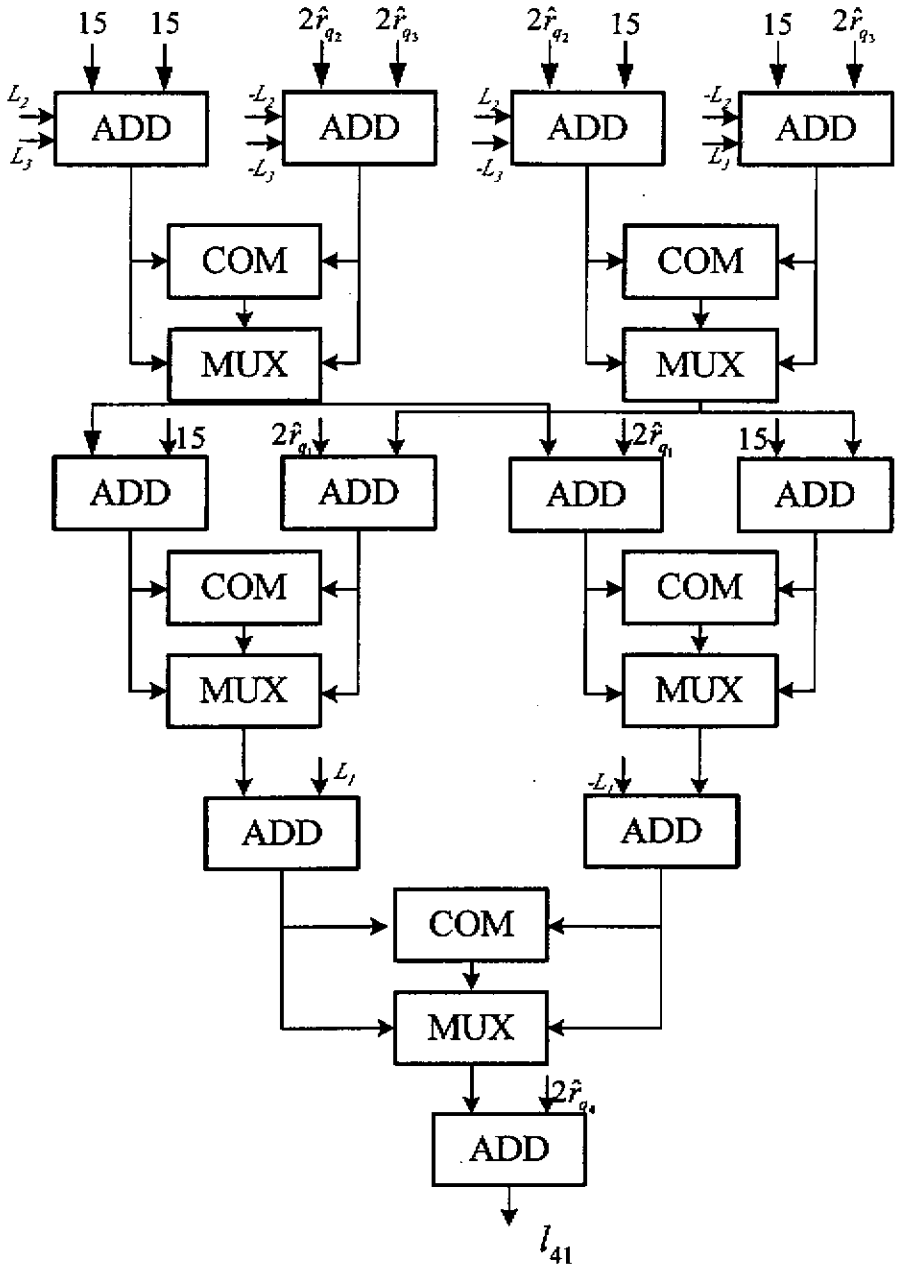


圖 3.9 續　第四子層級之計量的計算

● V_3單元：

圖 3.12(a)顯示了計算$v_3=v_3(t)$之位元計量的方塊圖，其包含了第一和第二子層級計量之單元以及一些加法器和多工器。我們將所收到的資料$\hat{r}_e$和以還原之資料的軟式回授值Le_1和Le_2送進第二子層級計量之單元以獲得子層級計量$-\|\hat{r}(t+4\lambda)-w(\hat{s}(t+4\lambda))\|^2$，另外將所收到的資料$\hat{r}_h$和已還原之資料的軟式回授值$Lh_1$和$Lh_2$送進第三子層級計量之單元以獲得子層級計量$-\|\hat{r}(t+1\lambda)-w(\hat{s}(t+1\lambda))\|^2$。然後，我們將這些子層級計量混合起來以求得$v_3$之共同地最大位元計量。

● V_4單元：

圖 3.12(b)顯示了計算$v_4=v_4(t)$之位元計量的方塊圖，亦即第四子層級計量之單元。我們將所收到的資料$\hat{r}_i$和以還原之資料的軟式回授值Li_1，Li_2和Li_3送進第四子層級計量之單元以獲得子層級計量$-\|\hat{r}(t)-w(\hat{s}(t))\|^2$，也就是$v_4$之位元計量。

有了這些從V_1，V_2，V_3和$V4$的輸出，我們利用子單元$\sum$來計算出16種可能之bm_0，…，bm_{15}。然後，我們將這16種可能之分支計量送進加總-比較-選擇單元。

