A Novel High-Speed Trellis-Coded Modulation Encoder/Decoder ASIC Design

A Thesis Submitted to the College of

Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of Master of Science

in the Department of Electrical Engineering

University of Saskatchewan

Saskatoon

By

Xiao Hu

PERMISSION TO USE

ABSTRACT

Trellis-coded Modulation (TCM) is used in bandlimited communication systems. TCM efficiency improves coding gain by combining modulation and forward error correction coding in one process. In TCM, the bandwidth expansion is not required because it uses the same symbol rate and power spectrum; the differences are the introduction of a redundancy bit and the use of a constellation with double points.

In this thesis, a novel TCM encoder/decoder ASIC chip implementation is presented. This ASIC codec not only increases decoding speed but also reduces hardware complexity. The algorithm and technique are presented for a 16-state convolutional code which is used in standard 256-QAM wireless systems. In the decoder, a Hamming distance is used as a cost function to determine output in the maximum likelihood Viterbi decoder. Using the relationship between the delay states and the path state in the Trellis tree of the code, a pre-calculated Hamming distances are stored in a look-up table. In addition, an output look-up-table is generated to determine the decoder output. This table is established by the two relative delay states in the code. The thesis provides details of the algorithm and the structure of TCM codec chip. Besides using parallel processing, the ASIC implementation also uses pipelining to further increase decoding speed.

The codec was implemented in ASIC using standard 0.18μm CMOS technology; the ASIC core occupied a silicon area of 1.1mm^2. All register transfer level code of the codec was simulated and synthesized. The chip layout was generated and the final chip was fabricated by Taiwan Semiconductor Manufacturing Company through the Canadian Microelectronics Corporation. The functional testing of the fabricated codec was performed partially successful; the timing testing has not been fully accomplished because the chip was not always stable.

ACKNOWLEDGEMENTS

I am deeply grateful to my supervisor, Dr. Anh Dinh, for his guidance, support, and patience during my graduate program. He has been an invaluable source of knowledge and has certainly helped inspire many of the ideas expressed in this thesis. The other special thanks go to the Canadian Microelectronics Corporation for their software and hardware support. This thesis could not have been accomplished without their support.

Also, I would like to thank the Department of Electrical Engineering and College of Graduate Studies and Research, University of Saskatchewan, for providing a graduate scholarship during this research.

Special appreciation goes to my husband, Tie Peng, for his understanding, endless support and help; and my son, Yuwei Peng, for his smile, listening, smart talk and being a good boy.

INDEX OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	Two-dimensional
4D	Four-dimensional
ARC	Antenna Rule Check
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit-Error-Rate
BIST	Built-In-Self-Test
CATV	Cable Television
CMC	Canadian Microelectronics Corporation
CMOS	Complementary Metal Oxide Silicon
CSU	Compare-Selection Unit
DAVIC	Digital Audio Visual Council
dB	Decibel
DBS	Direct Broadcast Satellite
DFF	D-Flip-Flop
DFII	Design Framework II (Cadence design tools)
DFT	Design-For-Test
DRC	Design Rules Check
DSP	Digital Signal Processing
ECC	Error-Correcting Coding

FEC	Forward Error Correction
FPGA	Field Programmable Gate Array
Gbps	Giga-bit per second
GDSII	Graphical Design Stream II
GHz	Gigahertz
HDL	Hardware Description Language
HDLUT	Hamming Distance Look-Up Table
HDTV	High Definition Television
IC	Integrated Circuit
I/O	Input/Output
IP	Intellectual Property
LFSR	Linear Feedback Shift Register
LSB	Least Significant Bit
LUT	Look-Up Table
LVS	Layout-Versus-Schematic
MATLAB	Matrix Laboratory (simulation tools)
Mbps	Mega-bit per second
MHz	Megahertz
MMDS	Multi-channel Multipoint Distribution Services
MSB	Most Significant Bit
MSED	Minimum Squared Euclidean Distance
MUX	Multiplexer
NC-Sim	Cadence® simulator

OLUT	Output Look-Up Table
PDP	Physical Design Planner
PROM	Programmable Read Only Memory
PRPG	Pseudo Random Pattern Generator
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quaternary Phase Shift Keying
RTL	Register Transfer Level
SE	Silicon Ensemble (Cadence layout tools)
SNR	Signal-to-Noise Ratio
SOC	System-On-Chip
STAU	State-Transition and Add Unit
TSMC	Taiwan Semiconductor Manufacturing Company
TCM	Trellis-Coded Modulation
TTL	Transistor-Transistor Logic
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VLSI	Very Large Scale Integration

CHAPTER 1

INTRODUCTION

This chapter introduces the history of coding theory and why Trellis-Coded Modulation is chosen to be the research topic and implemented into an Application Specific Integrated Circuit (ASIC). This chapter also presents the main research methodology, research achievement and the thesis outline.

1.1 Classic Coding

In 1948, Claude E. Shannon established the mathematical foundation for information transmission. Shannon demonstrated that the effect of transmitted power, bandwidth, and additive noise can be associated with a channel and incorporated into the channel capacity. In the case of additive white Gaussian noise interference with power spectral density N_0, an ideal band-limited channel of bandwidth W has a capacity C bits/s given by

$$C = W \log_2 (1 + \frac{P}{WN_0}) \qquad (1.1)$$

where P is the average transmitted power.

The significant meaning of the channel capacity is that if the information rate R from the source is less than C, (R<C), then it is theoretically possible to achieve reliable

(i.e., error-free) transmission through the channel by appropriate coding. On the other hand, if R>C, reliable transmission is not possible regardless of the amount of signal processing performed at the transmitter and receiver.

The information theory shows that for optimal communications, system design should have long sequences of signals, with maximum separation among them at the transmitter. At the receiver, the performing decision should be made over such long signal sequence rather than individual bit. If the process is performed properly then the message error probability, P_e, will decrease exponentially with sequence length n. The condition is that the rate R is less than R_0, which in turn is less than the channel capacity as stated in the information theory. The message error probability can be expressed as:

$$P_e < 2^{-(R_0-R)n} \tag{1.2}$$

where R_0 is a quantity that carries dimensions of bits per channel symbol [1]. In order to increase the sequence length n, error-correcting coding (ECC) is introduced. Error-correcting coding method introduces redundant bits into each symbol, which directly increase the codeword length n and decrease error probability.

In conventional ECC coding, the coding process is independent from the modulation process. Coding occurs at the digital symbol 0's and 1's before modulation and generally involves adding redundant bits into an input sequence. The resultant redundancy requires extra transmission bandwidth. At the receiver, the inversion process of coding is the decoding, which occurs after the signal demodulation. Since a digital bit (or symbol) stream that comes into the decoder is either in error or not, corresponding with detection method, decoding divides into hard decoding and soft decoding. The hard decoding operation is based on hard decisions of the digital signal, which is 0 or 1. The soft decoding is based on soft decision of the analog received samples, which could be any quantized value between the lowest and the highest voltage instead of only 0 and 1. The theoretical loss due to hard versus soft decoding leads to a reduction of 2dB in performance.

Trellis-coded modulation is different from conventional coding; it treats coding and modulation together. The coding process involves handling the mapping a codeword into a modulation scheme.

1.2 Trellis-Coded Modulation

In a power-limited environment, the desired system performance should be achieved with the smallest possible transmitted power. The use of error-correcting codes can increase power efficiency by adding extra bits to the transmitted symbol sequence. This procedure requires the modulator to operate at a higher data rate, which requires a wider bandwidth. In a bandwidth-limited environment, the use of higher-order modulation schemes can increase efficiency in frequency utilization. In this case, a large signal power would be required to maintain the same system bit-error-rate (BER). In order to achieve improved reliability of a digital transmission system without increasing transmitted power or required bandwidth, both coding and modulation are considered in TCM technology; therefore, TCM is a scheme combining error-correcting coding with modulation.

TCM is used for data communication with the purpose of gaining noise immunity over uncoded transmission without changing the data rate. The use of TCM also improves system reliability without increasing transmitting power and required channel bandwidth. Quadrature Amplitude modulation (QAM) and Quaternary Phase Shift Keying (QPSK) are used in TCM to increase data transmission rate. Since channel bandwidth is a function of the signal-to-noise ratio (SNR), larger signal power would be necessary to maintain the same signal separation and the same error probability if more signals are required to be transmitted without enlarging channel bandwidth. It seems that the Trellis code statement violates the basic power, bandwidth and error probability trade-off principle. Actually this is not true; the reason for this will be explained below and as well as in the next chapter.

Trellis coding introduces dependency between every successive transmitting data symbol. The optimum 2-dimensional modulation utilizes the dependency between in-phase and quadrature symbols and the 4-dimensional modulation employs the dependency between symbols of two successive time intervals. Trellis codes and multi-dimensional modulation are designed to maximize the Euclidean distance between possible sequences of transmitted symbols. Euclidean distance is a straight-line distance between two points in signal constellation. In N dimensions, the Euclidean distance between two points p and q is:

$$\sqrt{\sum_{i=1}^{N}(p_i - q_i)^2} \tag{1.3}$$

where p_i and q_i are the coordinate of p and q in the dimension i. The minimum Euclidean distance (i.e., the distance between the closest possible sequences) of transmitted symbols in signal space determines the system performance as:

$$P_e \sim e^{-d_{\min}^2/2\sigma^2} \tag{1.4}$$

where P_e is message error probability, $d_{\min}$ is the minimum Euclidean distance between signal sequences and σ is the noise power. Equation (1.4) indicates that if $d_{\min}$ increases, P_e will decrease. This is one of the reasons why TCM technique does not violate the basic trade-off principle between power, bandwidth, and error probability.

Since Ungerboeck invented TCM in 1976 and had his papers published in the 1980's [2-4], numerous researches have been working on TCM applications in numerous areas: voice band modems, satellite communications, wireless communications trials, digital subscriber loop, HDTV (high definition television), broadcast channels, CATV (community antenna television) and DBS (direct broadcast satellite) in the 1980's and 1990's. Many innovations in TCM technology have been introduced, such as multidimensional TCM (1984-1985), rotationally invariant TCM with M-PSK (1988), TCM with built-in time diversity (1988-1990), TCM with Tomlinson Precoder (1990-1991), TCM with unequal error protection (1990), multilevel coding with TCM (1992-1993), and concatenated coding with TCM (1993-present).

