

國立交通大學

電機學院 IC 設計產業研發碩士班

碩士論文

時空籬柵碼之解碼器設計與實現
Design and Implementation for Decoder of Space-Time
Trellis Codes



研究生：宋蓬麟

指導教授：王忠炫 教授

中華民國九十六年一月

時空籬柵碼之解碼器設計與實現
Design and Implementation for Decoder of Space-Time Trellis Codes

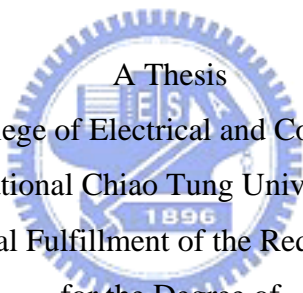
研究生：宋蓬麟

Student : Perng-Lin Sung

指導教授：王忠炫

Advisor : Chung-Hsuan Wang

國立交通大學
電機學院 IC 設計產業研發碩士班
碩士論文



A Thesis
Submitted to College of Electrical and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Industrial Technology R & D Master Program on
IC Design

March 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年一月

學生：宋蓬麟

指導教授：王忠炫
翁芳標
鄭立德

國立交通大學電機學院產業研發碩士班

摘 要

由於通訊結合多媒體服務需求的急遽成長，無線通訊(例如第三代行動通訊系統與無線區域網路)朝向高資料量傳輸發展已成為不可避免的趨勢。時空籬柵碼，結合錯誤更正碼，調變以及傳送和接收分集，可以有效的降低無線通訊環境的衰減效應。因為時空籬柵碼的編碼結構與迴旋碼相似，同樣由暫存器組成，所以產生相似的籬柵圖可用維特比演算法做解碼。一般而言，解碼器架構是由支計量單位，加-選擇-比較單位以及存活記憶體單位所組成。雖然加-選擇-比較單位是維特比演算法主要的計算單位，但是存活記憶體單位確是解碼器架構實現到硬體上需要做最多的考量，因為佔據的晶片面積，耗電量以及影響解碼速度都是最大的。本論文針對解碼器硬體架構，包括傳統存活記憶體管理架構，高速 VLSI 追溯解碼架構和混合式存活記憶體管理架構的優缺點做出分析。並提出結合改良式暫存器交換和排列網路電路的架構。最後用軟體模擬以及實現在 FPGA 硬體上。

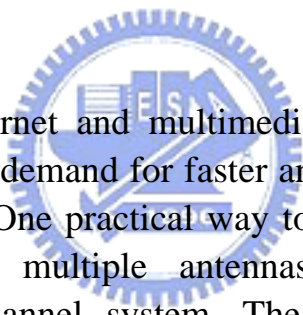
Design and Implementation for Decoder of Space-Time Trellis Codes

student : Perng-Ling Sung

Advisors : Dr. Chung-Hsuan Wang
Dr. Fang-Biau Ueng
Dr. Li-Der Jeng

Industrial Technology R & D Master Program of
Electrical and Computer Engineering College
National Chiao Tung University

ABSTRACT



With the integration of internet and multimedia applications in next generation wireless communication, the demand for faster and higher data rate communication service becomes inevitable. One practical way to increase the capacity of wireless channel is by employing multiple antennas to create multiple-input and multiple-output (MIMO) channel system. The coding technique that employs multiple antennas is space-time coding. Space-Time Trellis Codes, joins the design of error control coding, modulation, transmit and receive diversity, is an effective signaling scheme able to combat the effect of fading in wireless environment. The decoder for such codes employs the Viterbi algorithm to perform maximum likelihood decoding since the encoder structure consists of shift registers that can be interpreted to trellis diagram as in convolutional codes. The decoder architecture consists of branch metric, add-select-compare, path metric memory and survivor memory units. Although the add-compare-select unit is the main component that performs the Viterbi algorithm, the survivor memory unit requires more attention because it occupies most of the decoder chip area, power consumption and is a critical data-processing component that affects the speed of decoder. The goals of this thesis are to develop a survivor memory unit that can achieve the advantages of low latency, low hardware complexity, and reduced chip area. The proposed decoder architecture is implemented in both software simulation and FPGA implementation.

Acknowledgement

I would like to thank my supervisor Dr. Chung-Hsuan Wang. This work cannot be possibly finished without his assistance. Although my parents never understand my work, I would still like to thank their understanding and support. And finally I had great time spent with all my colleagues in the communication technology and audio processing lab. Thank you all.



Contents

Chinese Abstract	i
English Abstract	ii
Acknowledgement	iii
Contents	iv
List of Figures	vi
1 Introduction	1
2 Space-Time Coded System	3
2.1 Space-Time Trellis Encoder Structure	5
2.2 Performance Analysis of Space-Time Codes	9
2.3 Space-Time Code Design Criterias	14
3 Decoder Architectures for Space-Time Trellis Codes	18
3.1 The Viterbi Algorithm and Its Decoder Architecture	18
3.2 Add-Compare-Select Architectures	21
3.3 Survivor Memory Architectures	22
3.3.1 VLSI Design of Trace-Back Architectures	23
3.3.2 Pipelined Trace-Back Architectures	24



3.3.3	Hybrid Survivor Memory Architectures	29
3.3.4	Proposed Architecture	34
3.3.5	Performance Comparison	36
4	Software Simulation	37
5	FPGA Implementation	39
5.1	Software Development Environment	39
5.2	Practical Implementation Issues	39
6	Conclusion	42
	Bibliography	44



List of Figures

2.1	Baseband Model of Space-Time Coded System	3
2.2	Encoder for Space-Time Trellis Codes	5
2.3	Trellis for 4-state Space-Time QPSK Code with 2 Antennas	8
3.1	Trellis Example 1	19
3.2	Trellis Example 2	19
3.3	Decoder Architecture fo Viterbi Algorithm	19
3.4	Two-Way Trellis Butterfly	21
3.5	Add-Compare-Select Circuit	21
3.6	Systolic Trace-Back Decoder Architecture	25
3.7	Systolic Trace-Back Decoder Architecture (Continued)	26
3.8	Permutation Network Path History Unit	27
3.9	Memory Organization for K-Pointer Even Trace-Back	27
3.10	Memory Organization for K-Pointer Odd Trace-Back	28
3.11	Memory Organization for One-Pointer Trace-Back	28
3.12	Trace-Forward Architecture	30
3.13	Memory Organization for Trace-Forward	30
3.14	Memory Organization for Pipeline Version of Trace-Forward	31
3.15	Trace-Forward Unit	31
3.16	Modified Register Exchange Architecture	32

3.17	Memory Organization for Modified Register Exchange	32
3.18	Proposed Architecture	35
3.19	Area-Efficient Proposed Architecture	35
5.1	Add-Compare-Select Circuit using Two's Complement Number	41
5.2	Representation of Two's Complement Numbers	41



Chapter 1

Introduction

With the integration of internet and multimedia applications in next generation wireless communication, the demand for faster and higher data rate communication service becomes inevitable. One practical way to increase the capacity of wireless channel is by employing multiple antennas to create multiple-input and multiple-output (MIMO) channel system. The coding technique that employs multiple antennas is space-time coding. Space-time trellis codes, joins the design of error control coding, modulation, transmit and receive diversity, is an effective signalling scheme able to combat the effect of fading in wireless environment. The decoder for such codes employs the Viterbi algorithm to perform maximum likelihood decoding since its encoder consists of shift registers which can be translate to trellis diagram as convolutional codes. The decoder architecture consist of branch metric, add-compare-select, path metric memory and survivor memory Units. Although the add-compare-select unit is the main component that performs the Viterbi algorithm, the survivor memory unit requires more attention because it occupies most of the decoder chip area, power consumption and is a critical data-processing component that can affect the speed of decoder. There are five chapters in this thesis. The first chapter introduces the motive and outlines for this thesis. The second chapter gives overview of space-time coded systems which includes encoder structure for space-time trellis codes, performance analysis and design criteria for space-time codes. The third chapter first

introduces the Viterbi algorithm and its decoder architecture. Since the main subject of this thesis is on survivor memory architectures, several architectures are discussed in detail. We introduce the two traditional survivor memory architectures, Register Exchange and Trace-Back. Then two VLSI design of trace-back architecture, three other pipelined trace-back architectures and two hybrid survivor memory architectures are discussed. And finally we proposed one architecture for further improvement. The fourth chapter is on software simulation. The fifth chapter discusses the software development environment, design flow and practical implementation issues in FPGA implementation. The last chapter is our conclusion for this thesis.



Chapter 2

Space-Time Coded System

Consider a baseband space-time coded communication system with n_T transmit antennas and n_R receive antennas as figure shown below. The transmitted data are encoded by a space-time encoder. At each time instant t , a block of m binary information symbols denoted by

$$c_t = (c_t^1, c_t^2, \dots, c_t^m) \quad (2.1)$$

is fed into the space-time encoder. The space-time encoder maps the block of m binary input data into n_T modulation symbols from a signal set of $M = 2^m$ points. The coded data are applied to a serial-to-parallel (S/P) converter producing a sequence of n_T parallel symbols, arranged into an $n_t \times 1$ column vector

$$x_t = (x_t^1, x_t^2, \dots, x_t^{n_T})^T \quad (2.2)$$

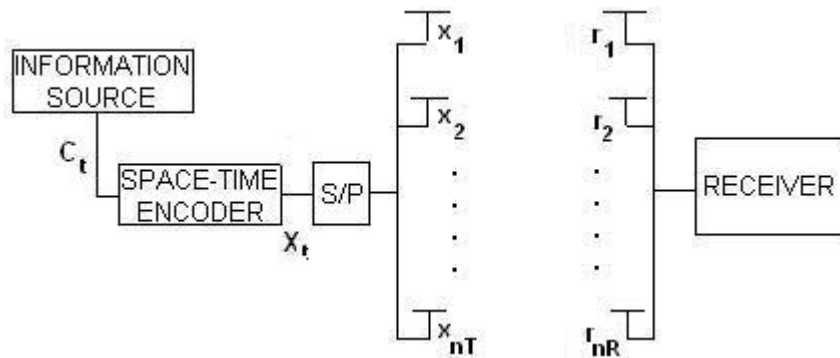


Figure 2.1: Baseband Model of Space-Time Coded System

where T means the transpose of a matrix. The n_T parallel outputs are simultaneously transmitted by n_T different antennas, whereby symbol $x_t^i, 1 \leq i \leq n_T$, is transmitted by antenna i and all transmitted symbols have the same duration of $Tsec$. The vector of coded modulation symbols from different antennas, as shown in figure, is called a *space-time symbol*. The multiple antennas at both transmitter and receiver side forms a MIMO channel, each link from a transmit antenna to a receive antenna can be modeled by flat fading, if the channel is assumed to be memoryless. The MIMO channel with n_T transmit and n_R receive antennas can be represented by an $(n_R \times n_T)$ channel matrix H . At time t , the channel matrix is given by

$$H_t = \begin{bmatrix} h_{1,1}^t & h_{1,2}^t & \dots & h_{1,n_T}^t \\ h_{2,1}^t & h_{2,2}^t & \dots & h_{2,n_T}^t \\ \vdots & \vdots & \ddots & \vdots \\ h_{n_R,1}^t & h_{n_R,2}^t & \dots & h_{n_R,n_T}^t \end{bmatrix} \quad (2.3)$$