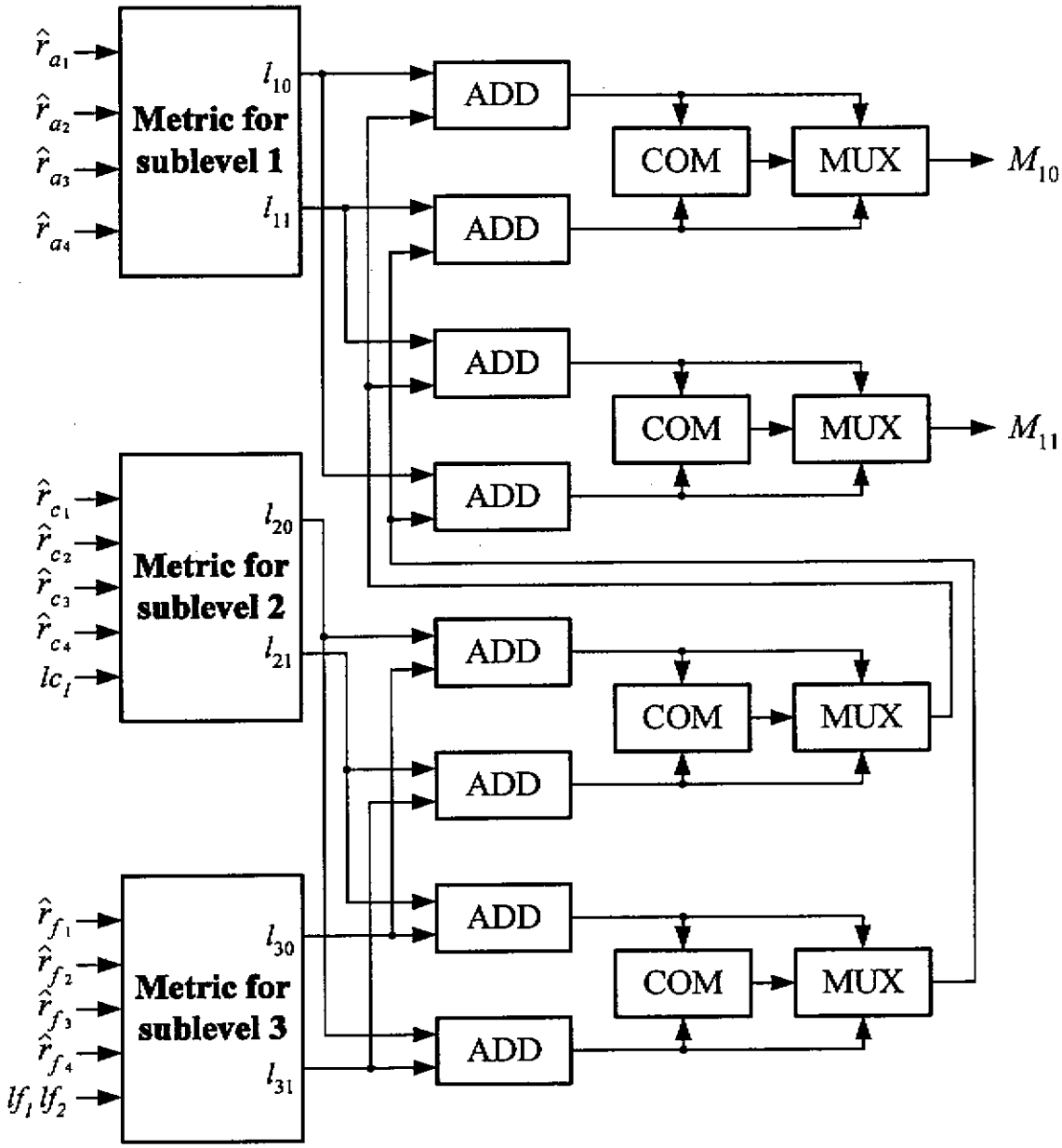


圖 3.10　V_1 單元：計算 v_1 之位元計量

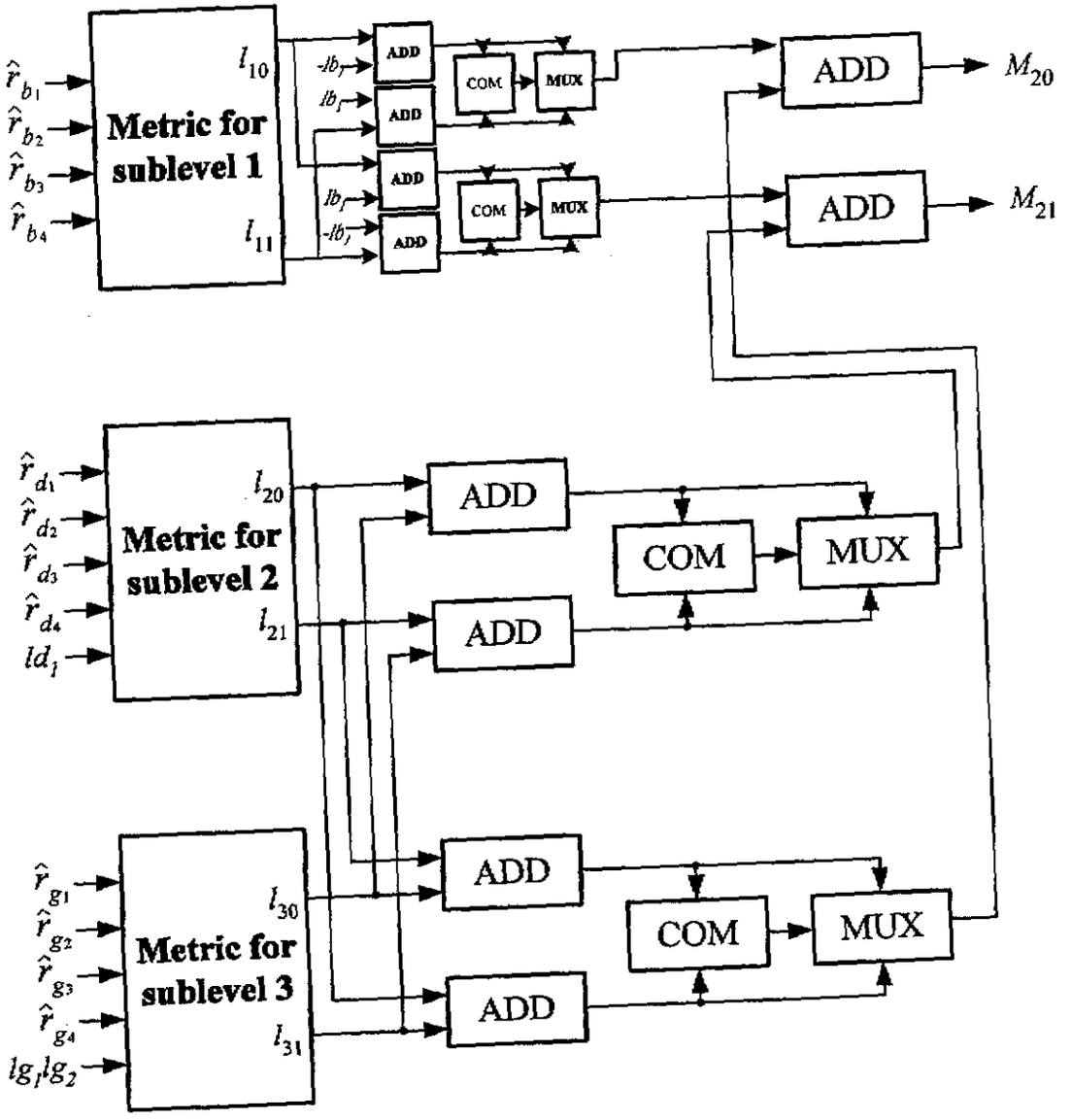
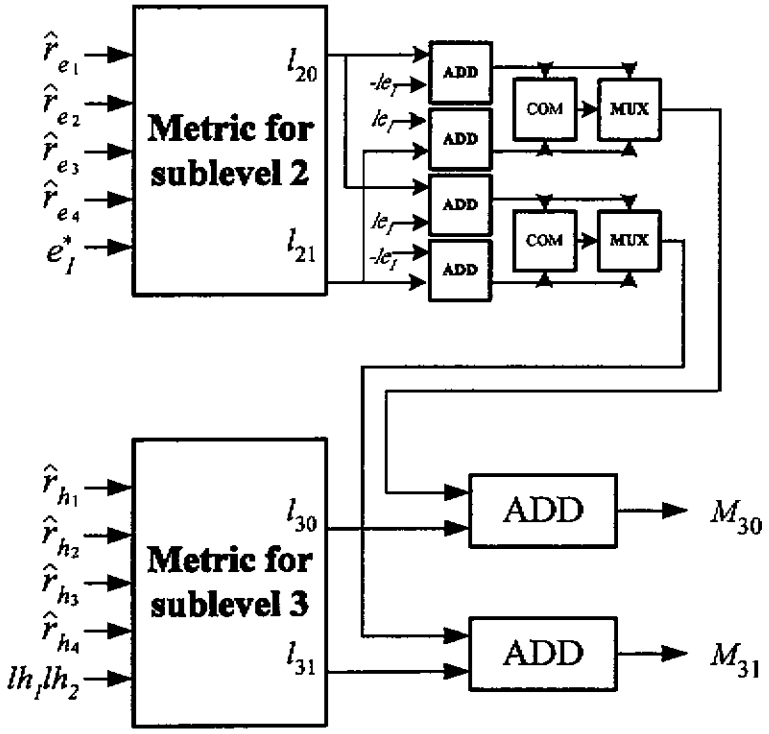
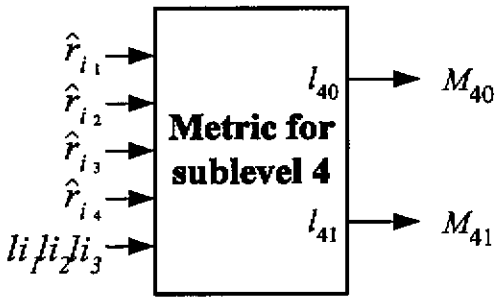