Even a number of researches focused on the Viterbi Algorithm to decode convolutional code ([5-10]) and on trellis coding with multi-dimensional modulation ([11-15]), but only a few was developed in the implementation a TCM encoder/decoder on a single chip. This research focuses on the ASIC implementation of TCM encoder/decoder with 2-dimensional and 4-dimensional modulation mapping, and provides an intellectual property (IP) core to incorporate TCM in system-on-chip (SOC). The designed chip was fabricated by Taiwan Semiconductor Manufacturing Company (TSMC) through a grant from Canadian Microelectronics Corporation (CMC). In order to show the advantages of using CMOSP18 technology to implement the ASIC TCM encoder/decoder chip, the ASIC hardware implementation results are compared with the Field Programmable Gate Array (FPGA) implementation from different aspects such as logic gates, chip area and speed. The followings introduce some background of the FPGA and ASIC.

1.3 FPGA Design

An FPGA is a programmable device. Programmable devices are a class of general purpose integrated circuits that can be configured in the field for a wide variety of applications. FPGAs were invented in the mid-1980s as devices that resemble ASICs but could be programmed after the chip was manufactured. A FPGA consists of a matrix of programmable logic cells with a grid of interconnecting lines and switches between them. I/O cells exist around the perimeter providing an interface between the interconnect lines and the chip external pins. Programming an FPGA consists of specifying the logic function of each cell and the switches in the interconnecting lines.

FPGA devices allow rapid design prototyping. They provide some benefits of custom CMOS VLSI design, while avoiding the initial cost, fabrication time delay, and inherent risk of a conventional masked gate array. The devices are customized by loading configuration data into the internal logic gates and memory cells. The FPGA can either actively read its configuration data from external serial or byte-parallel PROM (i.e., master mode), or the configuration data can be written into the FPGA (i.e., slave and peripheral mode). They offer more dense logic and less tedious wiring work than discrete component designs and faster turnaround than sea-of-gates, standard cells, or full-custom design.

FPGA designs can be programmed using one of the various hardware description languages (HDL), such as Very high speed integrated circuit HDL (VHDL) and Verilog HDL, or directly programmed with logic circuit diagrams and graphical schematic component layout.

1.4 ASIC Design

ASIC design is a process of integrating a specific complicated application into a chip. A typical ASIC design process is generalized into three steps: HDL design capture, HDL design synthesis and design implementation. HDL design capture is completed with “pre-synthesis” simulations to verify that the register transfer level (RTL) abstraction fully provides the desired functionality. HDL design synthesis is completed with “post-synthesis” simulations to verify that the gate-level circuit provides the desired functionality and meets appropriate timing requirements. Design implementation is completed with the physical verification to implement the chip layout under the Design Rules Check (DRC) error free and ready for microelectronic circuit fabrication.

ASIC generally runs faster than FPGA because of its specific place and route during the layout process. ASIC costs less than FPGA and DSP implementations under mass production; therefore it is widely used in industries. ASICs are not economical compared to FPGAs if they are used only for research purposes because ASIC implementations require extra time to develop and the chip can not be re-configured if the design requires further modifications; a new development process has to be started.

An ASIC design can be implemented in different ways depending on the hardware description languages (Verilog or VHDL), synthesis and/or layout tools to be used in the implementation process. Details on ASIC design implementation will be discussed in Chapter 4.

1.5 Research Objectives, Contribution and Methodology

The objective of this research is to design a high speed TCM encoder/decoder and to implement the codec into an ASIC. The results of ASIC implementation are then compared with FPGA implementation in terms of the operating frequency (i.e., system throughput or bit rate) and hardware requirement (i.e., chip cost).

This thesis work focuses on the constructing of a 16-state TCM codec scheme based on DAVIC specification [16], and then implementing it on a single ASIC. The research is accomplished by creating a novel architecture of the TCM decoder and implementing a 16-state TCM codec chip into an ASIC including mapping the signal to both 2-dimensional and 4-dimensional constellations. In the TCM decoder part, the main difference from the traditional technique is the use of look-up tables (LUTs). These tables are used to simplify the cost function calculation in the Viterbi algorithm. The use of these LUTs not only reduces hardware complexity but also increases the decoding speed. Another difference from the traditional decoding technique is the efficient use of the shift registers to perform the tracing back process in the Viterbi algorithm. This technique avoids using a large amount of memory required to store error metrics for all of the paths.

The ASIC consists of two parts: a TCM encoder and a TCM decoder. The TCM encoder starts from sampling digital source data, and ends at the mapping signal generation. The TCM decoder begins with the sampling of the codeword, and finishes with the restoring of the original digital source data. The VLSI implementation of the TCM decoder includes three parts: the de-mapping process, the convolutional decoding and the differential decoding. The main and complicated part is the convolutional decoding process in which the Viterbi algorithm is used. High-speed operation and low complexity of the codec are achieved by creating of the two LUTs, simplifying the Viterbi algorithm for VLSI implementation, and adopting the parallel algorithm and the pipelining technology.

RTL codes were written in VHDL. Simulation was run on the Cadence NC-Sim simulator to verify the functionality of the ASIC. The RTL codes are then synthesized into a gate-level netlist using Synopsys synthesis tools. The gate-level netlist was turned into layout for fabrication using various Cadence tools: Physical Design Planner (PDP or DP), Silicon Ensemble (SE), and Design Framework II (DFII). The final layout was streamed out to GDSII format file to be ready for fabrication. The ASIC was fabricated by TSMC and tested.

1.6 Outline of the Thesis

The rest of this thesis provides detail descriptions on the background, concept and implementation of the TCM codec. Chapter 2 introduces the background and key point of the TCM and illustrates the related algorithm and technology. Chapter 3 provides details of the TCM codec structures, illustrates methodology and implementation process including the mapping for 2D and 4D constellations; creates the architecture for hardware implementation. Chapter 4 provides further description on the hardware implementation, details of this ASIC design process and the structure of this TCM codec chip. Chapter 5 provides the results of the design TCM codec in three parts: (1) the MATLAB simulation on the TCM encoder/decoder algorithm, (2) the VHDL RTL codes compilation and simulation results on FPGA APEX® device, and (3) the ASIC implementation results including the chip layout. Chapter 6 provides conclusion and recommendation for future work.

CHAPTER 2

TRELLIS-CODED MODULATION BACKGROUND

This chapter introduces general error-correcting coding, fundamental and concepts of TCM, general encoder structure, mapping method, algorithm used in the TCM decoding, and the multi-dimensional TCM codec.

2.1 Error-Correcting Coding

Figure 2.1 shows a typical communication system incorporating with channel coding. In this system, digital data generated by a source encoder are coded through a channel encoder, and the coded information data is used to modulate a carrier for transmission over a communication channel. The channel always introduces attenuation, distortion, interference and noise, which affect the receiver's ability to receive correct information. The demodulator recovers possible values of the transmitted symbols, and the channel decoder recovers the information data before sending to its destination.

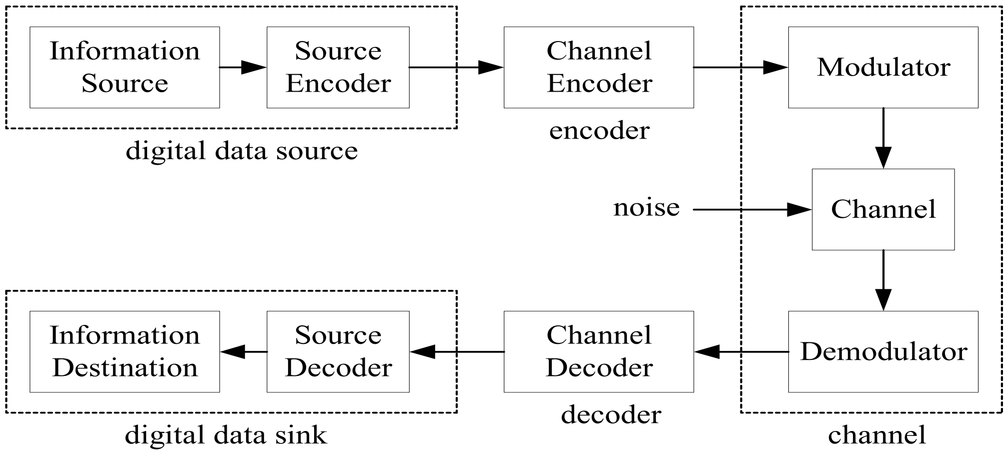


Figure 2.1 Block diagram of a digital-coded system

A widely used channel encoding method is error-correcting coding (ECC). In an ECC system, a digital information source sends a data sequence to an encoder; the encoder inserts redundant (or parity) bits, thereby outputting a longer sequence of the code bits. The sequence of the code bits is called a codeword. The codeword is then transmitted to a receiver, which has a suitable decoder to recover the original data sequence.

Error-correcting coding is related to the method of delivering information from a source to a destination with minimum error. The research on ECC began in 1948 with the publication of a landmark paper by Claude E. Shannon [17]. Shannon’s work showed that any communication channel could be characterized by a capacity at which information could be reliably transmitted. At any rate of information transmission up to the channel capacity, it should be possible to transfer information at any desired level of error rates. Introducing redundancy into transmissions can provide error control. This means more symbols are included in the message than strictly needed just to convey the information, with the result that only certain patterns at the receiver correspond to valid transmissions. Once an adequate degree of error control has been introduced, the error rates can be made as low as required by extending the length of the code, thus averaging the effects of noise over a longer period of time [18-19].

The research for error-correcting codes was primarily motivated by the problems arising in communications systems, in particular, systems having limitation in their transmitted powers. Error-correcting codes are an excellent means of reducing transmission power requirements because reliable communications can be achieved with the aid of codes even when the information is weakly received at its destination.

There are two different types of codes used in error-correcting coding: the block codes and the convolutional codes. The primary point of both codes is to add redundant bits to achieve error correction objectives. These redundant bits are added into each symbol of information sequence formulating the codeword. They will be used in the decoder at the receiver to correctly restore the original information sequence under specific decoding algorithms. These redundant bits are also called parity-check bits; the bits provide the code with the capability of combating channel noise.

The encoder for the block codes divides information sequences into information blocks of k-bit, represented by $d = (d_0, d_1, \ldots, d_{k-1})$ information sequence. The encoder transforms each k-bit information block d independently into a codeword, $c = (c_0, c_1, \ldots, c_{n-1})$. This transformation provides a coding rate R= k/n (R < 1). The code is called a (n, k) block code. Decoding algorithm of the block code is based on the method of constructing the codeword in the encoder.