The ji -th element, denoted by $h_{j,i}^t$, is the fading attenuation coefficient for the path from transmit antenna i to receive antenna j . The fading coefficients $h_{j,i}^t$ are assumed to be independent complex Gaussian random variables with mean $\mu_h^{j,i}$ and variance $1/2$ per dimension. This implies that the amplitude of the path coefficients are modeled as Rician fading. We also assumed the channel *quasi-static fading*, this means that the variation speed of fading coefficients are constant during a frame and vary from one frame to another. At the receiver, the signal at each of the n_R receive antennas is a noisy superposition of the n_T transmitted signals degraded by channel fading. At time t , the received signal at antenna $j, j = 1, 2, \dots, n_R$, denoted by r_t^j , is given by

$$r_t^j = \sum_{i=1}^{n_T} h_{j,i}^t x_t^i + n_t^j \quad (2.4)$$

where n_t^j is the noise component of receive antenna j at time t , which is an independent sample of the zero-mean complex Gaussian random variable with one sided power spectral density of N_0 . Let us represent the received signals from n_R receive antennas at time t by

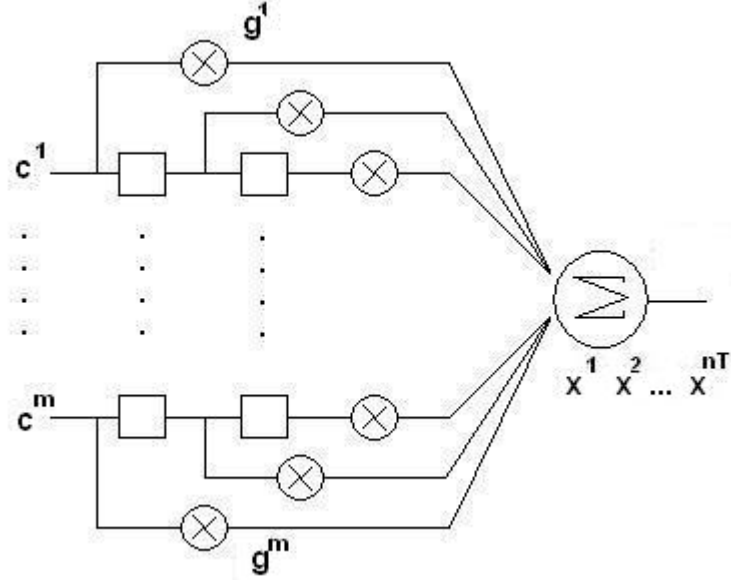


Figure 2.2: Encoder for Space-Time Trellis Codes

an $n_R \times 1$ column vector.

$$r_t = (r_t^1, r_t^2, \dots, r_t^{n_R})^T \quad (2.5)$$

The noise at the receiver can be described by an $n_R \times 1$ column vector, denoted by n_t

$$n_t = (n_t^1, n_t^2, \dots, n_t^{n_R}) \quad (2.6)$$

where each component refers to a sample of the noise at the receive antenna. Thus, received signal vector can be represented as

$$r_t = H_t x_t + n_t \quad (2.7)$$

2.1 Space-Time Trellis Encoder Structure

Space-time trellis encoder maps binary data to modulation symbols, where the mapping function is described by a trellis diagram. Let us consider an encoder of space-time trellis coded M-PSK modulation with n_T transmit antennas as shown in figure below. The input message stream, denoted by c is given by

$$c = (c_0, c_1, c_2, \dots, c_t, \dots) \quad (2.8)$$

where c_t is a group of $m = \log_2 M$ information bits at time t by

$$c_t = (c_t^1, c_t^2, \dots, c_t^m) \quad (2.9)$$

The encoder maps the input sequence into an M-PSK modulated signal sequence, which is given by

$$x = (x_0, x_1, x_2, \dots, x_t, \dots) \quad (2.10)$$

where x_t is a space-time symbol at time t and given by

$$x_t = (x_t^1, x_t^2, \dots, x_t^{n_T}) \quad (2.11)$$

The modulated signals, $x_t^1, x_t^2, \dots, x_t^{n_T}$, are transmitted simultaneously through n_T transmit antennas. The m binary sequence c^1, c^2, \dots, c^m are fed into the decoder, which consist of m feedforward shift registers. The k -th input sequence $c^k = (c_0^k, c_1^k, c_2^k, \dots, c_t^k, \dots)$, $k = 1, 2, \dots, m$, is passed to the k -th shift register and multiplied by an encoder coefficient set. The multiplier outputs from all shift registers are added modulo M , giving the encoder output $x = (x^1, x^2, \dots, x^{n_T})$. The connections between the shift register elements and the modulo M adder can be described by the following m multiplication coefficient set sequences

$$\begin{aligned} g^1 &= [(g_{0,1}^1, g_{0,2}^1, \dots, g_{0,n_T}^1), (g_{1,1}^1, g_{1,2}^1, \dots, g_{1,n_T}^1), \dots, (g_{v_1,1}^1, g_{v_1,2}^1, \dots, g_{v_1,n_T}^1)] \\ g^2 &= [(g_{0,1}^2, g_{0,2}^2, \dots, g_{0,n_T}^2), (g_{1,1}^2, g_{1,2}^2, \dots, g_{1,n_T}^2), \dots, (g_{v_1,1}^2, g_{v_1,2}^2, \dots, g_{v_1,n_T}^2)] \\ &\quad \vdots \\ g^m &= [(g_{0,1}^m, g_{0,2}^m, \dots, g_{0,n_T}^m), (g_{1,1}^m, g_{1,2}^m, \dots, g_{1,n_T}^m), \dots, (g_{v_1,1}^m, g_{v_1,2}^m, \dots, g_{v_1,n_T}^m)] \end{aligned} \quad (2.12)$$

where $g_{j,i}^k$, $k = 1, 2, \dots, m$, $j = 1, 2, \dots, v_k$, $i = 1, 2, \dots, n_T$, is an element of the M-PSK constellation set, and v_k is the memory order of the k -th shift register. The encoder output at time t for transmit antenna i , denoted by x_t^i , can be computed as

$$x_t^i = \sum_{k=1}^m \sum_{j=0}^{v_k} g_{j,i}^k c_{t-j}^k \text{mod} M, i = 1, 2, \dots, n_T \quad (2.13)$$

These outputs are elements of an M-PSK signal set. Modulated signals form the space-time symbol transmitted at time t

$$x_T = (x_t^1, x_t^2, \dots, x_t^{n_T})^T \quad (2.14)$$

The space-time trellis coded M-PSK can achieve a bandwidth efficiency of m bits/s/Hz.

The total memory order of the encoder, denoted by v , is given by

$$v = \sum_{k=1}^m v_k \quad (2.15)$$

where v_k , $k = 1, 2, \dots, m$, is the memory order for the k -th encoder branch. The total number of state for the trellis encoder is 2^v . The m multiplication coefficient set sequences are also called the *generator sequences*, since they can fully describe the encoder structure.

For example, let us consider a simple space-time trellis coded QPSK with two transmit antennas. The encoder consists of two feedforward shift registers. The encoder structure for the scheme with memory order of v is shown in figure. Two binary input stream $c^1 = (c_0^1, c_1^1, \dots, c_t^1, \dots)$ and $c^2 = (c_0^2, c_1^2, \dots, c_t^2, \dots)$ are fed into the upper and lower registers. The memory orders of the upper and lower encoder registers are v_1 and v_2 , respectively, where $v = v_1 + v_2$. The two input streams are delayed and multiplied by the coefficient pairs

$$g^1 = [(g_{0,1}^1, g_{0,2}^1, \dots, g_{0,n_T}^1), (g_{1,1}^1, g_{1,2}^1, \dots, g_{1,n_T}^1), \dots, (g_{v_1,1}^1, g_{v_1,2}^1, \dots, g_{v_1,n_T}^1)] \quad (2.16)$$

$$g^2 = [(g_{0,1}^2, g_{0,2}^2, \dots, g_{0,n_T}^2), (g_{1,1}^2, g_{1,2}^2, \dots, g_{1,n_T}^2), \dots, (g_{v_2,1}^2, g_{v_2,2}^2, \dots, g_{v_2,n_T}^2)] \quad (2.17)$$

respectively, where $g_{j,i}^k, 0, 1, 2, 3, k = 1, 2; i = 1, 2; j = 0, 1, \dots, v_k$. The multiplier outputs are added modulo 4, giving the output

$$x_t^i = \sum_{k=1}^2 \sum_{j=0}^{v_k} g_{j,i}^k c_{t-j}^k \text{mod} 4, i = 1, 2 \quad (2.18)$$

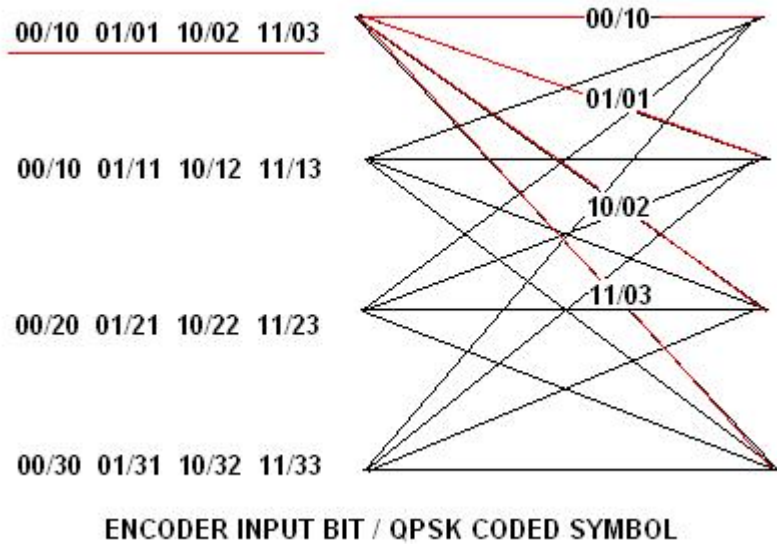
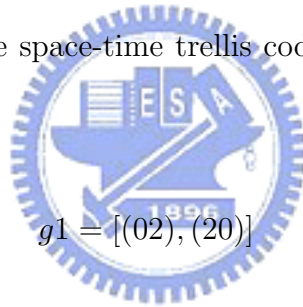


Figure 2.3: Trellis for 4-state Space-Time QPSK Code with 2 Antennas

The adder output x_t^1 and x_t^2 are points from a QPSK constellation. They are transmitted simultaneously through the first and second antenna, respectively. Let us assume that the generator sequences of a 4-state space-time trellis coded QPSK scheme with 2 transmit antennas are



$$g1 = [(02), (20)] \quad (2.19)$$

$$g2 = [(01), (10)] \quad (2.20)$$

The trellis structure for the code is shown in figure above. The trellis consists of $2^v = 4$ states, represented by state nodes. The encoder takes $m = 2$ bits as its input at each time. There are $2^m = 4$ branches leaving from each state corresponding to four different input patterns. Each branch is labelled by $c_t^1 c_t^2 / x_t^1 x_t^2$, where c_t^1 and c_t^2 are a pair of encoder input bits, and x_t^1 and x_t^2 represent two coded QPSK symbols transmitted through antennas 1 and 2, respectively. The row listed next to a state node in figure below indicates the branch labels for transitions from that state corresponding to the encoder inputs 00, 01,

10, and 11, respectively. Assume that the input sequence is

$$c = (10, 01, 11, 00, 01, \dots) \quad (2.21)$$

The output sequence generated by the space-time trellis encoder is given by

$$x = (02, 21, 13, 30, 01, \dots) \quad (2.22)$$

The transmitted signal sequences from the two transmit antennas are

$$x^1 = (0, 2, 1, 3, 0, \dots) \quad (2.23)$$

$$x^2 = (2, 1, 3, 0, 1, \dots) \quad (2.24)$$

Note that this example is actually a delay diversity scheme since the signal sequence transmitted from the first antenna is a delayed version of the signal sequence from the second antennas. For STTC, the decoder employs the Viterbi algorithm to perform maximum likelihood decoding. Assuming that perfect CSI is available at the receiver, for a branch labelled by $(x_t^1, x_t^2, \dots, x_t^{n_T})$, the branch metric is computed as the squared Euclidean distance between the hypothesised received symbols and the actual received signal as

$$\sum_{j=1}^{n_R} |r_t^j - \sum_{i=1}^{n_T} h_{j,i}^t x_t^i|^2 \quad (2.25)$$

The Viterbi algorithm selects the path with the minimum path metric as the decoded sequence.