圖 3.11　V_2 單元：計算 v_2 之位元計量



(a)



(b)

圖 3.12　V_3和 V_4單元：分別計算v_3和v_4之位元計量

3.2.2 加總-比較-選擇 (ACS) 單元

ACS 的功能是將新的分支計量加到先前舊的路徑計量，比較新的路徑計量，並且選擇存活路徑。且為了得到軟式回授值，在此單元內，我們也需求出第二大的存活計量、存活路徑及正確路徑的 LLR 值。解碼器可以採取並聯或是序列的方式來配置這些 ACS 單元。如果採用並聯的方式來實現腓特比解碼器，則所需之 ACS 電路的數目正比於狀態點數。因此，其優點是解碼速率將會變快許多，但當狀態點越多時，龐大的繞線面積是其主要缺點。反之，若採用狀態-序列式 (state-sequential) 架構，則先前之狀態計量和新的(已更新的)狀態計量必須被儲存在記憶體。雖然其解碼速率較慢，但卻可以達到縮小晶片面積的目的，故其常被具有大的狀態點數之解碼器所採用[9] [10]。由於迴旋碼 C 使用了 16 個狀態點之解碼籬柵，我們只需要 16 個 ACS 處理器，也就是對於具多處理器之 ACSU 的 ACS_0，ACS_1，ACS_2，....， ACS_{15}。這些 ACS 處理器皆相似，因此我們在此只展列 ACS_0 單元。由圖 3.13 可得之狀態點最大存活計量 Sm_0，另外 $pmax_0[0]$和 $pmax_0[1]$為這時間點狀態點所選到之存活路徑。存活路徑將被送進下一級 SMU 以選擇出訊息位元。為了求出每一狀態點的正確路徑之 LLR 值，在 ACS_0 單元我們尚需求出每一個狀態點的第二大的存活計量 Sm_0'，並且得到其正確路徑之 LLR 值 $\Delta^m = \frac{Sm_0 - Sm_0'}{2}$ 及第二大之存活路徑 $psec_0[0]$和 $psec_0[1]$。此 LLR 值及 $psec_0[0]$和 $psec_0[1]$將送入 SOVA 解碼器得到所要之軟式回授值。另外，在數位實現之解碼器裡，計量在有限長度之暫存器下被表示成二元之定點數值，因此為了避免當更正或儲存計量時由於溢位/低於準位所造成的錯誤，計量標準化 (normalization) 是需要的。在[11]裡，Shung，Ungerboeck 和 Thapar 提到了幾種計量標準化的方法。為了簡單起見，我們採用定量位移的方法，也就是當所有存活者路徑計量都變成正值時，將所有存活者路徑計量向下位移一固定量。這個技巧可以藉由檢驗所有存活者計量之符號位元 (sign bit) 並當他們都變成 1 時重置所有符號位元來實現。在圖 3.13 裡，nm_0 即表示 ACS_0 之符號位元。

3.2.3 存活記憶(SM)單元

對於腓特比解碼器之存活列序的儲存而言，有兩個已知的演算法已經建立。他們是暫存器—交換 (Register-Exchange) 和向後追蹤 (Trace-Back) 方法[12]，[13]。但不幸的，每一個演算法皆有其缺點。暫存器—交換方法是一種概念最簡單且常常被使用的一個技術，但其在超大型積體電路 (VLSI) 之實現時會消耗較大的功率和需要較大面積。然而對於長的限制長度而言，向後追蹤方法需要一個較長解碼延遲。在此篇論文裡，使用 SOVA 解碼器得到軟式回授值已經使電路變得相當複雜，因此為了減少面積，我們採用向後追蹤方法來儲存路徑資料。由圖 2.5 之籬柵圖我們可以觀察出，前一時間點的狀態值，其實為進入下一時間點所選擇之存活路徑之資料。如果讓所選擇之存活路徑表示為 $y_t(S_t)$。則前一狀態點的值則為

$$(a_{t-1}, b_{t-1}) = y_t(S_t) \tag{3.1}$$

圖 3.14 為 SMU 單元的方塊圖。整個電路是由 40 級相同之電路單元所構成，每個電路單元均包含兩組電路。一為儲存路徑資料之電路，它是由 2 個 16 個狀態點之路徑資料栓鎖電路組成，每個狀態點路徑資料有 2 位元。另一為狀態點之向後追蹤電路，它是由一個 2 位元栓鎖電路及一個 2 位元 16 選 1 之多工器組成。2 位元栓鎖電路的值為狀態點值，此 2 位元經由多工器選擇該狀態點的路徑資料，作為向後追蹤之新狀態點值。p_0、p_1、p_2,.....、p_{15} 為 16 個狀態點之路徑資料，經由 40 級之向後追蹤解碼，最後，可求得解碼器之輸出位元 $\hat{u}_1$ 和 $\hat{u}_2$ 就被還原了。