Figure 2.2 shows how a codeword is generated through a (6,3) block encoder. Each 6-bit output codeword is comprised of the original 3-bits message sequence and a 3-bits parity sequence.

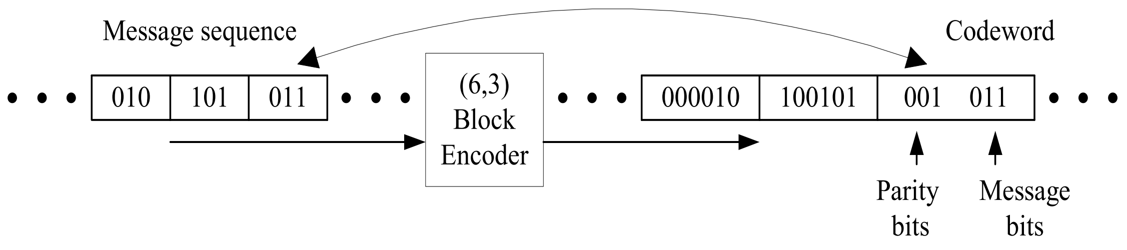


Figure 2.2 An illustration of a (6,3) block code

The encoder for convolutional codes generates codeword for transmission utilizing a sequential finite-state machine driven by the information sequence, or an arbitrarily long sequence. The convolutional encoding process installs the properties of memory and redundancy into the codeword stream, as for block codes. Figure 2.3 shows a generic description of a convolutional encoder.

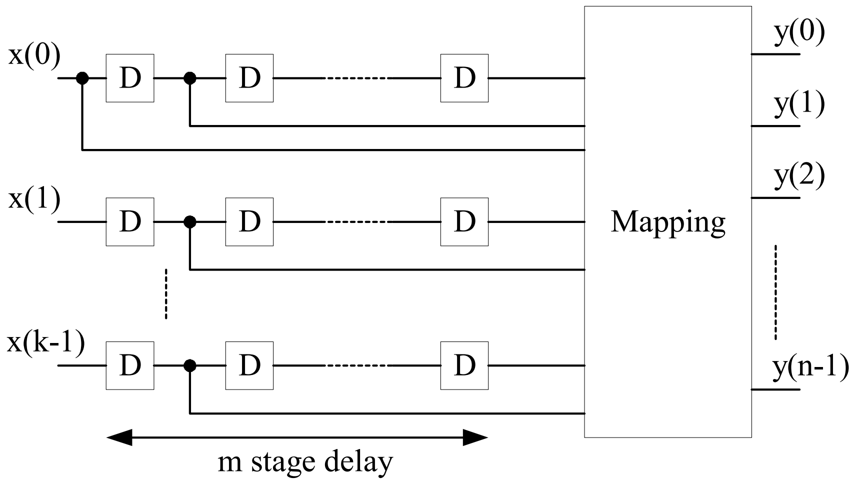


Figure 2.3 An illustration of a convolutional code

The k-bit information sequence as the input of the encoder is coded to the n-bit codeword through the m stage delay (represented by delay cell D). The coding rate is still R= k/n (R < 1). The code is called (n, k, m) convolutional code. Each of k parallel input lines can have different number of stage delay cells, which is smaller than or equal to m ($m \geq 0$).

The most common method used in decoding the convolutional codes is the probabilistic method. The probabilistic decoding method includes sequential decoding (Fano algorithm or stack algorithm) and maximum-likelihood decoding (Viterbi algorithm). The sequential decoding uses a systematic procedure to search for a good estimation of the message sequence; however, such procedure requires a large memory and typically suffers from buffer overflow problems.

Andrew Viterbi developed the Viterbi algorithm in 1967 [20]. Since then, it has become the standard algorithm to decode the convolutional codes. At each time interval, the Viterbi decoding algorithm compares the actual received codeword with the codeword that might have been generated for each possible memory-state transition. The algorithm chooses the most likely sequence within a specific time frame based on the metrics of similarity. The maximum-likelihood decoding requires less memory than the sequential decoding because unlikely sequences are dismissed early, leaving a relatively small number of candidate sequences that need to be stored.

The addition of parity bits in the transmitted sequence eventually increases the transmission bandwidth requirement. In 1976, Ungerboeck invented Trellis-Coded Modulation technique to solve this problem.

2.2 Trellis-Coded Modulation

In communication systems, error-correcting coding (ECC) reduces power utilization (i.e., the ratio of the received energy per bit to the noise spectral density) by adding redundancy to the transmitted signal. The performance improvement of the ECC can be achieved by expanding the bandwidth of the transmitted signal to the code rate in the power-limited region, which requires a high-order modulation scheme. For bandwidth-efficient multilevel amplitude and phase modulation such as PSK or QAM, without expanding the channel bandwidth required by ECC, increasing the number of signal phase or amplitude over the corresponding modulation constellation performs the same data throughput as uncoded modulation. However this increment requires an additional signal power to maintain the same level of system bit-error-rate [21-23]. In communications, Trellis-coded modulation is applied to solve the conflict of utility efficiency between transmission power and channel bandwidth.

The TCM coding process utilizes signal mapping combining error-correcting coding with modulation. The mapping by the set partitioning technique provides a combination of digital signals used in the modulation. This technique increases the minimum Euclidean distance between the pairs of coded signals; hence the loss from the expansion of the signal set is easily overcome and a significant coding gain is achieved with ECC. This is the reason why TCM technique does not violate the basic trade-off principle between power, bandwidth, and error probability. Therefore, in a bandwidth-limited communication system, without bandwidth enlargement, the redundancy bits are introduced into the signal to achieve good performance of coding gain through the TCM codes. This gain can be as high as 5dB.

The widely used ECC code in the TCM encoder is the convolutional code. As mentioned in Section 2.1, the Viterbi decoding algorithm is the standard method to decode convolutional codes, and commonly used in communication systems. In this TCM codec implementation, the TCM decoder design chooses the Viterbi algorithm as its decoding algorithm.

The following subsections describe details of the TCM, such as fundamentals and concepts, standard encoder structures, mapping by set partitioning method and the Viterbi algorithm.

2.2.1 Fundamentals and Concept of TCM

Assume there is a model for the transmission of data with discrete-time, continuous-amplitude over the additive white Gaussian noise (AWGN) channel [24]. In this communication model, messages to be delivered to the user are represented by points, or vectors, in an N-dimensional Euclidean space R^N, called a signal space. When a vector $x = [x_1, x_2, ..., x_i]$ is transmitted, the received signal can be represented by the vector Z as:

$$Z = x + v \tag{2.1}$$

where ν is a noise vector $[v_1, v_2, ..., v_i]$ whose components $v_1, v_2, ..., v_i$ are independent Gaussian random variables with zero mean and the same variance $N_0/2$. N_0 is noise power spectral density. The vector x is chosen from a set, the signal constellation Ω′, which consists the number of M′ signal vectors. The average square length is referred to as the average signal energy and represented by:

$$E' = \frac{1}{M'} \sum_{x \in \Omega'} \|x\|^2 \tag{2.2}$$

If a sequence $\{x_i\}$ of K signals, i = 0, 1, …, K-1 is transmitted, the receiver observes received sequence y_0, …, y_{k-1} and then decides that X_0, …, X_{k-1} was transmitted if the squared Euclidean distance d^2 is minimized for $x_i = X_i$ (i = 0, 1, …, K-1).

$$d^2 = \sum_{i=0}^{K-1} \left\| y_i - x_i \right\|^2 \tag{2.3}$$

That means if the sequence X_0, X_1, …, X_{k-1} is closer to the received sequence than to any other allowable signal vector sequence, then the resulting sequence error probability, as well as the symbol error probability, is upper bounded by

$$P(e) \le \frac{M'-1}{2} erfc(\frac{d_{\min}}{2\sqrt{N_0}}) \tag{2.4}$$

in which the complementary error function *erfc(x)* is defined as $erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$.

The information rate and the normalized squared minimum distance are useful in comparing different constellations. Let *R* represents the information rate, and δ^2 represents normalized squared minimum distance, these two can be defined as:

$$R = \frac{\log_2 M'}{N} \tag{2.5}$$

and $$\delta^2 = \frac{d_{\min}^2}{E'} \log_2 M' \tag{2.6}$$

R is the ratio between the number of information bits carried by a single signal in the constellation and the number of dimensions *N*, M' is the number of signal vectors in constellation Ω′, and E' is the average signal energy.

If M', *N*, and E' are given, the problem of designing a good communication system is choosing a set of vector signals such that the minimum distance between any two signals is maximize. The information rate is also referred to as the bandwidth efficiency of that signal set, and the normalized squared minimum distance is its energy efficiency. Equation (2.4) can be rewritten in the form

$$P(e) \le \frac{M'-1}{2} erfc\left(\frac{\delta}{2}\sqrt{\frac{E_b}{N_0}}\right) \tag{2.7}$$

where $E_b = \dfrac{E'}{\log_2 M'}$ represents the average energy per bit.

In TCM system, signals are dependent. To avoid a reduction of the transmission rate, the constellation should be expanded. The minimum free Euclidean distance d_{free} between two possible sequences in a large constellation is obtained to be greater than the minimum Euclidean distance d_{min} between two signals in the original constellation. This will be analyzed in Section 2.2.3. Hence using maximum-likelihood sequence detection will yield a distance gain $d^2_{free}/d^2_{\min}$. On the other hand, expanding constellation induces an increase in the average energy expenditure from E' to E (i.e., energy loss E/E'), where E' and E are the average energy spent to transmit with uncoded and coded transmission, respectively. The asymptotic coding gain γ of a TCM scheme is

$$\gamma = \frac{d^2_{free}/E}{d^2_{\min}/E'} \tag{2.8}$$

The following example shows the advantage of using TCM [24]. Assume the transmission of quaternary source symbols (i.e., 2 bits per symbol) with uncoded transmission a channel alphabet with M′=4 would be adequate, and PSK is used. Then

$$\frac{d^2_{\min}}{E'} = 2$$

In here, M=2M'=8 are used; M is the number of signal vectors in the new TCM constellation. Figure 2.4 shows the TCM scheme based on two quaternary constellations, {0, 2, 4, 6} and {1, 3, 5, 7}. This figure also shows a two-state Trellis construction.