2.2 Performance Analysis of Space-Time Codes

In the performance analysis we assume that the transmitted data frame length is L symbols for each antenna. We define an $n_T \times L$ *space – time codeword matrix*, obtained by

arranging the transmitted sequence in an array, as

$$X = [x_1, x_2, \dots, x_L] = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_L^1 \\ x_1^2 & x_2^2 & \dots & x_L^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n_T} & x_2^{n_T} & \dots & x_L^{n_T} \end{bmatrix} \quad (2.26)$$

where the i -th row $x^i = [x_1^i, x_2^i, \dots, x_L^i]$ is the data sequence transmitted from the i -th transmit antenna, and the i -th column $x_t = [x_t^1, x_t^2, \dots, x_t^{n_T}]^T$ is the space-time symbol at time t . The pairwise error probability $P(X, \hat{X})$ is the probability that the decoder selects as its estimate an erroneous sequence $\hat{X} = (\hat{X}_1, \hat{X}_2, \dots, \hat{X}_L)$ when the transmitted sequence was in fact $X = (x_1, x_2, \dots, x_L)$. In maximum likelihood decoding, this occurs if

$$\sum_{t=1}^L \sum_{j=1}^{n_R} |r_t^j - \sum_{i=1}^{n_T} h_{j,i}^t x_t^i|^2 \geq \sum_{t=1}^L \sum_{j=1}^{n_R} |r_t^j - \sum_{i=1}^{n_T} h_{j,i}^t \hat{x}_t^i|^2 \quad (2.27)$$

Assuming that ideal CSI is available at the receiver, for a given realization of the fading variable matrix sequence $H = (H_1, H_2, \dots, H_L)$, the term on the right hand side of is a constant equal to $d_h^2(X, \hat{X})$ and the term of the left hand side of is a zero-mean Gaussian random variable. $d_h^2(X, \hat{X})$ is a modified Euclidean distance between the two space-time codeword matrices X and \hat{X} , given by

$$d_h^2(X, \hat{X}) = |H(\hat{X} - X)|^2 = \sum_{t=1}^L \sum_{j=1}^{n_R} \left| \sum_{i=1}^{n_T} h_{j,i}^t (\hat{x}_t^i - x_t^i) \right|^2 \quad (2.28)$$

The pairwise error probability conditioned on H is given by

$$P(X, \hat{X}|H) = Q\left(\sqrt{\frac{E_s}{2N_0}} d_h^2(X, \hat{X})\right) \quad (2.29)$$

where E_s is the energy per symbol at each transmit antenna. By using the inequality

$$Q(x) \leq \frac{1}{2} e^{-\frac{x^2}{2}}, x \geq 0 \quad (2.30)$$

the conditional pairwise error probability can be upper bounded by

$$P(X, \hat{X}|H) \leq \frac{1}{2} \exp\left(-d_h^2(X, \hat{X}) \frac{E_s}{4N_0}\right) \quad (2.31)$$

On slow fading channels, the fading coefficients within each frame are constant. So we can ignore the superscript of the fading coefficients $h_{j,i}^1 = h_{j,i}^2 = \dots = h_{j,i}^L$; $i = 1, 2, \dots, n_T$, $j = 1, 2, \dots, n_R$. Let us define a *codeword difference matrix* $B(X, \hat{X})$ as

$$B(X, \hat{X}) = X - \hat{X} = \begin{bmatrix} x_1^1 - \hat{x}_1^1 & x_2^1 - \hat{x}_2^1 & \dots & x_L^1 - \hat{x}_L^1 \\ x_1^2 - \hat{x}_1^2 & x_2^2 - \hat{x}_2^2 & \dots & x_L^2 - \hat{x}_L^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n_T} - \hat{x}_1^{n_T} & x_2^{n_T} - \hat{x}_2^{n_T} & \dots & x_L^{n_T} - \hat{x}_L^{n_T} \end{bmatrix} \quad (2.32)$$

We can construct an $n_T \times n_T$ *codeword distance matrix* $A(X, \hat{X})$, defined as

$$A(X, \hat{X}) = B(X, \hat{X})B^H(X, \hat{X}) \quad (2.33)$$

where H denotes the Hermitian (transpose conjugate) of a matrix. It is clear that $A(X, \hat{x})$ is nonnegative definite Hermitian, as $A(X, \hat{X}) = A^H(X, \hat{X})$ and the eigenvalues of $A(X, \hat{X})$ are nonnegative real number. Therefore, there exist a unitary matrix V and a real diagonal matrix δ such that

$$VA(X, \hat{X})V^H = \Delta \quad (2.34)$$

The rows of V , v_1, v_2, \dots, v_{n_T} , are the eigenvectors of $A(X, \hat{X})$, forming a complete orthonormal basis of an N -dimensional vector space. The diagonal element of Δ are the eigenvalues $\lambda \geq 0$, $i = 1, 2, \dots, n_T$ of $A(X, \hat{X})$. The diagonal matrix Δ can be represented as

$$\Delta = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{n_T} \end{bmatrix} \quad (2.35)$$

For simplicity, we assume that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{n_T} \geq 0$. Next let

$$h_j = (h_{j,1}, h_{j,2}, \dots, h_{j,n_T}) \quad (2.36)$$

Equation can be rewritten as

$$d_h^2(X, \hat{X}) = \sum_{j=1}^{n_R} h_j A(X, \hat{X}) h_j^H = \sum_{j=1}^{n_R} \sum_{i=1}^{n_T} \lambda_i |\beta_{j,i}|^2 \quad (2.37)$$

where

$$\beta_{j,i} = h_j \cdot v_i \quad (2.38)$$

and \cdot denotes the inner product of complex vectors. Substituting into, we get

$$P(X, \hat{X}|H) \leq \frac{1}{2} \exp\left(-\sum_{j=1}^{n_R} \sum_{i=1}^{n_T} \lambda |\beta_{j,i}|^2 \frac{E_s}{4N_0}\right) \quad (2.39)$$

Inequality above is an upper bound of the conditional pairwise error probability expressed as a function of $|\beta_{j,i}|$, which is contingent upon $h_{j,i}$. For a large value of rn_R ($rn_R \geq 4$), which corresponds to a large number of independent subchannels, according to the central limit theorem, the expression

$$\sum_{j=1}^{n_R} \sum_{i=1}^{n_T} \lambda |\beta_{j,i}|^2 \quad (2.40)$$

approaches a Gaussian random variable D with the mean

$$\mu_D = \sum_{j=1}^{n_R} \sum_{i=1}^{n_T} \lambda_i (1 + K^{j,i}) \quad (2.41)$$

and the variance

$$\alpha_D^2 = \sum_{j=1}^{n_R} \sum_{i=1}^{n_T} \lambda_i^2 (1 + 2K^{j,i}) \quad (2.42)$$

The pairwise error probability can then be upper-bounded by

$$P(X, \hat{X}) \leq \int_{D=0}^{+\infty} \frac{1}{2} \exp\left(-\frac{E_s}{4N_0} D\right) p(D) dD \quad (2.43)$$

where $p(D)$ is the pdf of the Gaussian random variable D . Using the equation

$$\int_{D=0}^{+\infty} \exp(-\gamma D) p(D) dD = \exp\left(\frac{1}{2} \gamma^2 \alpha_D^2 - \frac{E_s}{4N_0} \mu_D\right) Q\left(\frac{E_s}{4N_0} \alpha - \frac{\mu_D}{\alpha_D}\right) \quad (2.44)$$

the upper bound can be further expressed as

$$P(X, \hat{X}) \leq \exp\left(\frac{1}{2} \left(\frac{E_s}{4N_0}\right)^2 \alpha_D^2 - \frac{E_s}{4N_0} \mu_D\right) Q\left(\frac{E_s}{4N_0} \alpha_D - \frac{\mu_D}{\alpha_D}\right) \quad (2.45)$$

Let us now consider the special case of Rayleigh fading. In the case $\mu_a^{j,i} = 0$ and thus $K^{j,i} = 0$. The mean and the variance of the Gaussian random variable D becomes

$$\mu_D = n_R \sum_{i=1}^r \lambda_i \quad (2.46)$$

and

$$\alpha_D^2 = n_R \sum_{i=1}^r \lambda_i^2 \quad (2.47)$$

respectively. Substituting and into, we obtain the pairwise error probability upper bound on Rayleigh fading channels as

$$P(X, \hat{X}) \leq \frac{1}{2} \exp\left(\frac{1}{2} \left(\frac{E_s}{4N_0}\right)^2 n_R \sum_{i=1}^r \lambda_i^2 - \frac{E_s}{4N_0} \sum_{i=1}^r \lambda_i\right) \cdot Q\left(\frac{E_s}{4N_0} \sqrt{n_R \sum_{i=1}^r \lambda_i^2 - \frac{\sqrt{n_R} \sum_{i=1}^r \lambda_i}{\sqrt{\sum_{i=1}^r \lambda_i^2}}}\right) \quad (2.48)$$

When the number of independent subchannels rn_R is small, e.g. $rn_R \leq 4$, the pairwise error probability is upper bounded by

$$P(X, \hat{X}) \leq \prod_{j=1}^{n_R} \left(\prod_{i=1}^{n_T} \frac{1}{1 + \frac{E_s}{4N_0} \lambda_i} \exp\left(-\frac{K^{j,i} \frac{E_s}{4N_0} \lambda_i}{1 + \frac{E_s}{4N_0} \lambda_i}\right) \right) \quad (2.49)$$

In the case of Rayleigh fading, the upper bound of the pairwise error probability becomes

$$P(X, \hat{X}) \leq \left(\prod_{i=1}^{n_T} \frac{1}{1 + \frac{E_s}{4N_0} \lambda_i} \right)^{n_R} \quad (2.50)$$

At high SNR's the above upper bound can be simplified as

$$P(X, \hat{X}) \leq \left(\prod_{i=1}^r \lambda_i \right)^{-n_R} \left(\frac{E_s}{4N_0} \right)^{-n_R} \quad (2.51)$$

where r denotes the rank of matrix $A(X, \hat{X})$, and $\lambda_1, \lambda_2, \dots, \lambda_r$ are the nonzero eigenvalues of matrix $A(X, \hat{X})$. Using a union bound technique, we can compute an upper bound of the code frame error probability, which sums the contributions of the pairwise error probabilities over all error events. Notes that the pairwise error probability in decreases

exponentially with the increasing SNR. The frame error probability at high SNR's is dominated by the pairwise error probability with the minimum product rn_R . The exponent of the SNR term, rn_R , is called the *diversity gain* and

$$G_c = \frac{(\lambda_1 \lambda_2 \dots \lambda_r)^{\frac{1}{r}}}{d_u^2} \quad (2.52)$$

is called the *coding gain*, where d_u^2 is the squared Euclidean distance of the reference uncoded system. Note that both diversity and coding gains are obtained as the minimum rn_R and $(\lambda_1 \lambda_2 \dots \lambda_r)^{\frac{1}{r}}$ over all pairs of distinct codewords. The diversity gain is an approximate measure of the power gain of the system with space diversity over the system without diversity at the same probability. The diversity gain determines the slope of an error rate curve plotted as a function of SNR, while the coding gain determines the horizontal shift of the uncoded system error rate curve to the space-time coded error rate curve obtained for the same diversity order. In general, to minimize the error probability, it is desirable to make both diversity and coding gain as large as possible. Since the diversity gain is an exponent in the error probability upper bound, it is clear that achieving a large diversity gain is more important than achieving a high coding gain for system with small value of rn_R .