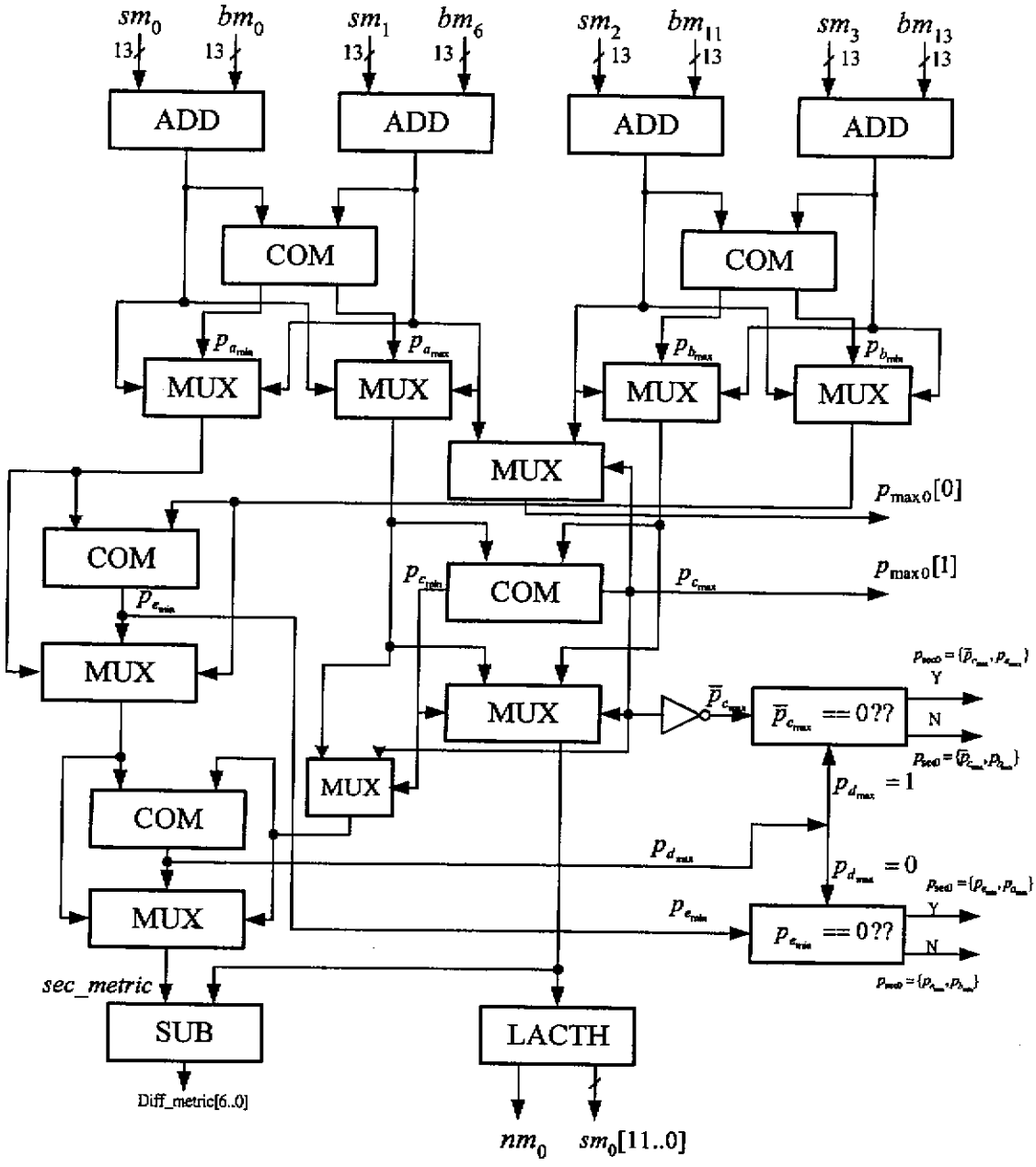


圖 3.13　加總-比較-選擇單元

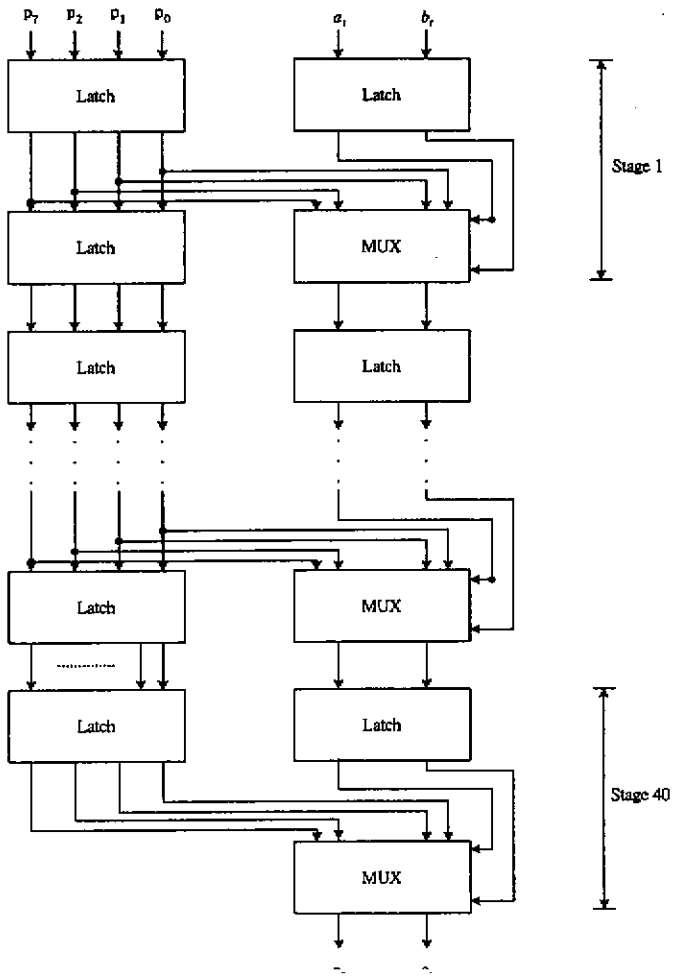

p_7 p_2 p_1 p_0
a_i b_i
Latch
Latch
Stage 1
Latch
MUX
Latch
Latch
Latch
MUX
Latch
Latch
Stage 40
MUX

圖 3.14　存活記憶單元

3.2.4 SOVA 解碼器

由 ACUS，SMU 單元，我們可得知每個狀態最大計量與次大計次之差值及選出最大相似路徑及次大相似路徑和解出位元$\hat{u}_t$，將這些值送入 SOVA 解碼器，以獲得對數-相似值作為軟式回授之用。因為原始的 SOVA 解碼器需比較存活路徑所有的狀態點，此種方式硬體複雜度太高，於是我們將兩個步驟 SOVA 解碼器[7]來實現 SOVA 演算法。

圖 3.15 顯示兩個步驟 SOVA 解碼器之方塊圖。在時間點$t+D$，由 ACUS，SMU 單元解出的且編過碼之位元$\hat{v}_t$、存活路徑P_t、路徑可靠度值Δ_t及狀態點S_t的前一時間之比較路徑狀態點S'_{t-1}送入 SOVA 解碼器。