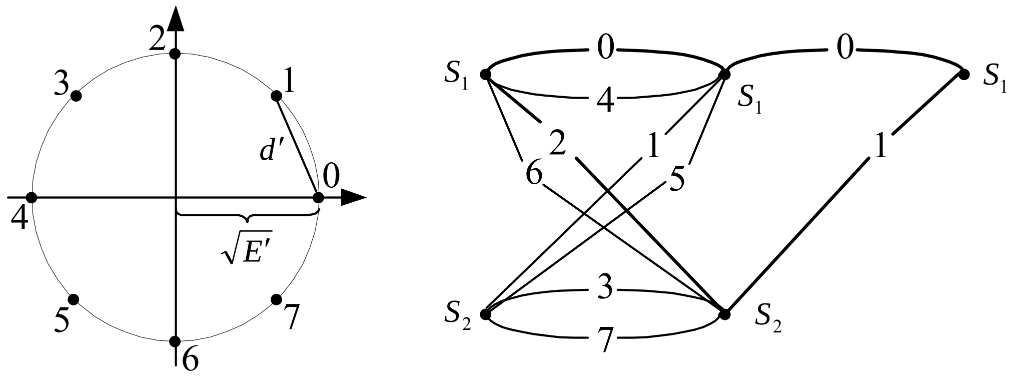


Figure 2.4 The constellation and two-state trellis used in the example TCM scheme

From Figure 2.4, the coding gain of PSK-based TCM can be calculated as:

$$E' = \frac{d'^2}{4\sin^2(\pi/8)}$$

There are distances between signals associated with the parallel transitions, and the distances associated with a pair of paths in the trellis those originate from a common node (i.e., state, S_1 or S_2 in Figure 2.4) and merge into a single node at a later time. The free distance of this TCM scheme is calculated by choosing the smallest distances from them. Then,

$$\frac{d^2_{free}}{E} = \frac{1}{E}[d^2(0,2) + d^2(0,1)] = 2 + 4\sin^2\frac{\pi}{8} = 2.586$$

in which d(i, j) denotes the Euclidean distance between signals i and j, where i, j∈(0, 1… 7). The coding gain of this TCM scheme becomes:

$$\gamma = \frac{d^2_{free}/E}{d^2_{\min}/E'} = \frac{2.586}{2} = 1.293 \Rightarrow 1.12dB$$

By increasing the number of states (i.e., memory bits) in TCM, the coding gain can be increased. For example, if the four-state or eight-state trellis coding is used in this TCM scheme instead of two-state, the coding gain will be 3.01dB and 3.6dB, respectively [25]. Figure 2.5 shows the coding gain tendency with different numbers of the trellis states. At each point on the graph, the second number shows the coding gain can be achieved when the number of states (the first number) is used.

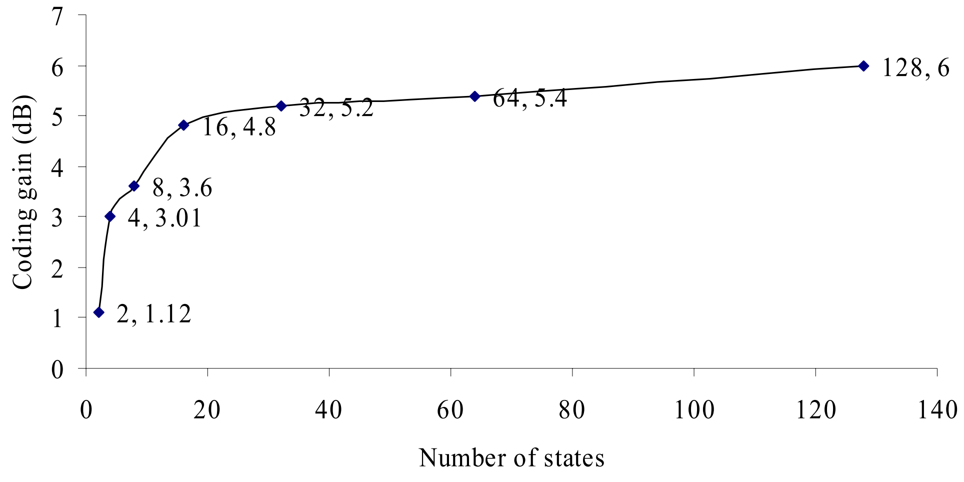


Figure 2.5 Result of coding gain versus the number of states

Increasing the number of states in TCM is a simple way to improve its performance. However, once the number of states grows high enough, the coding gain increases at much slower rate as illustrated in Figure 2.5. In addition, the error coefficient (i.e., the multiplicity of minimum-Euclidean-distance error events) of the code starts to dominate the code performance [11]. In order to solve this conflict, a multi-dimensional constellations model is introduced to the TCM system, which will be described in Section 2.3.

2.2.2 TCM Encoder

Figure 2.6 shows a general 2-dimensional TCM encoder scheme on one time interval. The input symbol consists of n bits data. While two bits (I_1 and I_2) are coded through both differential and convolutional encoder, the other n-2 bits remain uncoded.

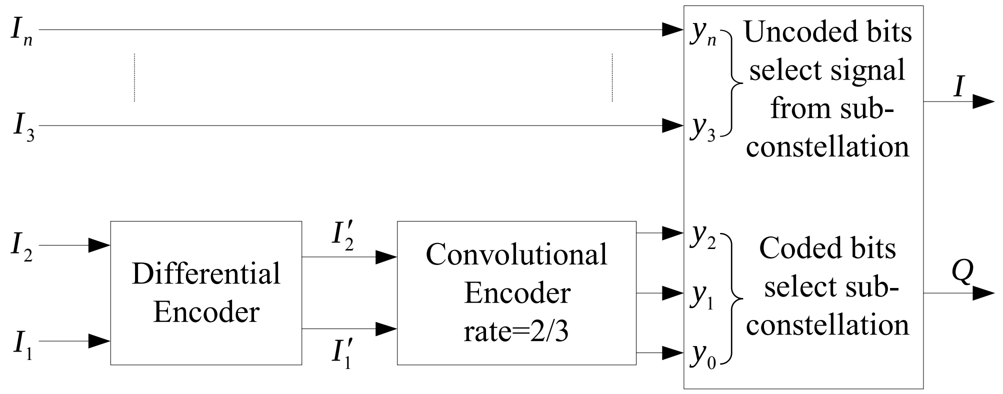

I_n
I_3
I_2
I_1
Differential Encoder
I'_2
I'_1
Convolutional Encoder rate=2/3
y_n
y_3
y_2
y_1
y_0
Uncoded bits select signal from sub-constellation
Coded bits select sub-constellation
I
Q

Figure 2.6 General scheme of a 2D TCM encoder

The differential encoder provides protection against the 180^0 phase ambiguity introduced by the communication channel. The convolutional encoder introduces forward error correction information for the transmitted data. This convolutional encoder is also called a Trellis encoder. This is a method of achieving coding gain by increasing the density of the constellation while keeping the minimum distance between each constellation points the same. It ensures that the transmitted sequence of points conforms to a valid trellis sequence, which is essential for proper decoding in the receiver. The signal mapping is used to convert the convolutional coded bits to an efficiency combination of Quadrature Amplitude Modulation (QAM) mode. QAM is a method to modulate the digital data in an analog signal in which each combination of phase and amplitude represents one of the 2^n n-bit patterns.

As shown in Figure 2.6, without any error correction information, each symbol has n bits, which requires a 2^n-point constellation. After the 2/3 convolutional encoder, each symbol has n+1 bits, which requires a 2^{n+1}-point constellation. In general, for the same average power, a modulation scheme using a 2^{n+1}-point constellation has higher BER if compared with the 2^n-point constellation scheme. The reason for this is that the minimum Euclidean distance between any two points on a 2^{n+1}-point constellation is smaller, which decreases the noise margin. However, convolutional encoding introduces constraints in transforming an n-bit input symbol to a (n+1)-bit output symbol. Specifically, it does not allow two consecutive output symbols to be in the eight neighborhood positions of each other. This results in an increment of the minimum Euclidean distance between two consecutive output symbols, which is achieved through mapping by the set partitioning method. This process provides an overall performance gain of 4dB. Set partitioning is an important process after convolutional encode, which will be illustrated in Section 2.2.3.

As differential encoding is used in this TCM scheme; using this coding, errors caused by the phase reversal in the channel are not allowed to propagate. The receiver will reconstruct the information sequence properly except for the errors at the points where phase reversal occurs.

2.2.3 Set Partitioning

In TCM, the modulation is an integral part of the encoding process. It is designed in conjunction with the code to increase the minimum Euclidean distance between the pairs of the coded signals. The loss from the expansion of the signal set is easily overcome and a significant coding gain is achieved with relatively simple codes. The key to this integrated modulation and coding approach is to devise an effective method for mapping the coded bits into appropriate signal points so that the minimum Euclidean distance is maximized. In 1982, Ungerboeck developed such method, based on the principle of mapping by set partitioning [2-4].

Based on the set partitioning, the M-ary constellation is successively partitioned into 2, 4, 8, …, $2^{(\log_2 M)-1}$ subsets, with size M/2, M/4, M/8, …, 2 with progressively larger minimum distances. The set partitioning method follows three Ungerboeck rules:

- U1: To parallel transitions are assigned members of the same partition;
- U2: To adjacent transitions are assigned members of the next larger partition;
- U3: To make all the signals are used equally often.

A set partitioning method is illustrated in the Figure 2.7. The figure shows a partitioning of the 16-points QAM constellation.

In Figure 2.7, the d_{min}^2 between points in the subsets is increased by at least a factor of 2 with each partitioning. In the first partitioning, the 16-points constellation is subdivided into two 8-point subsets. The square of the minimum Euclidean distance d_{min}^2 increases to $2d^2$ from d^2. In the second partitioning, each of the two 8-point subsets is subdivided into two subsets of 4-point, and the square of the minimum Euclidean distance d_{min}^2 is increased to $4d^2$. This process continues on the subsets until each subset has only two points, and the square of minimum Euclidean distance d_{min}^2 is now increased to $8d^2$.

In QAM constellation, each level of partitioning increases the minimum Euclidean distance by $\sqrt{2}$. The level to which the signal is partitioned depends on the characteristics of the code.