2.3 Space-Time Code Design Criterias

As the error performance upper bound indicate, the design criteria for slow Rayleigh fading channels depend on the value of rn_R . The maximum possible value of rn_R is $n_T n_R$. For small values of $n_T n_R$, corresponding to a small number of independent subchannels, the error probability at high SNR's is dominated by the minimum rank r of matrix $A(X, \hat{X})$ over all possible codeword pairs. The product of the minimum rank and the number of receive antennas, rn_R , is called the minimum diversity. In addition, in order to minimize the error probability, the minimum product of nonzero eigenvalues, $\prod_{i=1}^r \lambda_i$, of matrix

$A(X, \hat{X})$ along the pairs of codewords with the minimum rank should be maximized. Therefore, if the value of $n_T n_R$ is small, the space-time code design criteria for slow Rayleigh fading channels is summarized in following two criteria sets. Note that $\prod_{i=1}^r \lambda_i$ is the absolute value of the sum of determinants of all the principal $r \times r$ cofactors of matrix $A(X, \hat{X})$. This criteria set is referred to as *rank & determinant criteria*. It is also called Tarokh/Seshadri/Calderbank (TSC) criteria. The minimum rank of matrix $A(X, \hat{X})$ over all pairs of distinct codewords is called the minimum rank of space-time code. To maximize the minimum rank r means to find a space-time code with the full rank of matrix $A(X, \hat{X})$, e.g., $r = n_T$. However, the full rank is not always achievable due to the restriction of the code structure. For large values of $n_T n_R$, corresponding to a large number of independent subchannels, the pairwise error probability is upper-bounded. In order to get an insight into the code design for systems of practical interest, we assume that the space-time code operates at a reasonably high SNR, which can be presented as

$$\frac{E_s}{4N_0} \geq \frac{\sum_{i=1}^r \lambda_i}{\sum_{i=1}^r \lambda_i^2} \quad (2.53)$$

By using the complementary function inequality, the bound can be further approximated as

$$P(X, \hat{X}) \leq \frac{1}{4} \exp\left(-n_R \frac{E_s}{4N_0} \sum_{i=1}^r \lambda_i\right) \quad (2.54)$$

The bound shows that the error probability is dominated by the codewords with the minimum sum of the eigenvalues of $A(X, \hat{X})$. In order to minimize the error probability, the minimum sum of all eigenvalues of matrix $A(X, \hat{X})$ among all the pairs of distinct codewords should be maximized. For a square matrix the sum of all the eigenvalues is equal to the sum of all the elements on the matrix main diagonal, which is called the *trace* of the matrix. It can be expressed as

$$\text{tr}(A(X, \hat{X})) = \sum_{i=1}^r \lambda_i = \sum_{i=1}^{n_T} A^{i,i} \quad (2.55)$$

where $A^{i,i}$ are the element of the main diagonal of matrix $A(X, \hat{X})$. Since

$$A^{i,j} = \sum_{t=1}^L (x_t^i - \hat{x}_t^i)(x_t^j - \hat{x}_t^j)^* \quad (2.56)$$

substitute into, we get

$$\text{tr}(A(X, \hat{X})) = \sum_{i=1}^{n_T} \sum_{t=1}^L |x_t^i - \hat{x}_t^i|^2 \quad (2.57)$$

Equation above indicates that the trace of matrix $A(X, \hat{X})$ is equivalent to the squared Euclidean distance between the codewords X and \hat{X} . Therefore, maximizing the minimum sum of all eigenvalues of matrix $A(X, \hat{X})$ among the pairs of distinct codewords, or the minimum trace of matrix $A(X, \hat{X})$, is equivalent to maximizing the minimum Euclidean distance between all pairs of distinct codewords. This design criterion is called the *trace criterion*. It should be pointed out that formula is valid for a large number of independent subchannels under the condition that the minimum value of rn_R is high. In this case, the space-time code design criteria for slow fading channels can be summarized as following subsection. It is important to note that the proposed design criteria are consistent with those for trellis codes over fading channels with a large number of diversity branches. A large number of diversity branches reduces the effect of fading and consequently, the channel approaches an AWGN model. Therefore, the trellis code design criteria derived for AWGN channels, which is maximizing the minimum code Euclidean distance, applying to fading channels with a large number of diversity. In a similar way, in space-time code design, when the number of independent subchannels rn_R is large, the channel converges to an AWGN channel. Thus the code design is the same as that for AWGN channels. From the above discussion, we can conclude that either the rank & determinant criteria or the trace criterion should be applied for design of space-time codes, depending on the diversity order rn_R . When $rn_R \leq 4$, the rank & determinant criteria should be applied and when $rn_R \geq 4$, the trace criterion should be applied. The boundary value of rn_R

between the two design criteria sets was chosen to be 4. This boundary is determined by the required number of random variables rn_R in to satisfy the central limit theorem. In general, for random variables with smooth pdf's, the central limit theorem can be applied if the number of random variables in the sum is larger than 4. In the application of the central limit theorem in, the choice of 4 as the boundary has been further justified by the code design and performance simulation, as it was found that as long as $rn_R \geq 4$, the best codes based on the trace criterion outperform the best codes based on the rank and determinant criteria.



Chapter 3

Decoder Architectures for Space-Time Trellis Codes

This chapter divides into three sections. The first section introduces the Viterbi algorithm and its general decoder architecture. The second section discusses main component that performs the Viterbi algorithm, the add-compare-select architectures. The third section introduces two traditional survivor memory unit architectures, Register Exchange and Trace-Back. Then two VLSI design for trace-back architectures, Systolic Trace-Back and Trace-Back on Permutation Network are discussed. Other pipelined trace-back architectures such as K -pointer Even/Odd and One-pointer Trace-Back are also discussed. The two hybrid survivor memory architectures, Trace-Forward and Modified Register Exchange, combine the advantages of both Register Exchange and Trace-Back, are the current research direction in this field and are also discussed. And finally, we discuss how the proposed architecture can improve both latency and chip area compare with conventional architectures.

3.1 The Viterbi Algorithm and Its Decoder Architecture

In the Viterbi algorithm we use trellis diagrams for the computation of path metrics. We have labeled the branches of our trellis with the output bits corresponding to a particular

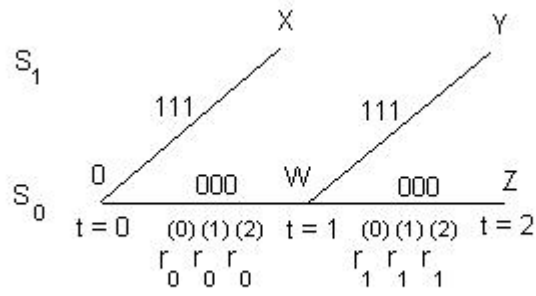


Figure 3.1: Trellis Example 1

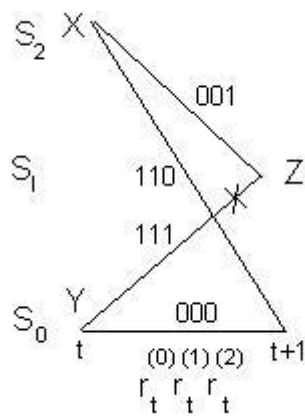


Figure 3.2: Trellis Example 2

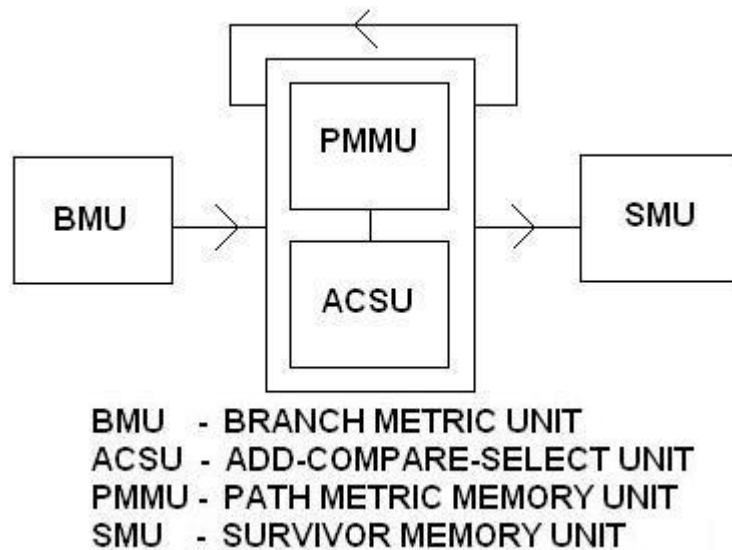


Figure 3.3: Decoder Architecture fo Viterbi Algorithm

input to the encoder and the encoder's current state. Now assume that we have a received sequence r . r is written at the bottom of the trellis one block at a time, with each block corresponding to a trellis stage. For example, if we are working with the code whose trellis is shown in figure below, the beginning of the trellis would be as shown in the accompanying drawing. In the Viterbi algorithm, each node in the trellis is assigned a number. This number is the partial path metric of the path that starts at state S_0 at time $t=0$ and terminates at that node. For example, in the accompanying drawing the label Y corresponds to the two-branch path that terminates at state S_1 at time $t=2$. Given that the output bits corresponding to this path consist of three zeros followed by three ones, we have

$$\begin{aligned}
 Y = M^2(r|y' = (000, 111, \dots)) &= W + M(r_1|y'_1 = (111)) \\
 &= (M(r_0^{(0)}|0) + M(r_0^{(1)}|0)) + M(r_0^{(2)}|0) \\
 &\quad + (M(r_1^{(0)}|1) + M(r_1^{(1)}|1) + M(r_1^{(2)}|1))
 \end{aligned} \tag{3.1}$$

The assignment of numbers to trellis is routine until we reach the point in the trellis in which more than one path enters each node. In this case we choose as the node label the "best" (smallest) partial path metric among the metrics for all entering paths. The path with best metric is the survivor, while the other entering paths are nonsurvivors. If the best metric is shared by two or more paths, the survivor is selected from among the best paths at random. The algorithm terminates when all of the nodes in the trellis have been labeled and their entering survivors determined. We then go to the last node in the trellis (state S_0 at time $L+m$) and trace back through the trellis. At any given node, we can only continue backward on a path that survived upon entry into that node. Since each node has only one entering survivor, our trace-back operation always yield a unique path. This is the maximum likelihood path.