圖 3.15 於水平分為 5 列，於垂直分為 D 階(stage)，而解碼隱藏時間(latency)為 $2D$。水平五列分別為：比較路徑狀態點(S'_{t-1} to S'_{t-D})，存活路徑(P_t to P_{t-2D+1})，已被解碼位元($\hat{v}_t$ to $\hat{v}_{t-2D}$)，對數-相似值($\hat{L}_{t-1}$ to $\hat{L}_{t-2D}$)路徑可靠度值(Δ_t to Δ_{t-D+1})。

SOVA 解碼器之工作原理描述如下：在時間點$t+\lambda$，假設比較路徑的狀態點為S'_{t-2}，由S'_{t-2}狀態點對應P_{t-3}的存活路徑，即可解出所對應之解碼位元u'_{t-3}即又經過腓特比編碼器得到編碼位元v'_{t-3}，且可向後追蹤下一個比較狀態點為S'_{t-3}。在同時間，v'_{t-3}跟$\hat{v}_{t-3}$比較，假如$\hat{v}_{t-3} \neq v'_{t-3}$則就更新 LLR 值依據$\hat{L}_{t-3} = \min(\hat{L}_{t-3}, \Delta_{t-1})$。在時間點 t+1，即可得到位元$\hat{v}_{t-2D}$及對應之 LLR 值$\hat{L}_{t-3}$。

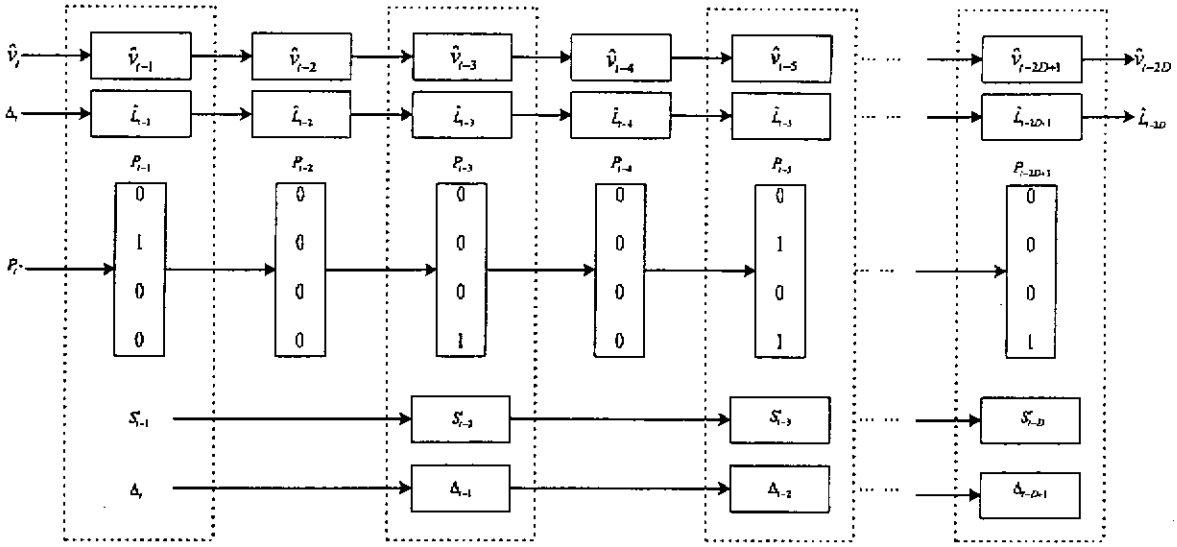


圖 3.15 兩個步驟 SOVA 解碼器之方塊圖

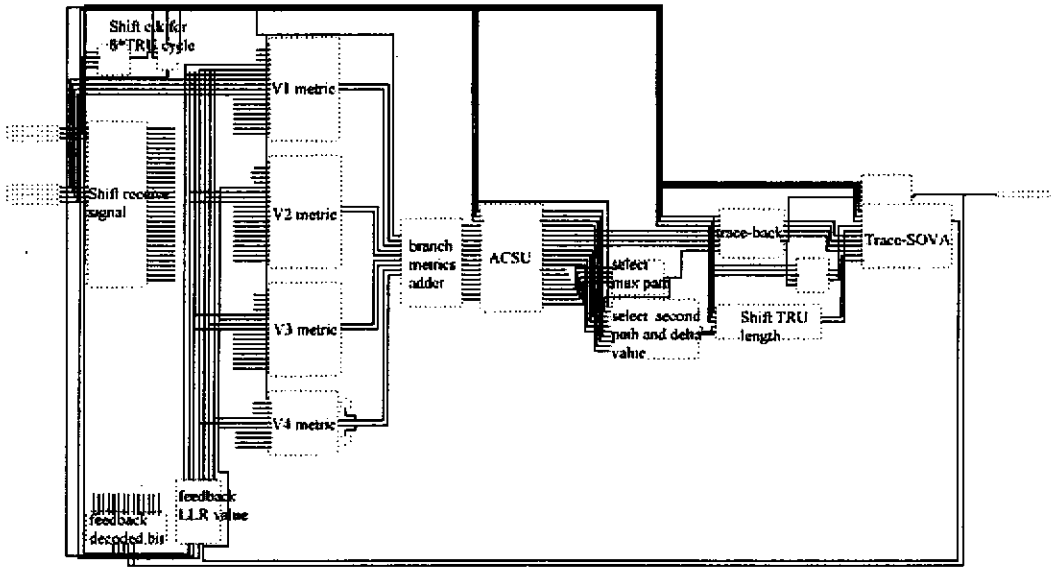


圖 3.16 籬柵碼 T 之 SOVA 解碼器的電路圖

圖 3.16 顯示了 4 個狀態點之籬柵碼 T 之 SOVA 解碼器的電路圖。在這裡我們顯示此解碼器之最上層電路，也就是如圖 3.2 及圖 3.3 所示之方塊圖。此圖是經由掃描器掃瞄後獲得，故解析度不是很高，但我們仍可以大約看出其與方塊圖之間的關係。最左邊的方塊為用來儲存已量化

之資料，接下來的四個方塊分別與圖 3.2 之 V_1 ~ V_4 相對應。中間之方塊是用來計算分之記量，而右邊的四個方塊分別與 ACS_0 ~ ACS_{15} 相對應。右邊之方塊分別為 SMU 及 SOVA 解碼器。