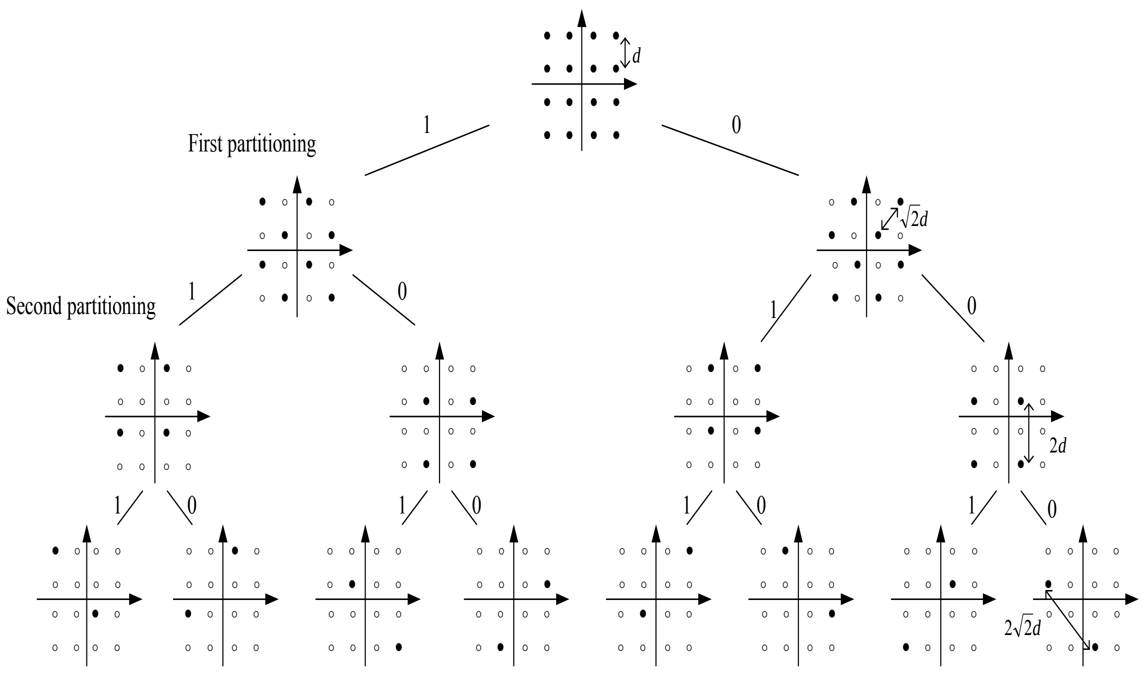


Figure 2.7 Set partition of a 16-QAM constellation

In general, the TCM encoding process is performed as illustrated in Figure 2.6. A block of the n information bits is separated into two groups, one will be coded and the other remains uncoded. The group of coded bits will be used to select one of the possible subsets in the partitioned signal set, while the uncoded bits are used to select the points in each subset. Section 2.2.4 will further explain the mapping based on the set partitioning method. Section 2.3.3 will illustrate the mapping for the multi-dimensional constellation set partitioning.

2.2.4 Mapping and Trellis Diagram

An 8-state, rate 2/3 convolutional encoder is used as an example to analyze the trellis diagram and mapping method. Figure 2.8 illustrates a convolutional encoder used in the V.32 modem on one time interval [26-27]. Input of this TCM encoder is a 4-bit symbol, and the output is a 5-bit symbol. The output includes a 3-bit codeword and a 2-

bit uncoded word. This encoder works at the full baud rate. The baud rate is a measure of the speed of a serial communication using a modem or null-modem, roughly equivalents to bits per second.

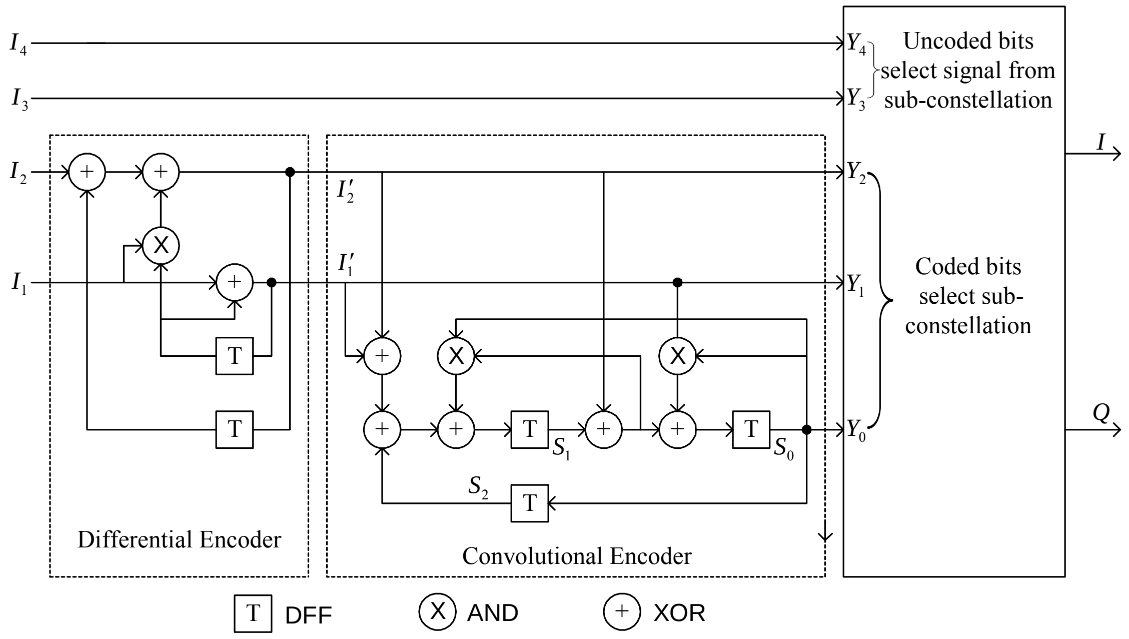


Figure 2.8 Structure of the TCM encoder used in V.32 modem

The encoder operates as follows, the redundant bit Y_0 is functionally obtained from I'_1, I'_2 which are differential coded bits obtained from I_1, I_2, and three memory bits, S_0, S_1, S_2 which interconnected by D-flip-flops (DFF), and a combination of AND and XOR logic gates. The logic functions are expressed by the following rational equations:

$$Y_2 = I'_2 = \frac{I_2 + I_1 I'_1 D}{1 + D} \tag{2.3}$$

$$Y_1 = I'_1 = \frac{I_1}{1 + D} \tag{2.4}$$

$$Y_0 = S_0 \tag{2.5}$$

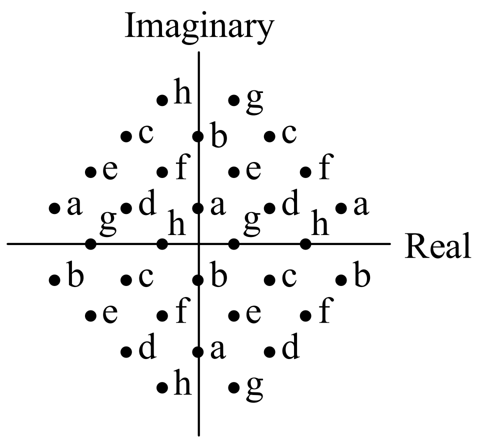
$$S_2 = S_0 D \tag{2.6}$$

$$S_1 = \frac{S_2 D + I_1' D + I_2' D + S_0 I_2' D}{1 + S_0 D} \tag{2.7}$$

$$S_0 = \frac{S_1 D + I_2' D}{1 + I_1' D} \tag{2.8}$$

The symbol D represents a delay through a DFF which functions as a memory. There are three memory units S_0, S_1, S_2 used in this convolutional encoder. The size of the encoder memory is referred to as a constrained length (k) of the code. The constraint length of the V.32 modem encoder shown in Figure 2.8 is 3. As shown in Figure 2.8, this convolutional encoder is eventually a finite-state machine. The number of states in the encoder is 2^k for a given constraint length k. In this encoder, the number of states is 2^3=8, and represented by the three bits S_0, S_1, and S_2. The states S_0, S_1, S_2 are also called delay states. The delay states at the last time interval and the output of the encoder at the current time interval Y_0, Y_1, Y_2 control the conversion between each delay state. The output $Y_0Y_1Y_2$ is also called the path state of the convolutional encoder.

The constraint condition of the convolutional encoder is that given a particular set of delay states (S_0, S_1 and S_2) in which not all path states (Y_0, Y_1, and Y_2) are possible in that time interval. For example, based on the equations (2.5), (2.6), (2.7), (2.8), if $S_{0n}S_{1n}S_{2n}$ = 000, the states that can be reached ($S_{0(n+1)}S_{1(n+1)}S_{2(n+1)}$) are 000, 010, 100 and 110. "000" is for Y_1Y_2 = 00, since $Y_0 = S_{0n}$, hence $Y_0Y_1Y_2$ = 000, so that the subset of signals denoted "a" in Figure 2.9 is associated. "010" is for Y_1Y_2 = 10, $Y_0Y_1Y_2$ = 010, corresponding with the subset of signals denoted "c". "100" is for Y_1Y_2 = 11, $Y_0Y_1Y_2$ = 011, corresponding with the subset of signals denoted "d". "110" is for Y_1Y_2 = 01, $Y_0Y_1Y_2$ = 001, corresponding with the subset of signals denoted "b. In this discussion, n and n+1 represent two continual time intervals. Figure 2.9 shows each subset formed by signals labeled a, b,… h.



	Y_0	Y_1	Y_2
a	0	0	0
b	0	0	1
c	0	1	0
d	0	1	1
e	1	0	0
f	1	0	1
g	1	1	0
h	1	1	1

Figure 2.9 A 32-signal constellation for the V.32 Modem TCM scheme

The uncoded bits Y_3Y_4 in Figure 2.8 decide every single point of each subset shown in Figure 2.9. Each set of four points is symmetrically arranged and equally spaced on the constellation as far apart as possible. Eight different $S_{0n}S_{1n}S_{2n}$ delay states corresponding with $Y_0Y_1Y_2$ path states is summarized in the trellis diagram shown in Figure 2.10

Each current delay state in Figure 2.10 has four possible new delay state destinations. In each new delay state, there are four possible inputs coming from the last delay states, which are controlled by the path states. Each path state is also a part of the output of the encoder. This path state corresponds to the delay states that traverse from the left to the right side of the trellis diagram. This trellis diagram clearly shows the input bits (Y_1, Y_2) and output bits (Y_0, Y_1, Y_2) relationship of the system associated with memory bits (S_0, S_1, S_2). The decoder algorithm is created based on the relationship shown in this trellis diagram.