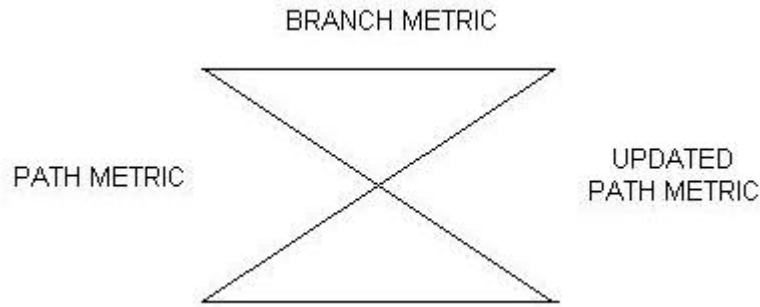


Figure 3.4: Two-Way Trellis Butterfly

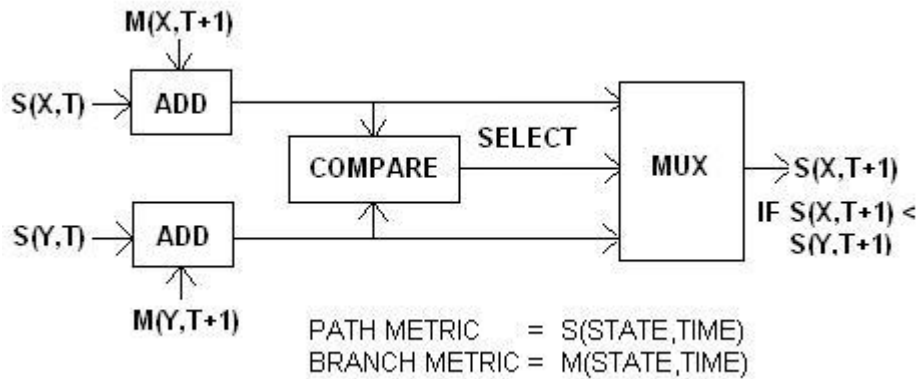


Figure 3.5: Add-Compare-Select Circuit

3.2 Add-Compare-Select Architectures

To update partial path metric at each node efficiently, the trellis can be broken up into a number of identical elements contain a pair of origin and destination states and inter-connecting branches as shown in figure below. Since the entire trellis is multiple images of the same simple element, we can design a single circuit that can be used repeatedly in our decoder. Figure above shows such circuit. The add-compare-select (ACS) circuit performs the basic function of adding partial path metric with branch metrics twice, compare both results and select the least one. The entire decoder can be based on one such ACS, resulting in a very slow but low-cost implementation. On the other hand, a separate ACS circuit can be dedicated to every element in the trellis, resulting in a fast, massively parallel implementation.

3.3 Survivor Memory Architectures

During decoding process, the decision bits of every state must be stored continuously until the transmitted information sequence terminates. But for hardware implementation, there is usually limited memory available, therefore we need to truncate the decisions bits coming from the ACS units. Forney's work indicated that the minimum truncation length for store the decision bits is approximately five to six times of the constraint length of the encoder. This make sure that the survivor path of every state will eventually merged to one state (we called this the merge state). There are two traditional survivor memory management architectures, Register Exchange (RE) and Trace-Back (TB) method. RE assigns a register to every state in the trellis diagram. The content of each register is exchanged (or copied) to another register according to the decision bit written into the survivor path. When the size of register exceeds minimum truncation length (T), the first decision bit written is the decoded bit. Because every state need a register and multiplexer, when the constraint length of encoder increases (the number of states increases), this can result large chip area, wiring complexity and higher power consumption. The advantage of RE is that, because trace-back operation is not required, therefore the latency will be much smaller. TB also assigns register to every state in the trellis. But the decision bits stored are the bits that lost during state transition. Every decision bit stored in each state will be used with current state number to find previous state number. When trace-back to the first decision bits, we can find the merge state for decode data. The disadvantage of TB is large latency due to time for trace-back operation and re-order decoded data in LIFO (Last-In-First-Out) buffer. The advantages, chip area and power consumption are clearly less than RE. One problem for conventional TB is that, in order to decode one bit, we need to trace-back T bits. The following two sections discuss how this problem is

tackled.

3.3.1 VLSI Design of Trace-Back Architectures

The VLSI design for trace-back architectures discussed in this section are Systolic Trace-Back, and Trace-Back on Permutation Networks. The advantage of using a systolic array decoder is that the trace-back operation can be accomplished continuously in an array of registers in a pipe-line fashion, instead of waiting for the entire trace-back procedure to be completed at each iteration. The operation of systolic trace-back decoder is shown in figure below and work as follows. At time unit 1, the selection unit compute decision vector $y_1 = [0, 0, X, X]^T$, then data y_1 is stored in register Y_1 . At time unit 2, the selection unit computes $y_2 = [0, 0, 0, 0]$ and data in register Y_1 is shifted into register Y'_1 . At time unit 3, the selection unit computes $y_3 = [0, 0, 0, 0]$, and data in register Y'_1 is shifted into register Y_2 , and data in register Y_1 is shifted into register Y'_1 and so on. At time unit 10 (assumed to be minimum truncation length T), selection unit select the state with minimum path metric as the merge state of first completed trace-back and is shifted in state register X_1 . After time unit 10, new merge states continuously shifted into state register X_1 and old merge state continuously shift into right state registers. After time unit 20, the decoded bits continuously popped out at the most right-end register. The disadvantage of systolic trace-back is the requirement of huge number of registers as buffer to store all decision vectors and merge states. This causes greater chip area. And because of the continuously data shift operation between registers, this also causes large power consumption. The other VLSI design of trace-back on permutation networks solve large power consumption problem by built fixed trellis consist of multiplexers that can take all decision bits from decoding window and built fixed survivor path and therefore find the merge state in one clock cycle. In Permutation Network Path History Management

(PNPH) Unit, the survivor path in each state is determined by the permutation of decision bit. The trace-back operation begins at the right most end of PNPH unit in an all-path broadcast fashion. Because permutation network is consists of combinational logic circuits, the speed of trace-back operation is simply determined by the propagation of the combination circuits, and usually can be completed in one clock cycle. The goal of permutation network is to mimic the trellis diagram with minimum truncation length T as a large decoding window. Inside the decoding window, each state consists of a 1-to- 2^K de-multiplexer, a K -bit register and a $2K$ input OR gate. The decoding process works as follow, the decision vectors generated from add-compare-select units are inserted into K -bit registers from right-most end of PNPH unit in a pipeline fashion. Because trace-back operation is in an all-path broadcast fashion, inputs of each de-multiplexer are set to one. Since the length of decoding window is minimum truncation length T , by the property of convolutional code, there will be at least one path survived at the left-most end of the PNPH unit. The disadvantage of trace-back on permutation network is similar to register exchange, every state need to connect to both previous and later states, therefore if number of states increases, the area of permutation network also increases. This problem can be improved with several techniques, which will be discussed later.

3.3.2 Pipelined Trace-Back Architectures

In this section we first discuss the pipelined trace-back architectures based on efficient partition on survivor memory, K -Pointer Even/Odd and One-Pointer Trace-Back. K -Pointer Even Trace-Back divides survivor memory to $2K$ blocks, each block has $\frac{T}{K-1}$ columns, and the latency is $\frac{2KT}{K-1}$. K -Pointer Even uses single write pointer and K read pointer to make sure the speed for trace-back and decode read match the speed for write decision vectors. From figure shown above (in this case, $K = 3$), assume that if we want