第四節

硬體模擬和性能測試

在本章裡，我們將描述在第四章裡所設計出之籬柵碼 T 之次佳化解碼器電路的測試方法。在良好之測試環境下，我們可以輕易的完成電路的功能驗證。

4.1 測試環境之介紹

4.1.1 模擬程序

籬柵碼 T 之肥特比解碼器電路模擬程序如下：

1) 使用 Altera 之 MAX+PLUS II 可程式邏輯發展軟體作邏輯設計和邏輯模擬（功能性）。

2) 使用 Altera FPGA 邏輯合成、平面配置和繞線產生可下載之.pof 檔及包含時序資訊之.snf 檔。

3) 對邏輯時序確認作後置模擬。

4) 下載至 Altera FPGA。

4.1.2 性能測試向量之產生

為了方便完成測試工作，我們使用 PC 上之 C 語言來產生用來模擬籬柵碼 T 之肥特比解碼器的輸入向量，也就是這些經過量化之訊號。首先籬柵碼 T 之編碼器由 $R=2/4$ 之迴旋碼 C 的編碼器、迴旋處理器和信號對應器所組成。籬柵碼 T 之編碼器的輸入是由隨機數值產生器所產生，也就是隨機地選出我們所要傳送的訊息，且這些訊息是數位的形式。由於我們所考慮之解碼器的輸出是二元之 4-tuple，其中數字 4 表示編碼器之輸出符號的位元數，也就是迴旋碼 C 之編碼器的輸出碼字（code word）之位元數，因此編碼器的輸出經過 4 維之 BPSK 調變後送出。這些信號進入傳輸通道，在這裡我們所考慮的通道是訊雜比（signal to noise ratio）為 4dB 之 AWGN 通道。最後，在接收端我們將這些每一個時間點所收到之連續信號分別經由量化器量化成 Q 個層級，也就是用 log_2Q 位元來表示一個符號。這些 $4*logQ$ 位元之資料（4 層之籬柵編碼）就被用來當作籬柵碼 T 之肥特比解碼器的輸入，且被儲存在檔案中。在這裡，我們選擇輸入資料的取樣點數為十萬個點。另外，我們還需準備用來比對硬體解碼電路上所量測到的資料，這些資料是由軟體模擬產生，也就是使用 C 語言來完成解碼，並取出我們所要的一些資料，在這裡我們選擇狀態點 0 之狀態計量以及解碼器的輸出作為檢視電路是否正常工作的依據。同樣地，這些資料亦分別被儲存在不同的檔案中，且其取樣點數均為十萬個點。

4.1.3 解碼器之測試環境

圖 4.1 顯示了籬柵碼 T 之腓特比解碼器的測試環境。首先，我們將經由軟體 Maxplus II 編譯過之解碼電路檔案由下載纜線(download cable or byteblester）下載至 FPGA 上，然後再把由 PC 產生之已量化的資料轉換成可被樣本產生器（pattern generator）接受之輸入檔案，也就是格式是十

六進制（hexadecimal）的檔案。另外，我們將欲觀察之解碼器的輸出點接至邏輯分析儀（logic analyzer）探針上以獲得欲比對之取樣資料。在這裡我們選擇狀態 0（S_0）之存活路徑計量以及解碼器的解碼輸出以作為檢查電路是否正常運作的依據且取樣資料長度為十萬個取樣點。注意，包含 FPGA、邏輯分析儀和樣本產生器之所有電路之接地（grounding）需連接在一起以確保邏輯準位相同。

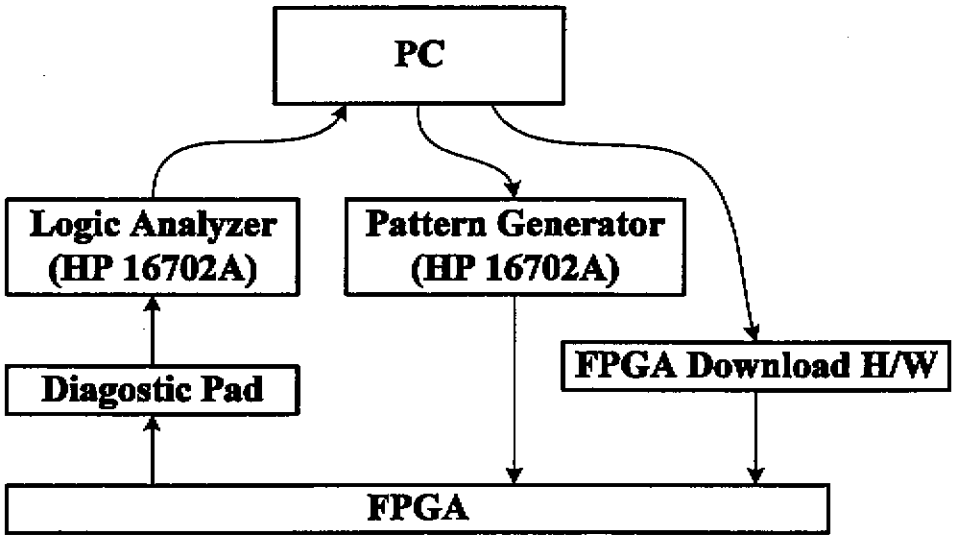


圖 4.1 解碼器測試環境

在獲得這些從硬體解碼電路上所獲得之取樣資料之後，我們將檔案從邏輯分析儀上拷貝至 PC。為了比對這十萬個取樣資料，我們使用 C 語言撰寫一個簡單的小程式來幫助我們完成比對工作，並將比對結果儲存在檔案之中以便我們檢視。注意，由於我們所實現的電路是數位的，因此只要脈衝週期大於記憶原件（如暫存器，RAM 等）之設定時間（set-up time），使得其能避開暫態效應並在穩態時擷取正確的資料，則這些從硬體解碼電路上所獲得之取樣資料將會與軟體模擬產生之資料完全相等。

4.2.1 解碼器之測試結果

在上述之測試環境下，我們將在低速下所量測到之數據與 C 語言模擬之資料相比較，發現兩者之數據完全相等。於是，經由不斷地提升脈衝速率且比對數據，我們發現具有 4 個狀態點之迴旋碼 T 之解碼器的脈衝速率最快可以達到 20MHz 且每個脈衝週期輸出 2 個資訊位元。因此，這個具有 4 個狀態點之迴旋碼 T 之解碼器的解碼速率在我們所設計的架構下可以達到 40MHz；而 16 個狀態點之迴旋碼 T 之解碼器在我們所設計的架構下可以達到 20MHz；而 16 個狀態點之迴旋碼 T 之 SOVA 解碼器在我們所設計的架構下只可達到 8MHz。

4.2.2 討論

由圖 2.5 可知，具 4 層迴旋處理器和信號對應器之 4 個狀態點的迴旋碼 T 的次佳化解碼器在中至高訊雜比時，其位元錯誤率優於傳統之 64 個狀態點之迴旋碼。但為了數位電路實現，我們採用量化層數 $Q=8$ 之量化器來量化這些受到雜訊干擾之信號。雖然這造成我們的解碼器之性能略微下降，但與傳統之 64 個狀態點的迴旋碼相比，我們的解碼器之位元錯誤率仍具有相當之優越性。

另外，表 4.1 顯示了迴旋碼 T 之次佳化解碼器和傳統之 64 個狀態點之迴旋碼在硬體複雜度上之比較，其中兩者皆採用併聯處理器的實現方式來設計解碼器。對於碼率 $R=1/2$，$m=6$，$Q=8$ 之傳統迴旋碼而言，其需要 9 個位元來表示存活路徑計量，與 $Q=8$ 之具 4 層迴旋處理器和信號對應器之 4 個狀態點的迴旋碼 T 的次佳化解碼器相同。