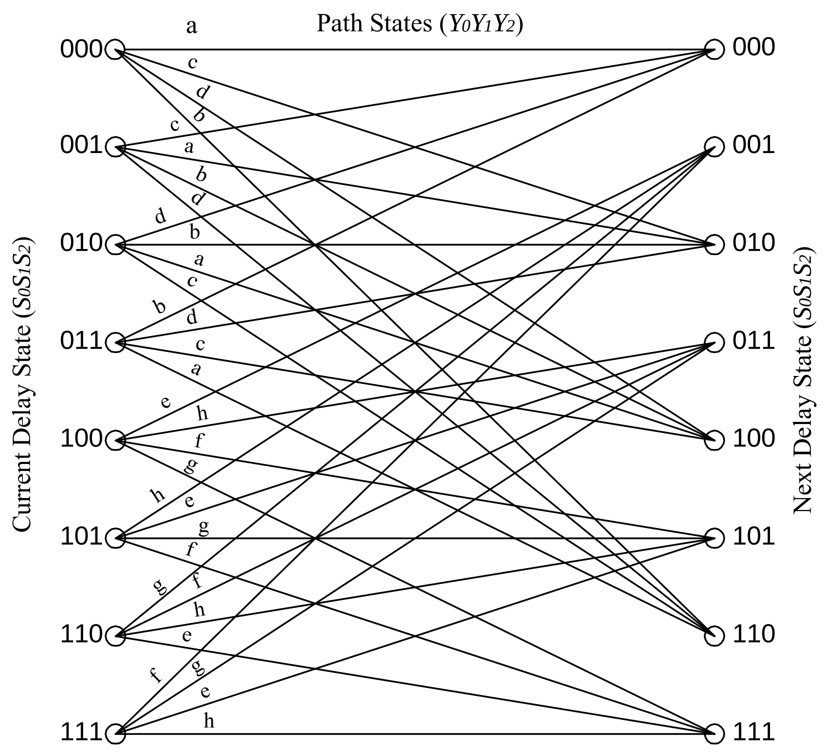

Path States ($Y_0Y_1Y_2$)
Current Delay State ($S_0S_1S_2$)
Next Delay State ($S_0S_1S_2$)
000
001
010
011
100
101
110
111
a
b
c
d
e
f
g
h

Figure 2.10 Trellis diagram of the V.32 modem TCM encoder

2.2.5 TCM Decoder

The TCM encoder is illustrated using a trellis whose branches are associated with transitions between encoder states and codeword transmitted over the channel. The primary task of the TCM decoder is to estimate the path that the codeword sequence traverses through the trellis. In this manner, TCM decoder is a reverse process of TCM encoder. In addition to the convolutional decoding, the de-mapping algorithm is a reverse function of the mapping logic function and the differential decoder performs the reverse function of the differential encoder.

Details of the TCM decoder will be described in Chapter 3. The kernel of the TCM decoder is the convolutional decoder, which completes the path estimation. The decoder algorithm used in this thesis is based on the Viterbi algorithm.

2.2.6 Viterbi Algorithm

Andrew Viterbi proposed an algorithm in 1967 to decode convolutional code and this became the Viterbi Algorithm [20]. This algorithm is an application of dynamic programming that finds “shortest paths” (maximum likelihood sequences) widely used in solving minimization problems. A critical feature of this algorithm is the complexity of the decoding process grows linearly with the number of symbols being transmitted, rather than exponentially with the number of the transmitted symbols. The Viterbi algorithm finds the sequence at a minimum Hamming distance (or Euclidean distance) from the received signal with the minimized equivalent accumulated squared error.

The Trellis diagram in Figure 2.10 shows a relationship between the delay states and the path states in each timing interval. The corresponding relations between delay states and path states construct the basic database used in the Viterbi algorithm cost function calculation. Normally, the calculation of cost functions uses two different types of distance: the Hamming distance and the Euclidean distance. In this thesis, the Hamming distance is chosen to be the cost function because the use of this distance is suitable in LUT method to simplify hardware implementation.

The Viterbi algorithm finds the path with the minimum path metric cost by sequentially moving through the trellis for each time interval. In a time interval T, from kT to (k+1)T, the receiver observes the sample received in that interval and computes the metric associated with all the branches. This metric stores the cost value of all branches as shown in the trellis diagram Figure 2.10. For example, if the memory bits (m bits) is used in the convolutional encoder, the $N = 2^m$ is the number of the states in the convolutional encoder. If each delay states have four branches associated with the new delay states, the total 4N branch costs will be calculated; but only N path metrics are retained (i.e., stored in memory). The cost of each branch is defined as the Hamming distance (or Euclidean distance) between the received symbol and the possible true symbol. The minimum branch cost will be the survivor branch for the destination states.

The cost of the path is a sum of the survivor branch cost at the current time interval and the accumulating path cost of the last time interval. Only one path with minimum accumulate cost is kept as a tracing path. Since it is experimentally determined that the optimal length of a convolutional decoder is four or five times the constraint length of the convolutional encoder [26], after 4m or 5m time interval for (n, k, m) convolutional code, the path with minimum accumulate cost of the N paths will be the survivor path. Using the survivor path, the decoder will trace back to recreate the original signal data.

Decoding begins with comparing received channel symbol pairs then building the accumulated error metric for each states branch and path. Based on these metrics, Viterbi decoder will recover the original symbol (i.e., the input of the convolutional encoder). The Viterbi decoding process is accomplished as follows:

1. Selecting the Delay State having the smallest accumulated error metric and saving the number of that delay state.
2. Iteratively performing the following trace back steps until the beginning of the trellis is reached:
    - Working backward through the state history table, for the selected state;

- Selecting a former state which is listed in the state history table as being the predecessor to that state;
- Saving the state number of each selected state.

3. Working forward through the list of selected states saved in the previous steps. Look up what input bits correspond to the transition from each predecessor state to its successor state. That is, those bits that must have been encoded by the convolutional encoder.

Figure 2.11 shows a simple example illustrating the Viterbi algorithm. The branch metrics, the survivor paths at each node, and the partial path metric of each surviving path are illustrated in this figure. The two-state (S0, S1) trellis with the branch metrics of the transitions is marked and the Viterbi algorithm is illustrated. The Viterbi algorithm finds iteratively the path with the minimum path metric of 0.5.

In Figure 2.11, pattern (1) shows the branch metric values from S0 to S1, or from S1 to S0 at every time interval k=0 to k=4. Pattern (2) assumes starting state is S0 at time interval k=0, the forward states will be S0 and S1. The cost for S0-S0 branch is 0.0, for S0-S1 is 0.8. Because 0.0<0.8, the survival branch is S0-S0, and the accumulate cost is 0.0 at k=1. Pattern (3) shows starting state is S0 at k=1 and the forward states will be S0 and S1. The cost for S0-S0 branch is 0.3, for S0-S1 is 0.0. At k=2, accumulate cost for S0 is 0.3 (i.e., 0.0 plus 0.3), for S1 is 0.0 (i.e., 0.0 plus 0.0). Because 0.0<0.3, the survival branch is S0-S1. Same process works on patterns (4) and (5). Finally, at k=4, the minimum accumulate cost is 0.5 for branch S0-S0. After tracing back, the survival path is S0-S0-S1-S0-S0; this path is shown in the bold line in Figure 2.11.

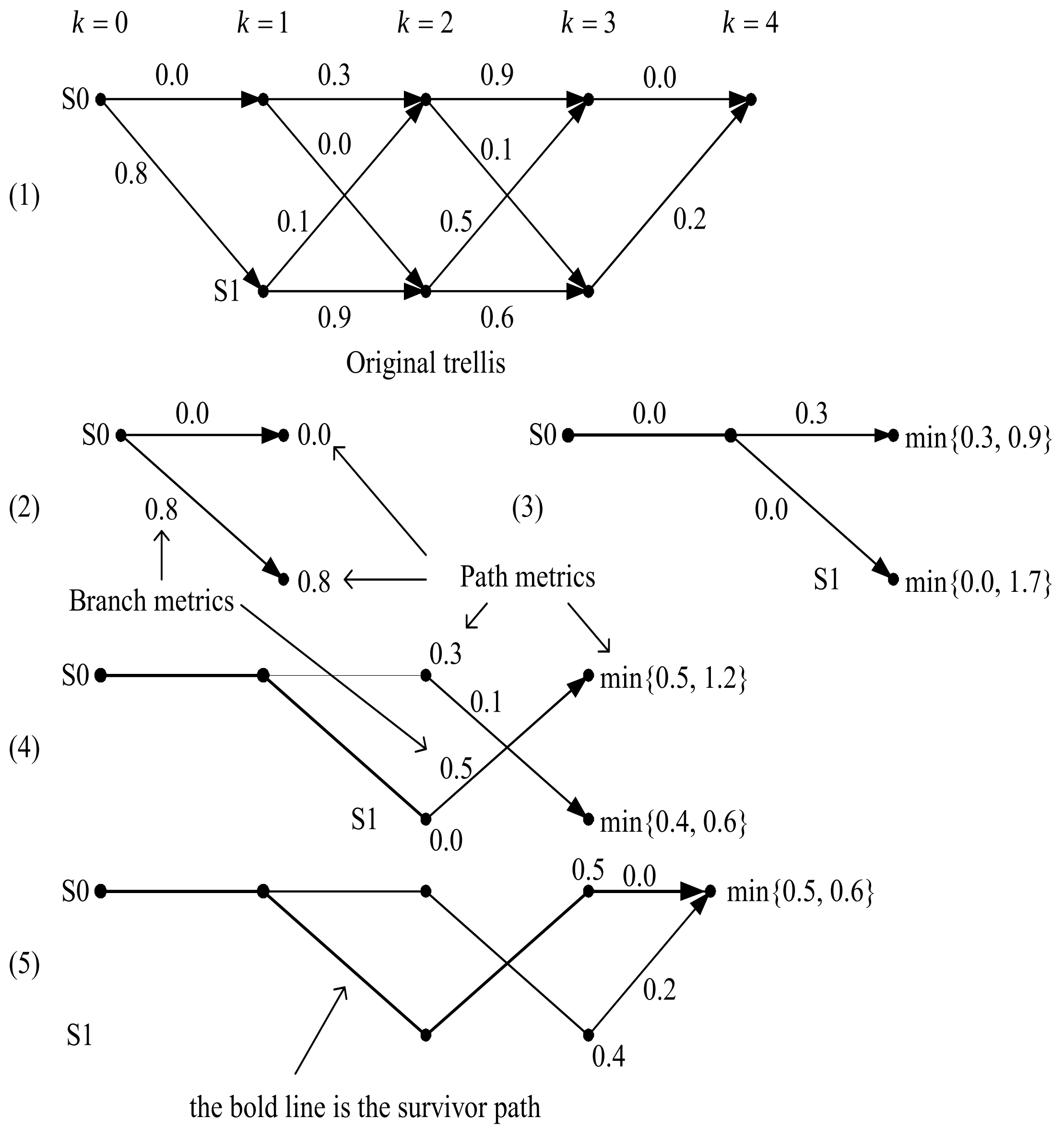


Figure 2.11 Viterbi algorithm illustration based on two-state trellis

2.3 Multi-Dimensional Trellis-Coded Modulation

The performance of TCM can be improved by increasing the number of shift register (i.e., memory bits) used in the convolutional code. This is equivalent to the increment of the number of states in the trellis coding. This has been described in

Section 2.2.1. As mentioned earlier, the coding gain saturates at the high number of states, the multi-dimensional TCM concept is introduced in 1980's to overcome this deficiency [11-15].