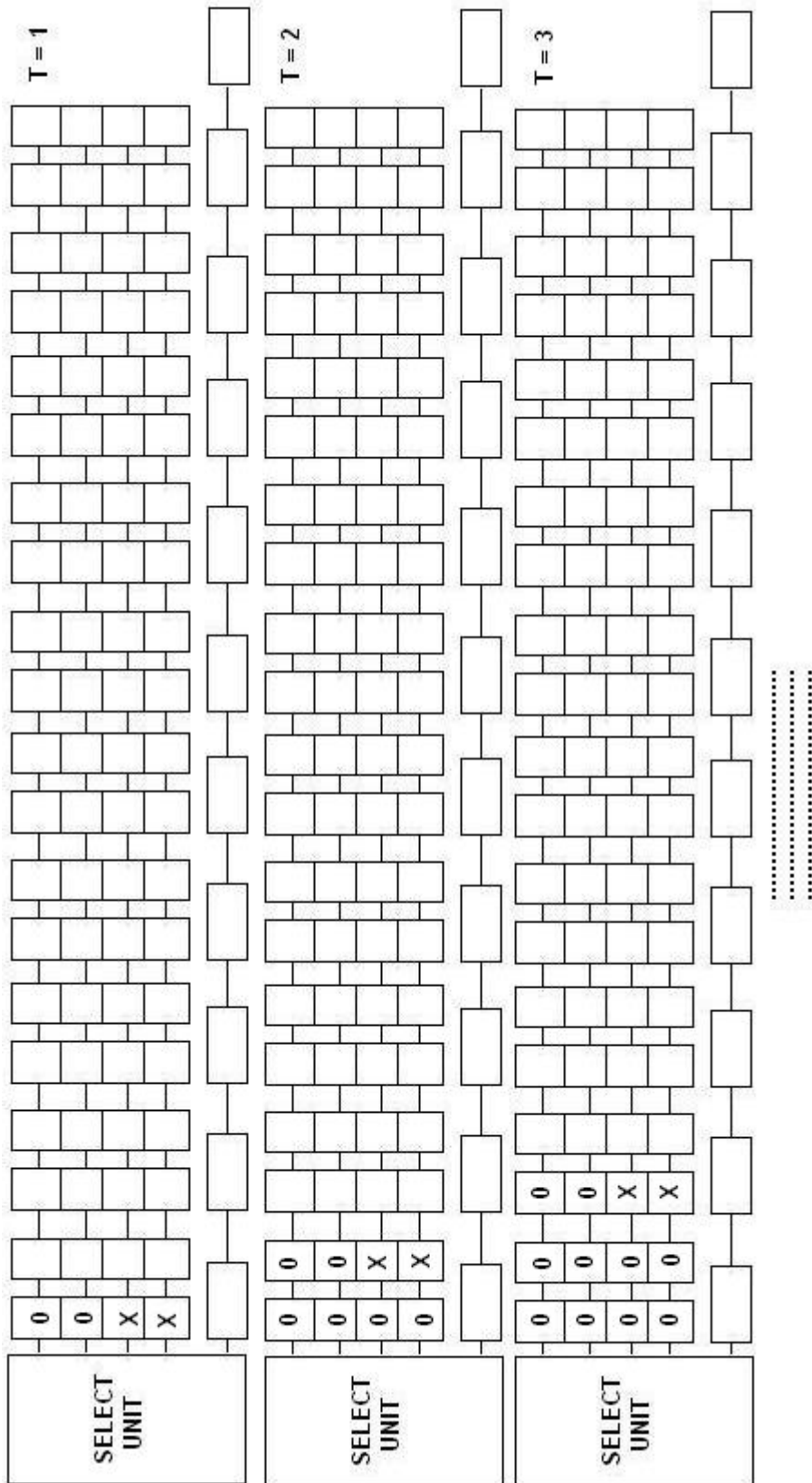


Figure 3.6: Systolic Trace-Back Decoder Architecture

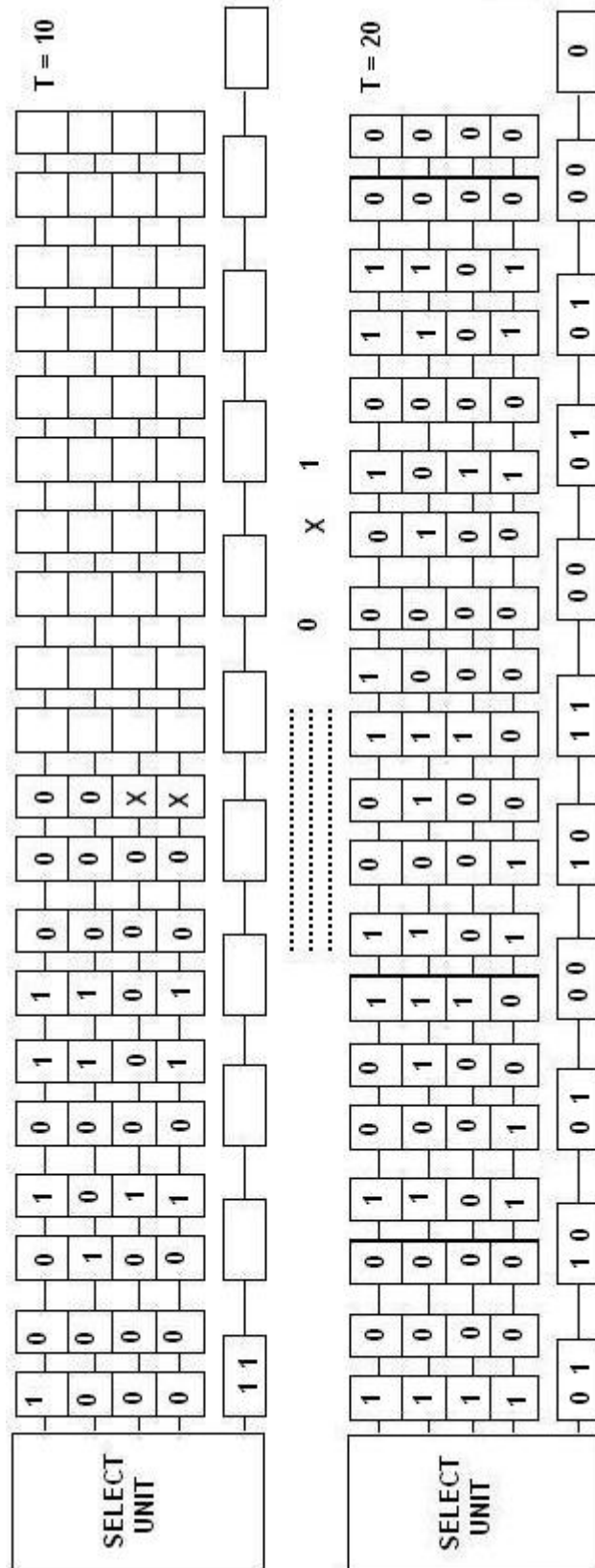


Figure 3.7: Systolic Trace-Back Decoder Architecture (Continued)

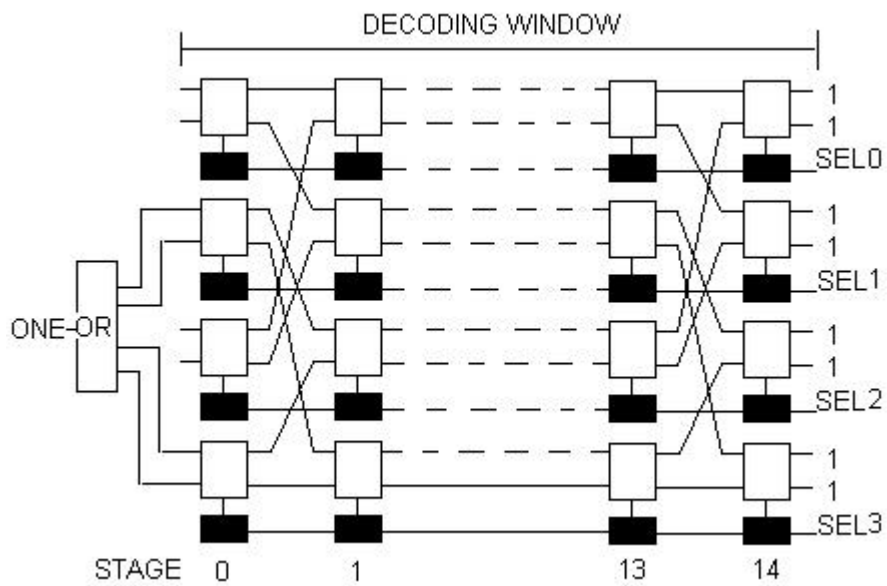


Figure 3.8: Permutation Network Path History Unit

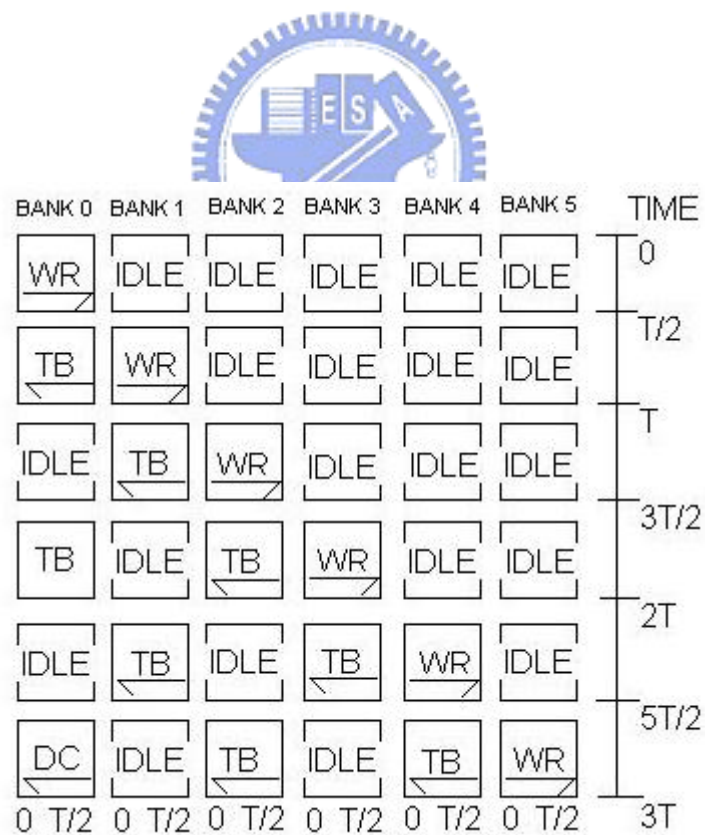


Figure 3.9: Memory Organization for K-Pointer Even Trace-Back

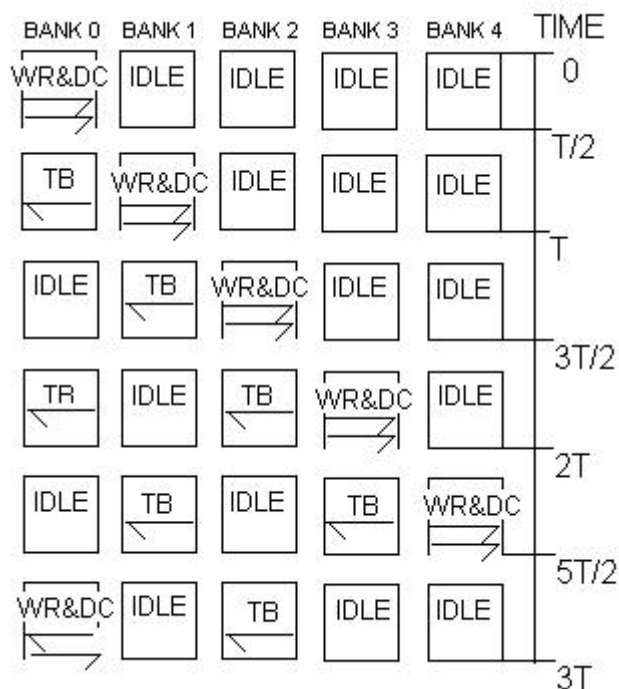


Figure 3.10: Memory Organization for K-Pointer Odd Trace-Back

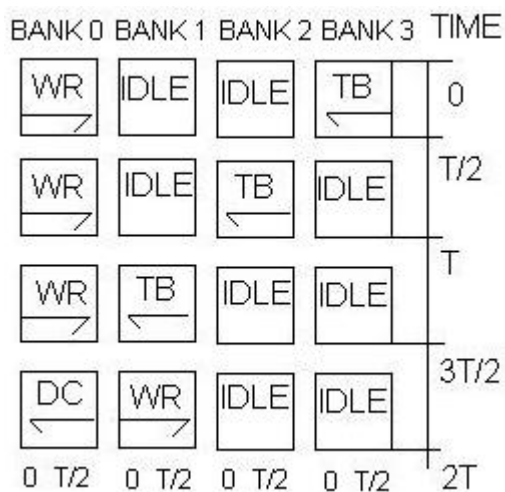


Figure 3.11: Memory Organization for One-Pointer Trace-Back

decode the first block, then we must write decision bits into the first three blocks during $t = 0$ to $\frac{3T}{2}$. After $t = \frac{3T}{2}$, the read pointer start to trace-back the third and second block, and write pointer also continue to write decision bits into fourth and fifth blocks. At $t = \frac{5T}{2}$, when write pointer writes sixth block, the read pointer also start to decode read the first block. K -Pointer Odd divides survivor memory to $2K - 1$ blocks, each block also has $\frac{T}{K-1}$ columns and the latency is also $\frac{2KT}{K-1}$. K -Pointer also uses single write pointer and K read pointer to make sure the speed for trace-back and decode read match the speed for write decision vectors. Although K -Pointer Odd uses one less memory block compare to K -Pointer Even, the control of read pointer is more difficult. K -Pointer Odd needs to assign individual column counter for both write and read pointers. Because the direction for write and read pointers and sometime on opposite direction. The counter design for K -Pointer Even is much easier. One-Pointer Trace-Back divides memory to $K + 1$ blocks, each block has $\frac{T}{K-1}$ columns, and the latency is $\frac{K+1}{k-1}$. Because there is only one read pointer for trace-back and decode read operation, the speed of read pointer must be K times faster than the write pointer. Therefore, when write pointer finish writing one block in memory, the read pointer must also complete trace-back $K - 1$ blocks and decode read one block at the same time.

3.3.3 Hybrid Survivor Memory Architectures

The hybrid survivor memory architectures discussed in this section include Trace-Forward (TF) and Modified Register Exchange (MRE). Before we begin to discuss Trace-Forward, we need to first define what a tail state is. If time m is used as reference, every state at time m will have a survivor path connected to some state at time $m + \Delta$, then we called this the tail state. When Δ is greater than the minimum truncation length T , all the survivor paths will merge to same state. Thus any tail state can be used to