雖然使用相同的演算法之傳統脒特比解碼器在硬體複雜度上會根據不同之電路設計者而有所變化，但表 4.1 所列之基本元件的數目仍可以提供有價值之資訊。我們可以看到籬柵碼 T 之解碼器所需要的位元數低於

傳統的脒特比解碼器。另外，假使考慮設計 VLSI 晶片，那麼使用迴旋碼 T 之次佳化解碼器的好處將會更顯著，這是因為其在 ACSU 裡只需要 4 個處理器的緣故。而傳統之脒特比解碼器在採用併聯架構下需要 64 個處理器，因此其繞線將非常複雜且電路面積也會因此而大大地增加。故對於 4 個狀態點的籬柵碼 T 的次佳化解碼器而言，其在電路實現和解碼速率上仍保有相當之優勢。

	$m=2, R=2/4$			$m=6, R=1/2$		
	a*	b*	c*	a*	b*	c*
ADD	4	46				
	5	34				
	6	18				
	7	6				
	8	32		3	4	
	9	32	1048	8	128	1036
COM	5	16				
	6	14				
	7	4				
	8	2				
	9	12	316	8	64	512
MUX	2	192				
	4	18				
	5	14				
	6	24		1	2048	
	7	4		8	64	
	8	14	810	64	1	2624
FLIP-FLOP	2	96				
	9	4		1	2048	
	12	192	2532	8	64	2560

a*: 位元數, b*: 總數, c*: 總位元.

表 4.1　籬柵碼 T 之次佳化解碼器與 64 個狀態點之傳統的迴旋碼之解碼器在複雜度上之比較

由圖 2.11 可知，具 4 層迴旋處理器和信號對應器之 16 個狀態點的籬柵碼 T 的次佳化解碼器在中至高訊雜比時，其位元錯誤率優於傳統之 256 個狀態點之迴旋碼。但為了數位電路實現，我們採用量化層數 $Q = 16$ 之量化器來量化這些受到雜訊干擾之信號。我們可以看到 $Q = 16$ 之籬柵碼 T 與傳統之 256 個狀態點的迴旋碼相比，我們的解碼器之位元錯誤率仍具有相當之優越性。

另外，假使考慮設計 VLSI 晶片，那麼使用迴旋碼 T 之次佳化解碼器的好處將會更顯著，這是因為其在 ACSU 裡需要 16 個處理器的緣故。而傳統之腓特比解碼器需要 256 個處理器，因此其繞線將非常複雜且電路面積也會因此而大大地增加，在實作上是不能被忍受的。因此 256 個狀態點之標準的肥特比解碼器通常採用狀態—序列式的方式來實現，但這卻會遭受到解碼速率降低的損害。對於 16 個狀態點的籬柵碼 T 的次佳化解碼器而言，由於其只有 16 個狀態點，故仍可採用併聯處理器的架構來實現之，因此其在解碼速率上仍保有相當之優勢。

具 4 層迴旋處理器和信號對應器之 4 個狀態點的迴旋碼 T 的次佳化解碼器在中至高訊雜比時，其位元錯誤率優於傳統之 64 個狀態點之迴旋碼。而由圖 3.4 的模擬得知，回授方式採用軟式輸出值的錯誤率優於採用硬式輸出方式。為了數位電路實現，我們採用量化層數 $Q = 16$ 之量化器來量化這些受到雜訊干擾之信號。雖然這造成我們的解碼器之性能略微下降，但與採用硬性輸出回授方式比較，我們的解碼器之位元錯誤率仍具有相當之優越性。

另外，表 4.2 顯示了迴旋碼 T 之次佳化解碼器採用硬式回授方式和迴旋碼 T 之次佳化解碼器採用軟式回授方式，在硬體複雜度上之比較，其中後者使用 SOVA 方式實現軟式回授來設計解碼器。

採用 SOVA 軟式回授的方式，因為在 BMU 單元需加上軟式輸出值作

為調整分支計量用，且於 ACSU 單元需選擇出第二大的相似路徑，及加上 SOVA 解碼器，所以其電路之複雜度比硬式回授方式複雜多了。

	m=2 ，R=2/4			*M=2 ，R=2/4* with SOVA		
	Bit	Total	Total bits	Bit	Total	Total bits
ADD	4	46	1048	5	40	1746
	5	34		6	10	
	6	18		7	2	
	7	6		10	72	
	8	32		11	32	
	9	32		12	16	
				13	16	
COM	5	16	316	6	14	666
	6	14		7	2	
	7	4		8	1	
	8	2		10	28	
	9	12		14	20	
MUX	4	18	426	6	14	834
	5	14		7	2	
	6	24		8	1	
	7	4		10	28	
	8	14		14	32	
FLIP FLOP	2	96	2532	2	132	2730
	9	4		3	22	
	12	192		8	300	

表 4.2　離柵碼 T 之 SOVA 解碼器與具有迴旋處理器之解碼器採用硬式迴授在複雜度上之比較

第五節

硬體複雜度之分析

在此部份描述了使用 FPGA 來完成籬柵碼 T 的次佳化解碼器的設計和實現，其中籬柵碼 T 是由一個 $R=2/4$ 之傳統迴旋碼（case (a): $m=2$，case (b): $m=4$，case(c):m=4(SOVA)），後面接著一個 4 層之迴旋處理器和一個信號對應器所組成。藉由使用迴旋碼 C 之解碼籬柵和一些迴授資訊，這個籬柵碼 T 可以被次佳化地解碼。在這種編碼架構之下，這個籬柵碼 T 相等於一個具有極大的編碼器記憶數目之傳統迴旋碼，也就是一個具有很大的限制長度之籬柵碼。case (a)之迴旋碼 C 為一個 $R=2/4$，$m=2$，且生成函數 $g_0=(0\ 3)_8$，$g_1=(3\ 3)_8$，$g_2=(3\ 1)_8$，$g_3=(1\ 2)_8$ 之傳統迴旋碼。而 case (b)之迴旋碼 C 為一個 $R=2/4$，$m=4$，且生成函數 $g_0=(0\ 7)_8$，$g_1=(2\ 5)_8$，$g_2=(7\ 7)_8$，$g_3=(7\ 2)_8$ 之傳統迴旋碼。由圖 2.5 可知，4 個狀態點之籬柵碼 T 的次佳化解碼器在中至高訊雜比時，其位元錯誤率優於傳統之 64 個狀態點之迴旋碼。另外，由圖 2.11 可知，16 個狀態點的籬柵碼 T 的次佳化解碼器在中至高訊雜比時，其位元錯誤率優於傳統之 256 個狀態點之迴旋碼。而由圖 3.4 可知，16 個狀態點的籬柵碼 T 採用 SOVA 解碼器，其位元錯誤率優於採用硬式迴授值的 16 個狀態點的籬柵碼 T。