2.3.1 Introduction to Multi-Dimensional TCM

In a 2-Dimensional TCM encoder, if the performance of the trellis code needs to be increased, more states may be used. This implies that the number of convolutional coding memory bits (i.e., constrained length) also increases. However the returns of this manner diminishes as the coding gain increases at much slower rate as shown in Figure 2.5.

An inherent cost of the coded schemes is that the size of the 2D constellation is doubled over uncoded schemes. This is due to the fact that a redundant bit is added every signaling interval. Without the cost, the coding gain of those coded schemes would be 3 dB to 6dB through 4-states to 128-states. Using a multi-dimensional constellation with a trellis code of rate n/n+1 can reduce that cost because fewer redundant bits are added compared to the 2D constellation. In the 4D TCM encoder structure, the redundant bits generated through convolutional encoder are added alternatively in each signaling interval. Compared to 2D constellation, that cost is reduced to 1.5dB in the 4D constellation.

In the following sections, the 4D constellation TCM encoder will be described and the difference between 2D and 4D constellations can be viewed. Because the differential encoder and convolutional encoder are still being used in the coding process, the decoder algorithm is the same in both cases. The main differences between 2D and 4D are the mapping method and the clock rate in the convolutional coding.

2.3.2 Four-Dimensional TCM Encoder

Figure 2.12 shows a general 4D TCM encoder scheme. The differential encoder provides protection against 180^0 phase ambiguity in the channel as described earlier. The convolutional encoder provides Forward Error Correction information for the transmitted data.

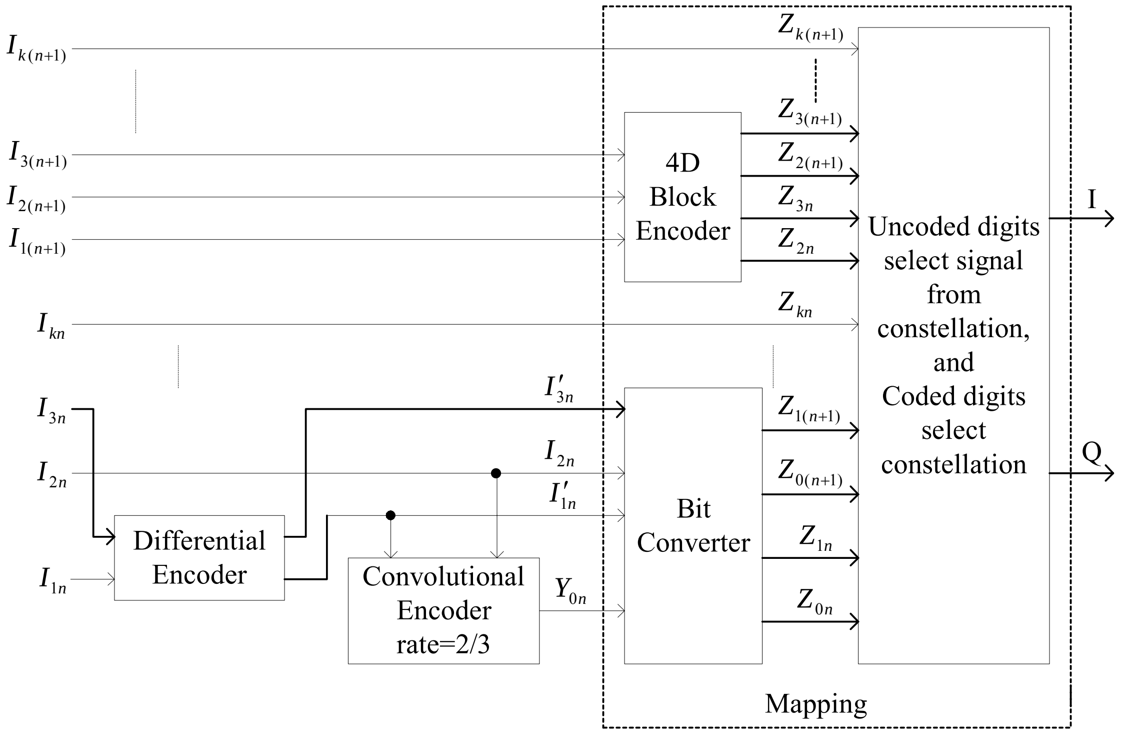


Figure 2.12 General scheme of a 4D TCM encoder

The 4D block encoder generates two pairs of selection bits, which are used to select the inner group or outer group of the two-selected 2D subset of the 4D constellation points. The bit converter generates another two pairs of the selection bits, which is used to select pair subsets in the 4D constellation. Chapter 3 will provide detail of these selection pairs.

As shown in the Figure 2.12, the trellis coding part has a same structure as in Figure 2.6. The difference is on the input signals sequence, which is continual input into differential encoder in Figure 2.6, but is alternatively input into differential encoder in Figure 2.12. The input signals are k-bit symbol sequence, and n and n+1 represent two continual time intervals.

In this thesis, k is selected to be 7 based on the DAVIC specification of the TCM used in the MMDS system [16]. After TCM coding, the signal is mapped into a 256-QAM constellation. How the 4D 256-QAM constellation mapping is performed is described in the next section.

2.3.3 TCM 4D 256-QAM Constellation Mapping

TCM schemes using 4D constellations have been reported in the literatures [11-15]. The 4D constellation is partitioned into thirty-two 4D subsets with eight times larger than the intra-subset minimum squared Euclidean distance (MSED) shown in Figure 2.13. One partitioning method of the 4D constellation is based on the partitioning of each constituent 2D constellation into four 2D subsets; concatenating a pair of 2D subsets forms each 4D subset [12]. The other partitioning method accomplishes algebraically without referring to the partitioning of the constituent 2D constellations [15].

Using the partition of 4D rectangular lattice as an example, Figure 2.13 illustrates a geometrical approach to partitioning multi-dimensional lattices into sublattices with enlarged intrasublattice MSED. The partitioning of lattice is based on the partitioning of its constituent 2D rectangular lattices. Furthermore, the partitioning of a multi-dimensional lattice is done in an iterative manner. That is, the partitioning of a 2N-dimensional lattice is based on the partitioning of the constituent N-dimensional lattices, which is in turn based on the partitioning of the constituent N/2-dimensional lattice [11]. As shown in Figure 2.13, each time the intrasublattice MSED is doubled; the number of 4D sublattices increases fourfold.

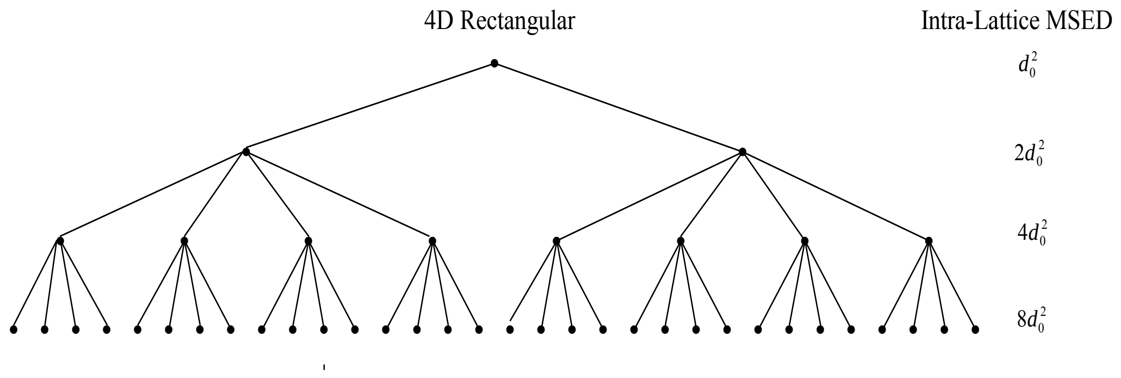


Figure 2.13 Partitioning of a 4D rectangular lattice

To transmit k information bits per signaling interval using a rate of n/n+1 trellis code with a 4D rectangular constellation, the 4D constellation of 2^{2k+1} points is constructed as follows. The first step is to obtain a constituent 4D rectangular constellation. The 2D constellation is divided into two groups, the inner group and the outer group [11]. The number of points in the inner group is 2^k, the same as that in the corresponding uncoded scheme. The number of points in the outer group is 1/2 of that of the inner group. The inner and outer groups must satisfy the following two requirements:

1) Each subset has the same number of points (inner and outer).
2) Each group is invariant under 90^0, 180^0, and 270^0 rotations.

The first requirement is necessary to convert the 4D constellation mapping into a pair of 2D constellation mappings. The second requirement preserves the symmetries of the lattice in the constellation.

As shown in Figure 2.12, on the 4D constellation mapping, a bit converter converts the four bits $Y_{0n}, I'_{1n}, I_{2n}, I'_{3n}$ into two pairs of selection bits, $Z_{0n}, Z_{1n}, Z_{0n+1}, Z_{1n+1}$, which are used to select the pair of 2D subsets corresponding to the 4D type. The 4D block encoder takes three of the remaining eleven uncoded information bits, I_{1n+1}, I_{2n+1}, and I_{3n+1}, and generates two pairs of selection bits, $Z_{2n}, Z_{3n}, Z_{2n+1}, Z_{3n+1}$, which are used to select the inner group or outer group of each selected 2D subset. The detail of this corresponding relationship will be illustrated in Chapter 3.