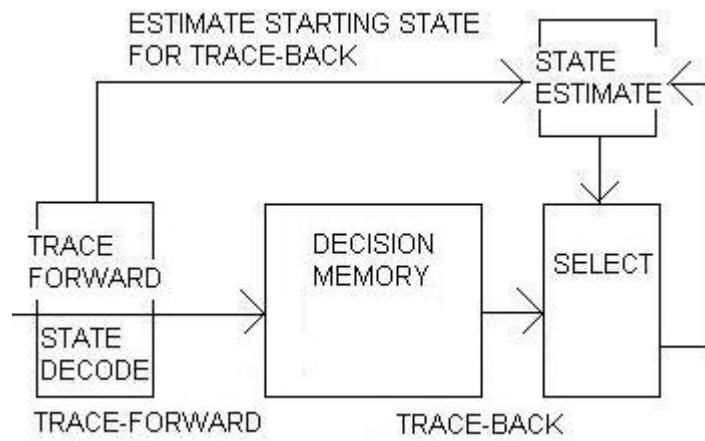


Figure 3.12: Trace-Forward Architecture

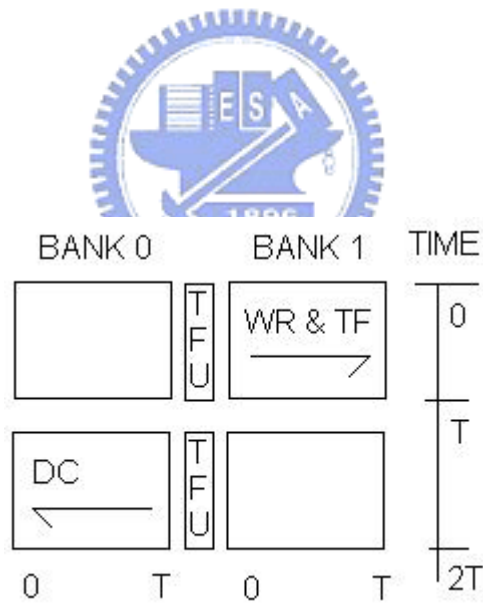


Figure 3.13: Memory Organization for Trace-Forward

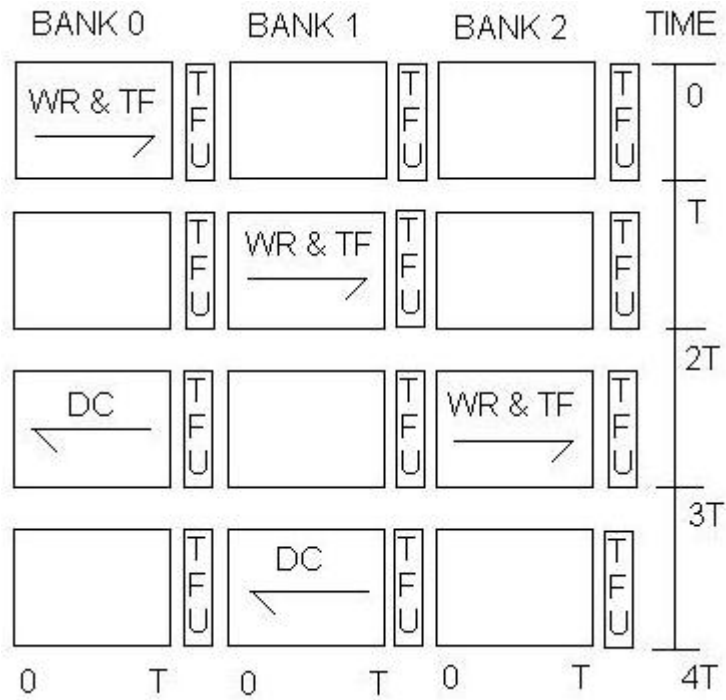


Figure 3.14: Memory Organization for Pipeline Version of Trace-Forward

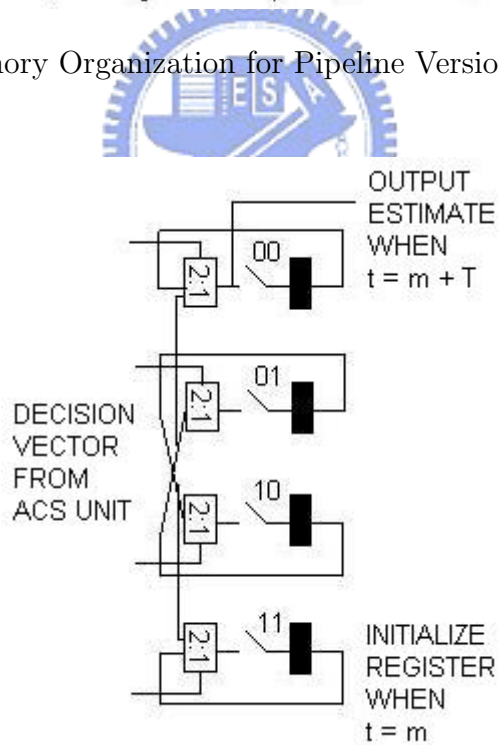


Figure 3.15: Trace-Forward Unit

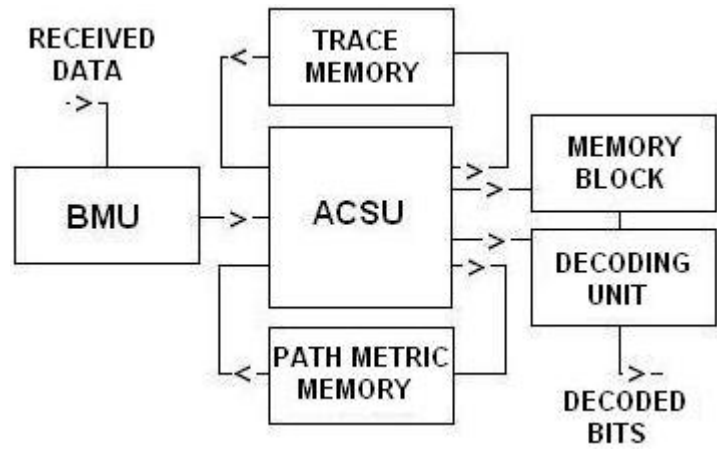


Figure 3.16: Modified Register Exchange Architecture

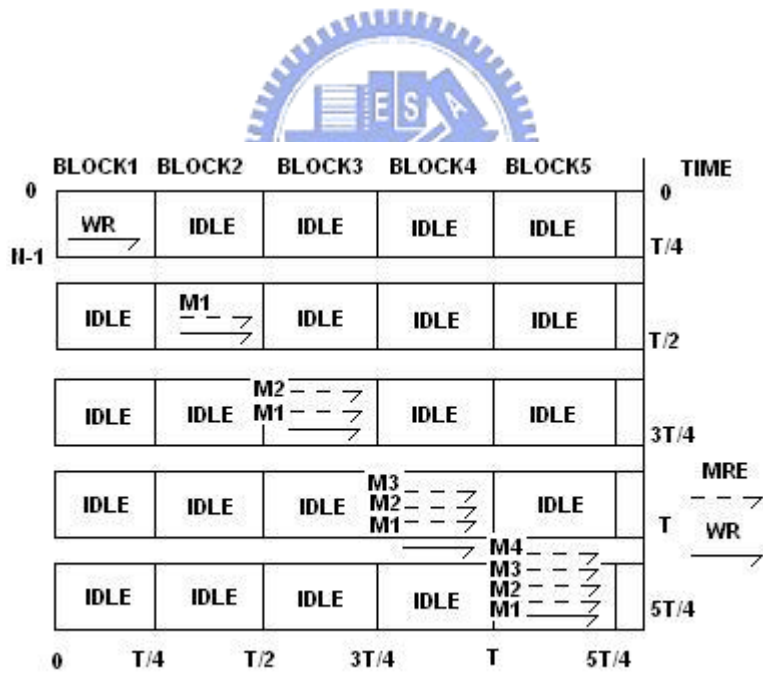


Figure 3.17: Memory Organization for Modified Register Exchange

estimate the merge state at time m . Trace-Forward is a continuous forward procedure for calculate the tail states. Every state uses a register to record its state, the content of the register stored its own state number initially. Then the decision bit received is use to select one of its previous state as tail state and then update in the register. This architecture is similar, except that content of register does not popped out like register exchange. Trace-Forward architecture divides survivor memory to two blocks, write and decode block, each block with length T . When decision vectors write into write block, trace-forward unit is also used to estimate the merge state at the beginning of write block, in order to decode the entire data in the decode block. Figure below shown a four state trace-forward unit, which is consist of multiplexers and registers. The registers initially store their own state number, then the trace-forward unit start to work from time m to $m + \Delta$ to estimate the merge state. The latency and memory size for Hybrid Trace-Forward is $2T$. One disadvantage of trace-forward architecture is that it needs period of $2T$ to output T decode bits. We can improve this by add one extra memory bank of length T . Although the latency becomes $3T$, it will only take period of T to output T decode bits after the latency $3T$. Another disadvantage of trace-forward is that decode read (DC) need to go backward to obtain decode bits from decision bits and need to store in extra LIFO (Last-In First-Out) buffer to obtain correct decode sequence. This can be improved by modified register exchange architecture. The Modified Register Exchange without the trace-back operation stores the decision bits and the state address of the first stage, and their positions are rearranged in the following trace-forward stage. As soon as the trace-back operation finishes at a minimum truncation length T , and the survivor path is determined, and the decision bit and the state that survivor path is pointing become the desired decoded bit and the starting state. The decision memory is divided into several blocks as shown in figure below. Select bits are stored in decision memory

block 1 during the first writing process like k -pointer method. During the second writing process, decision bits are store in decision memory block 2, and the state addresses of the first stage are stored in the trace-memory. During the third writing process, M1 and M2 are processed at the same time, which means the contents of the trace memory 1 and 2 are exchanged simultaneously in the same manner. The starting (or merging) state of the block 2 is determined at the end of the fifth writing process where M1 process is finished after the minimum truncation depth. Now, the decoding process for the decision memory block 1 starts from the merging state. When the decoding bit is determined, it is written to a LIFO (Last-In First-Out) memory and the memory locations that the corresponding select bits have occupied and released. The next trace-forward process (sixth process) can start simultaneously for the decision memory block 1, the merging state for the block 2 is determined, and the decoding process can start continuously. The latency of the decoding process is $\frac{3T}{2}$.

3.3.4 Proposed Architecture

From section 3.1, we found that permutation network trace-back circuit works faster than systolic trace-back although there is the disadvantage of large chip area that needs further improvement. And in section 3.2, we found another way to increase the decoding speed is by adding extra memory bank to deepen the pipeleine. This is used in trace-forward architectures to improve the decoding speed. However, trace-forward still has the disadvantage that require to store and reversed the decision bits during decode read. To further improve the architectures mentioned above, we proposed an architecture that combines modified register exchange with permutation network circuit. The reason is that modified register exchange combines the advantage of RE and TB and outperforms both. And the permutation network circuit is also outperforms the systolic architecutre