由於 case (a)之迴旋碼 C 只有 4 個狀態點，而 case (b)之迴旋碼 C 只有 16 個狀態點，case(c)之迴旋碼 C 也只有 16 個狀態點，因此籬柵碼 T 之次佳化解碼器皆適合採用多處理器的方式來實現。傳統的 256 個狀態點的迴旋碼其腓特比解碼器中的 ACSU 單元如採用狀態-並列式的方式來實現，則需要 256 個處理器，因此其繞線將非常複雜且電路面積也會因此大大地增加，在實作上是不能忍受的。因此 256 個狀態點之標準的腓特比解碼器通常採用狀態-序列式的方式來實現，但卻會遭受解碼速率降低的損害。對於 16 個狀態點的籬柵碼 T 的腓特比解碼器而言，由於其只

有 16 個狀態點，故仍可採用並聯處理器的架構來實現之，其複雜度其解碼速度皆比傳統的 256 狀態點腓特比解碼器仍保有相當之優勢。對於 4 個狀態點之籬柵碼 T 的次佳化解碼器，其最大之解碼速率可以達到 40Mbit/s。對於 16 個狀態點之籬柵碼 T 的次佳化解碼器，其最大之解碼速率可以達到 20Mbit/s。對於 16 個狀態點之籬柵碼 T 的 SOVA 解碼器，其最大之解碼速率可以達到 8Mbit/s。相較於傳統之腓特比解碼器，籬柵碼 T 之次佳化解碼器在硬體複雜度和解碼速率上均保有較大之好處。而採用 SOVA 解碼器，其錯誤執行能力會優於採用硬式迴授方式，但其硬體複雜度及解碼速率均劣於後者。而採用 SOVA 解碼器之 16 個狀態點的籬柵碼，其解碼步驟中，只採用一次的 SOVA 解碼理論，而 16 個狀態點的渦輪碼(Turbo Code),因此，採用 SOVA 解碼器的具有迴旋處理器之籬柵碼的硬體複雜度比渦輪碼的大大地降低，但其錯誤執行能力較渦輪碼低。

另外，我們將所設計之電路實現在 Altera 之 FLEX10K200 FPGA 上，且使用 4dB 之 AWGN 資料。性能測試結果顯示了 4 個和 16 個狀態點之籬柵碼 T 的次佳化解碼器在 4dB 雜訊下之錯誤更正能力完全正常。

第六節

結論

在此部份，我們利用 FPGA 實現驗證 4 個狀態點及 16 個狀態點之具有迴旋處理器之籬柵碼硬體電路，迴授則採用硬式迴授方式。亦利用 FPGA 設計採用 SOVA 演算法之具有迴旋處理器之 16 個狀態點之籬柵碼硬體電路。

採用硬式迴授之具有迴旋處理器和信號對應器，且狀態數分別為 4 和 16 的籬柵碼，而它們較分別具有 64 和 256 個狀態點之傳統迴旋碼擁有更好的錯誤性能。而採用 SOVA 解碼器之具迴旋處理器和信號對應器之 16 狀態點籬柵碼，其因只採用一次的 SOVA 演算法，比起渦輪碼的需採用多次之 SOVA 演法，其硬體複雜度和解碼時間皆大幅度地降低。藉由硬體實現，我們證明具有迴旋處理器的籬柵碼的結構比傳統之 256 個狀態點迴旋碼和渦輪碼有些電路設計上的優點，因此可用於第三代行動通訊之編解碼技術中。

參考文獻

[1] Wang, J.-Y. and Lin, M.-C. "On constructing trellis codes with large free distance and low decoding complexities," *IEEE Trans. Inform. Theory,* vol. 45, no. 9, pp. 1017-1020, Sept, 1997.

[2] J. Hagenauer and P. Hoeher, "A Viterbi Algorithm with Soft-Decision Outputs and its Applications," *Institute for Communications Technology,* West-Germany, *IEEE,* 1989.

[3] J. Hagenauer, E. Offer and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. Inform. Theory,* vol. 42, no. 2 , March 1996.

[4] C. Berrou, P. Adde, E. Angui and S. Faudeil, "A Low Complexity Soft-Output Viterbi Decoder Architecture," *Integrated Circuit for Telecom. Laboratory,* France, *IEEE,* 1993.

[5] Olaf J. Joeressen and Heinrich Meyr, "A New Postprocessing Architecture For Soft output Viterbi Decoding," *Integrated System for Signal processing*, Germany, *IEEE 1994.*

[6] Y. Wang, C. Y. Tsui and R. Cheng, "A Low Power VLSI Architecture of SOVA- based Turbo-code decoder using Scarce State Transition Scheme," *IEEE International Symposium on Circuit systems,* May 28-31, 2000, Geneva, Switzeriand.

[7] C. Y. Shieh, "Circuit Design of Suboptimal Deocoders for Trellis Codes with a Convolutional Processor," *Paper, National Taiwan University,* 2000.

[8] Y. K. Lee and H. Lou, "Normalization, windowing and quantization of soft decisition Viterbi decoder inputs in CDMA," *Vehicular Technology Conference,* 1999 *IEEE* 49th Volume: 1 , 1999 , Page(s): 221-225 vol.1

[9] Jerold A. Heller, "Viterbi Decoding for satellite and Space Communication," *IEEE Trans. Comm. Tech.,* Oct,1971.

[10] Park, J. -H. and Rho, Y.-C., "Performance Test of Viterbi Decoder for Wideband CDMA System," Design Automation Conference, 1997. Proceedings of the ASP-DAC '97 Asia and South pacific , 1997 , : 19-23.

[11] C. Bernard Shung, G. Ungerboeck and Hemant K. Thapar, "VLSI Architecture for Metric Normalization in the Viterbi Algorithm, " *Communitcation, ICC '90, IEEE International Conferecnce ,pp. 1723-1729* vol.4

[12] G. Feygin, and P. G. Gulsk, "Survivor Sequence Memory Management in Viterbi Decoder," *Circuits and Systems,* 1991., *IEEE International Sympoisum on,* pp 2697-2970 vol.5

[13] T. K. Truong, M. T. Shin, I. S. Reed and E. H. Satorius, " A VLSI Design for Trace-Back Viterbi Decoder" *IEEE Trans. Comm.,* vol.40, No. 3, Mar. 1992.

Conclusion

In this project, we not only do the performance analysis but also consider the practical implementation of trellis codes with additional processing units. In Part A, we apply these trellis coding schemes with proposed suboptimum decoding in the additive white Gaussian noise (AWGN) channel, independent fading channel and correlated fading channel. Simulation shows that trellis coding with additional processing units can achieve comparable and sometimes better error performance in these channels with moderate decoding complexity and moderate decoding delay as compared to the conventional trellis code and turbo code. In Part B, we use FPGA to implement convolutional codes with a convolutional processor for which its decoding trellis has only 4 or 16 states.With the circuit design, we verify that convolutional coding with a convolutional processor does have the advantage of low complexity in circuit implementation. Based on the result of this project, trellis coding with additional processing units can achieve high transmission reliability and low complexity of implementation and hence can be served as a candidate in the third generation wireless mobile system in addition to the conventional convolutional code and turbo code.