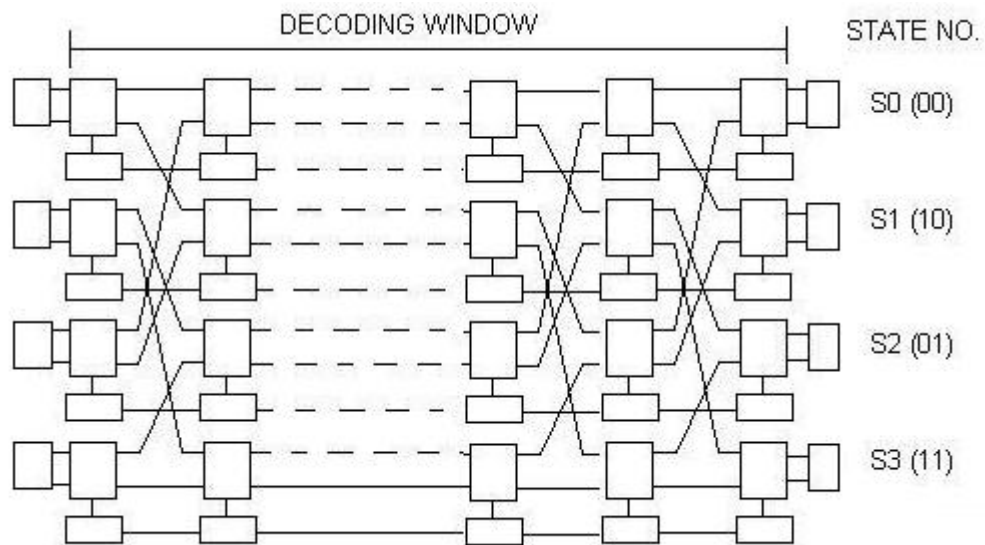


Figure 3.18: Proposed Architecture

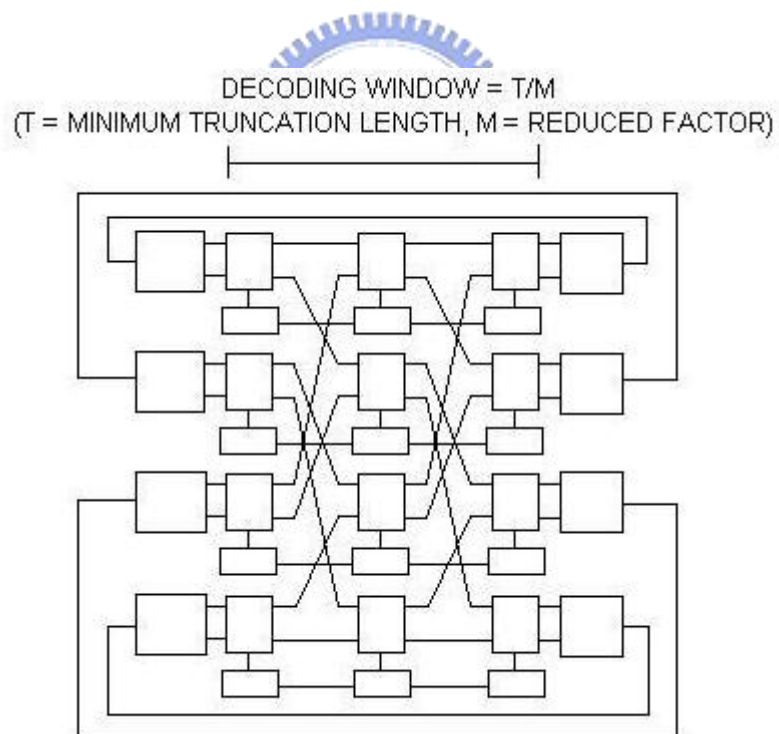


Figure 3.19: Area-Efficient Proposed Architecture

and we found a simple way to reduce the area of permutation networks by add extra buffers to truncate the trellis diagram. The proposed architectures are shown in figures below. The operation of modified register exchange work as follows. The registers below the multiplexers receive decision bits in order to build path for one of the contents of left-hand side side registers to pass the decode bits to right-hand side of the permutation network circuit. To further improve this architecture, we can reduce the area of the permutation network circuit by insert extra registers at both end to temporarily store the result of partial path of the trellis. The area of permutation network can be reduced by factor of $\frac{1}{M}$. The permutation network can now only read $\frac{NT}{M}$ decision bits and take extra M cycles to complete.

3.3.5 Performance Comparison

SMU Architecture	Latency	Memory Requirement
Systolic Trace-Back	$2T$	$2NT$
TB on Permutation Networks	T	NT
K-Pointer Even Trace-Back	$\frac{2K}{K-1}T$	$\frac{2K}{K-1}NT$
One-Pinter Trace-Back	$\frac{K+1}{K-1}T$	$\frac{K+1}{K-1}NT$
Trace-Foward	$2T$	$2NT$
Modified Register Exchange	$\frac{3T}{2}$	$\frac{3NT}{2}$
Proposed Architecture	T	NT

Chapter 4

Software Simulation

Software simulation is important in twofolds. First it can estimate the bit-error-rate (BER) performance of our decoder architecture. And second, it gives us a rough estimation on the sizes of branch metrics, path metrics and sliding window for our hardware implementation. For error correcting code applications, the size of process data is usually very large (more than one million) and therefore will be time-consuming with Matlab simulation. The programming language and software environment we used is C and Microsoft Visual C++. In our software simulation, the encoder we used is space-time trellis QPSK coded with two transmit antennas. The length of sliding window for our decoder implementation is determined to be 20 because any length large does not improve the BER performance, but less than 20 showed BER performance deteriorate. The size of path metrics accumulate to very large values after one million bits have been processed. But, the differences between the path metrics are actually quite small. We decided that 7 to 8 bit is enough to cover the path metric difference. The length of branch metrics are two bits. The software simulation are build up from several smaller functions. These functions include random input sequence generator, additive white gaussian noise generator, and mapper that multiplex input sequence to separate sequences for the two antennas. The trellis function is the main component that contains two-dimension arrays of branch metrics and symbols in constellation. Then several for loops are included, for perform

add-compare-select operation for every states. There are also for loops to output The decision bits stored in two-dimension arrays as decode bits.



Chapter 5

FPGA Implementation

5.1 Software Development Environment

The FPGA board used in our implementation contains the Xilinx's FPGA chip XC2v1000, which contains more than ten million logic gates. The hardware description language (HDL) used for develop our decoder is Verilog. The software tools used is Xilinx ISE 8.1 (Integrated Software Environment) and Modelsim XE (Xilinx Edition). ISE is used for edit, compile and synthesis verilog source code written for construct the decoder. Modelsim is used to view the input and output waveform of the decoder. The design flow of FPGA implementation consist of four steps. The first step is design entry. The second step is synthesis. The third step is placement and route. The last step is verification and programming.

5.2 Practical Implementation Issues

For FPGA implementation, all the parameters in our decoder architecture are represented by binary digits (ones and zeros). Since addition and compare are frequently used in the Virebi algorithm, overflow can also occur frequently as well. Overflow occurs when the number of proper results of addition cannot be represented by those rightmost hardware bits. To resolve this problem, we adopt the two's complement numbering system. The two's complement numbering system uses the most significant bit as sign and other bits

as magnitude for representing binary numbers. This gives the advantage that overflow can be easily detected by check the most significant bit. For our decoder implementation, the path metrics is continuously updated by the add-compare-select unit, therefore it is necessary to normalize the magnitude of path metrics to prevent overflow. The three common methods to normalize path metrics are variable shift, fixed shift and modulo normalization. The third method is used in our decoder, since we adopted the two's complement number system. This method maps all the path metrics onto a circle. This gives the advantage that the differences between the path metrics can still be found even when overflow occurs. The two figures below show the design of add-compare-select circuit in two's complement number and the circular representation of two's complement numbers.



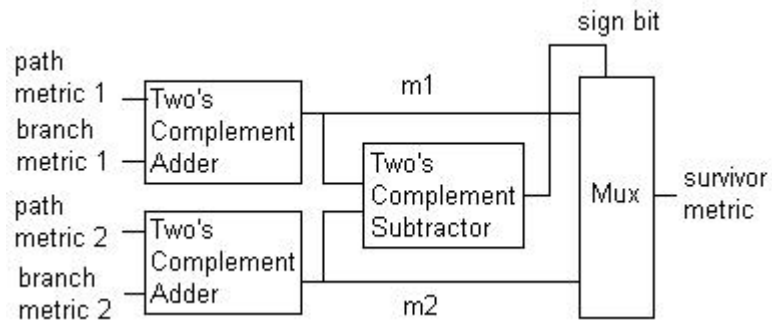


Figure 5.1: Add-Compare-Select Circuit using Two's Complement Number

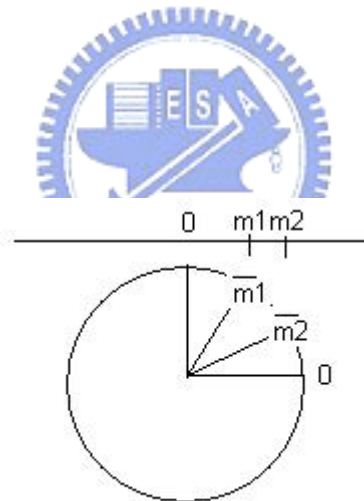


Figure 5.2: Representation of Two's Complement Numbers

Chapter 6

Conclusion

In this thesis, we studied the overview of space-time coded systems which includes the encoder structure for space-time trellis codes, performance analysis and the design criteria for space-time codes. We have also learned the Viterbi algorithm and its decoder architecture. The survivor memory architectures is our main subject in this thesis, we studied several architectures in detail. And finally, we have gathered several ideas and proposed one architecture for further improvement. We found that trace-back on permutation network circuit works faster than systolic trace-back although the disadvantage of large chip area still need further improvement. And we found another method to increase the decoding speed is by adding extra memory bank to deepen the pipeline. We have used this method to improve the decoding speed of trace-forward architectures. However, trace-forward still has the disadvantage that require to store and reverse the decision bits during decode read, which can be quite slow. We further improve the architectures mentioned above by propose an architecture that combines modified register exchange with permutation network circuit. The reason is that modified register exchange has the advantages of both RE and TB and outperformed both. And the permutation network circuit also outperformed the systolic architecture. We also found a simple method to reduce the area of permutation networks by add extra register and truncate the trellis diagram. We choose implement the optimal space-time trellis codes with two transmit

antennas. we have estimated the performance of our decoder from software simulation. And for FPGA implementation, we discussed the software environment, design flow and some important practical issues in FPGA implementation.



Bibliography

- [1] J. G. Proakis, *Digital Communication*, New York: McGraw-Hill, 4rd ed., 2001.
- [2] Stephen B. Wicker, *Error Control Systems for Digital Communication and Storage*, Englewood Cliffs, New Jersey: Prentice-Hall, 1995.
- [3] B. Vucetic and J. Yaun, *Space-Time Coding*, West Sussex, England: John Wiley & Sons, 2003.
- [4] V. Tarokh, N. Seshadri, and A. R. Calderbank, "Space-time codes for high data rate wireless communication: performance criterion and code construction," *IEEE Trans. Inform. Theory*, vol. 44, pp. 744–765, Mar. 1998.
- [5] Charles M. Rader, "Memory Management in a Viterbi Decoder," *IEEE Trans. Commun.*, vol. 29, pp. 1399–1401, Sep. 1981.
- [6] P. Glen Gulak and Thomas Kailath, "Locally Connected VLSI Architectures for the Viterbi Algorithm," *IEEE Journal on Selected Areas in Communications.*, vol. 6, no. 3. pp. 527–537, Apr. 1988.
- [7] Gerhard Fettweis and Heinrich Meyr, "Parallel Viterbi Algorithm Implementation: Breaking the ACS-Bottleneck," *IEEE Trans. Commun.*, vol. 37, no. 8. pp. 785–790, Aug. 1989.
- [8] H A. Bustamante, I. Kang, C. Nguyen and R. E. Peile "Standford Telecom VLSI design of a convolutional decoder," *IEEE Conf. Military Commun.*, vol. 1, Boston, MA, Oct. 1989.
- [9] T. K. Troung, Ming-Tang Shih, Irving S. Reed and E. H. Satorius, "A VLSI Design for a Trace-Back Viterbi Decoder," *IEEE Trans. Commun.*, vol. 40, no. 3. pp. 616–624, Aug. 1992.
- [10] Gennady Feygin, P. G. Gulak, "Architectural Tradeoffs for Survivor Sequence Memory Management in Viterbi Decoders," *IEEE Trans. Commun.*, vol. 41, pp. 425–429, Mar. 1993.
- [11] C. Bernard Shung, Horng-Dar Lin, Robert Cypher, Paul H. Siegel and Hermant K. Thapar, "Area-Efficient Architectures for the Viterbi Algorithm-Part I: Theory," *IEEE Trans. Commun.*, vol. 41, no. 4. pp. 636–644, Apr. 1993.
- [12] P. J. Black and T. H. Y Meng, "Hybrid survivor path architectures for Viterbi decoders," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pp. 433–436, 1993.

- [13] Montse Boo, Francisco Arguello, Javier D. Bruguera, Ramon Doallo and Emilio L. Zapata, "High-Performance VLSI Architecture for the Viterbi Algorithm," *IEEE Trans. Commun.*, vol. 45, no. 2. pp. 168–176, Feb. 1997.
- [14] Ming-Bo Lin, "New Path History Management Circuits for Viterbi Decoders," *IEEE Trans. Commun.*, vol. 48, pp. 1605–1608, Oct. 2000.
- [15] J S. Han, T J. Kim and Chanhoo Lee, "High Performance Viterbi Decoder Using Modified Register Exchange Methods," in *Proc. IEEE Int. Symp. Circuit and System.*, vol. 3, pp. 553–536, May. 2004.
- [16] Dalia A. El-Dib and Mohamed I. Elmasry, "Modified Register-Exchange Viterbi Decoders for Low-Power Wireless Communications," *IEEE Trans. Circuits and Systems-I: Regular Papers.*, vol. 51, pp. 371–378, Feb. 2004.