CHAPTER 3

TCM CODEC IMPLEMENTATION ALGORITHM

This chapter focuses on the implementation algorithm and architecture of the TCM encoder and decoder. Illustrations of the encoder and the decoder will separate into two parts based on the mapping process (i.e., mapping for 2-dimensional and 4-dimensional).

3.1 Encoder Implementation

The algorithm and hardware implementations in this thesis focus on the 16-state codec recommended in DAVIC 1.2 specifications [16]. The coded bits are ready for the 256-point constellation QAM modulation. There are two implementations on mapping: one is mapping for the 2-dimensional (2D) constellation, and the other is mapping for the 4-dimensional (4D) constellation. The following sections illustrate details of both processes.

3.1.1 A 2-Dimensional Encoder Implementation

A 16-state 2D TCM scheme is constructed with reference to both DAVIC [16] and V.32 TCM scheme [26]. The input signal symbol sequences are continually coming into the 2D TCM encoder at every time interval (i.e., 2D TCM encoder works at the full baud rate). A combination of logic gates and D-Flip-Flops (DFF) as memory devices is used to implement the differential encoder and to generate the convolutional encoder states. Figure 3.1 shows a schematic diagram of the 2D TCM encoder.

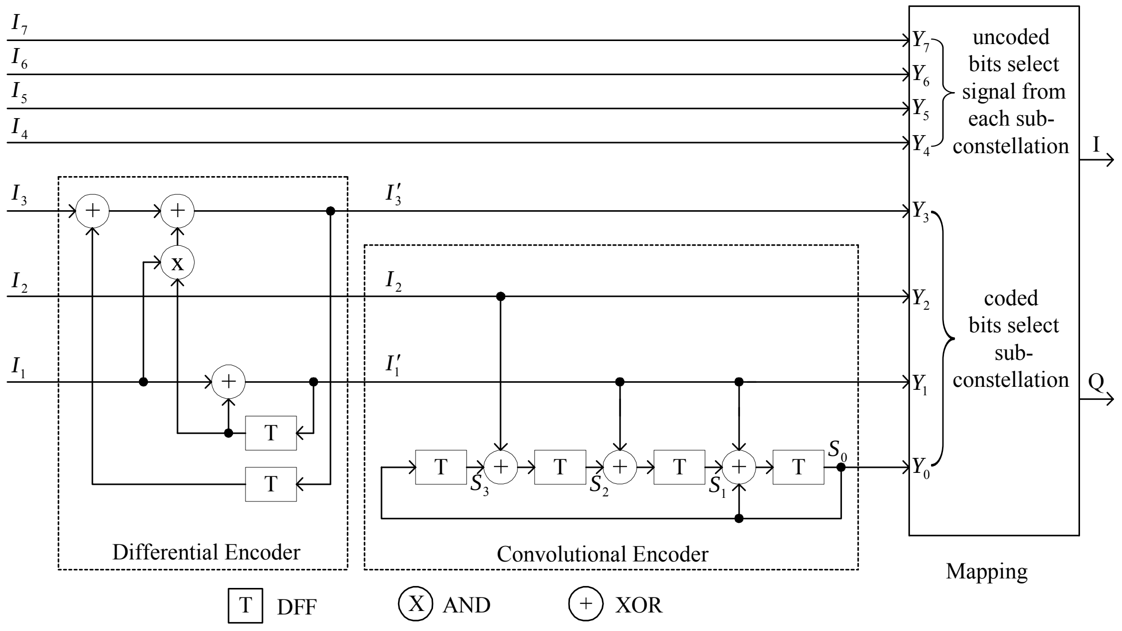


Figure 3.1 A 16-state 2-dimensional TCM encoder

The differential encoder functions as the following Boolean expression:

$$I'_1(n) = I_1(n) \oplus I'_1(n-1) \tag{3.1}$$

$$I'_3(n) = \left[I_1(n) \otimes I'_1(n-1)\right] \oplus I'_3(n-1) \oplus I_3(n) \tag{3.2}$$

where n and n-1 represent the two continual time interval.

After differential encoding, a rate of 2/3 convolutional encoder generates an extra redundant parity bit Y_0 using two bits I'_1 and I_2. The parity bit Y_0 carries only forward error-correction information. Figure 3.2 shows the relationship between I'_1, I_2 and Y_0. Outputs of the memory elements S_0, S_1, S_2 and S_3 are called delay states and values of the 3 bits Y_0, I'_1 and I_2 are called path states.

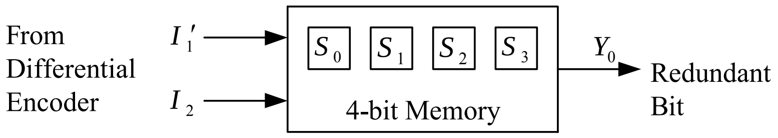


Figure 3.2 Convolutional coding in TCM encoder

The logic function of all symbol bits and memory bits for the TCM scheme shown in Figure 3.1 are obtained from the following equations:

$$Y_i = I_i, \qquad i = m \in \{2,4,5,6,7\} \tag{3.3}$$

$$Y_3 = I_3' \tag{3.4}$$

$$Y_1 = I_1' \tag{3.5}$$

$$Y_0 = S_0 \tag{3.6}$$

$$S_0 = (S_1 \oplus I_1' \oplus S_0)D \tag{3.7}$$

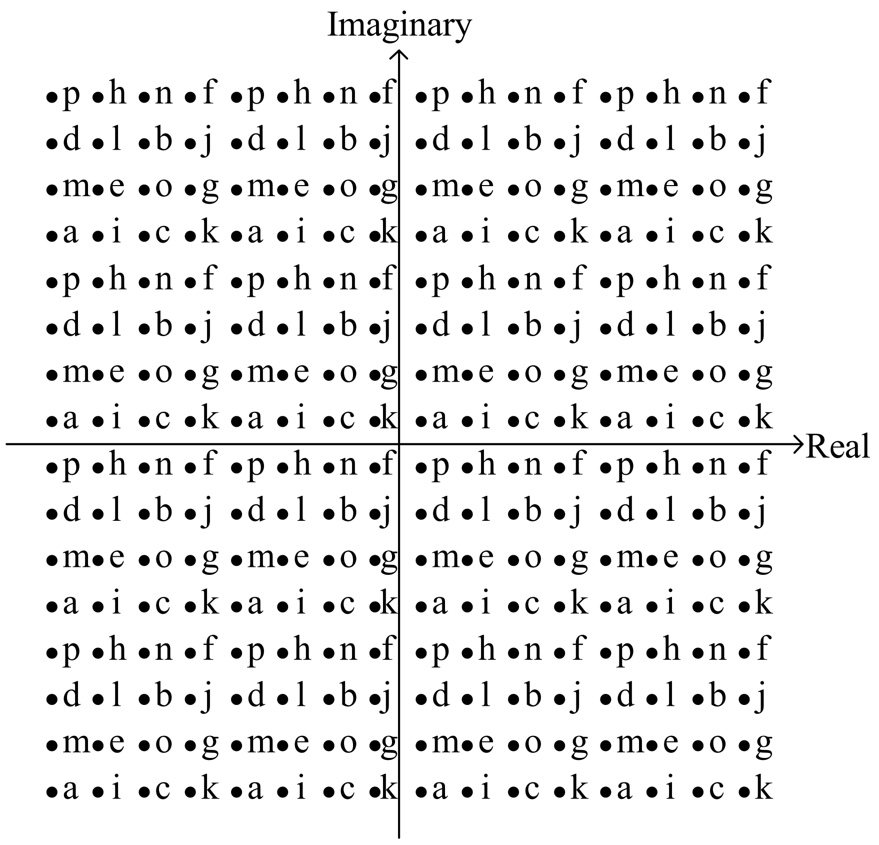
$$S_1 = (S_2 \oplus I_1')D \tag{3.8}$$

$$S_2 = (S_3 \oplus I_2)D \tag{3.9}$$

$$S_3 = S_0 D \tag{3.10}$$

where D represents the time delay of memory element (i.e., a DFF). In Figure 3.1, four coded bits (Y_0, Y_1, Y_2, Y_3) combined with the other uncoded bits (Y_4, Y_5, Y_6, and Y_7) are mapped into a 256-point constellation. The inphase signal (I) and quadrature signal (Q) are modulated and transmitted over communication channels. The 256-point constellation has 16 subsets (i.e., 2^4 from 4 coded bits). Each subset uniquely identifies a set of 16 points (i.e., 2^4 from 4 uncoded bits) out of 256 constellation points. This TCM scheme signal space mapping is defined in such a way that each set of 16 points is symmetrically arranged and equally spaced on the constellation as shown in Figure 3.3. Furthermore, each set of 16 points is as far apart as possible.

The figure illustrates sixteen subsets of a 256-point constellation formed by signals labeled a, b, c, d, e, f, g, h, i, j, k, l, m, n and o, p. The four coded bits (Y_0, Y_1, Y_2 and Y_3) select specific subset out of subsets a, b, c, d, e, f, g, h, i, j, k, l, m, n and o, p. The uncoded bits (Y_4, Y_5, Y_6, and Y_7) select a particular signal out of the signals of each subset. Figure 3.3 is based on the mapping by set partitioning methods described in Chapter 2, Section 2.2.3.


Imaginary
Real
p h n f p h n f p h n f p h n f
d l b j d l b j d l b j d l b j
m e o g m e o g m e o g m e o g
a i c k a i c k a i c k a i c k

	Y_0	Y_1	Y_2	Y_3
a	0	0	0	0
b	0	0	0	1
c	0	0	1	0
d	0	0	1	1
e	0	1	0	0
f	0	1	0	1
g	0	1	1	0
h	0	1	1	1
i	1	0	0	0
j	1	0	0	1
k	1	0	1	0
l	1	0	1	1
m	1	1	0	0
n	1	1	0	1
o	1	1	1	0
p	1	1	1	1

Figure 3.3 A 256-signal constellation for the 2D 16-state TCM scheme

3.1.2 A 4-Dimensional Encoder Implementation

Figure 3.4 depicts a schematic diagram of the 16-state TCM encoder with mapping for the 4D constellation. A combination of logic gates and DFFs as memory devices are used to implement the differential coding and generate the convolutional coding states. The two 2D symbols I_{mn} and $I_{m(n+1)}$ (m = 0, 1… k) are simultaneously input into a 4D TCM encoder at every two successive time intervals n and n+1. This means that the signal is sampled at every two clock cycles, then a 2k-bit sampled symbol at the 2T clock cycle (i.e., half baud rate) is sent out.

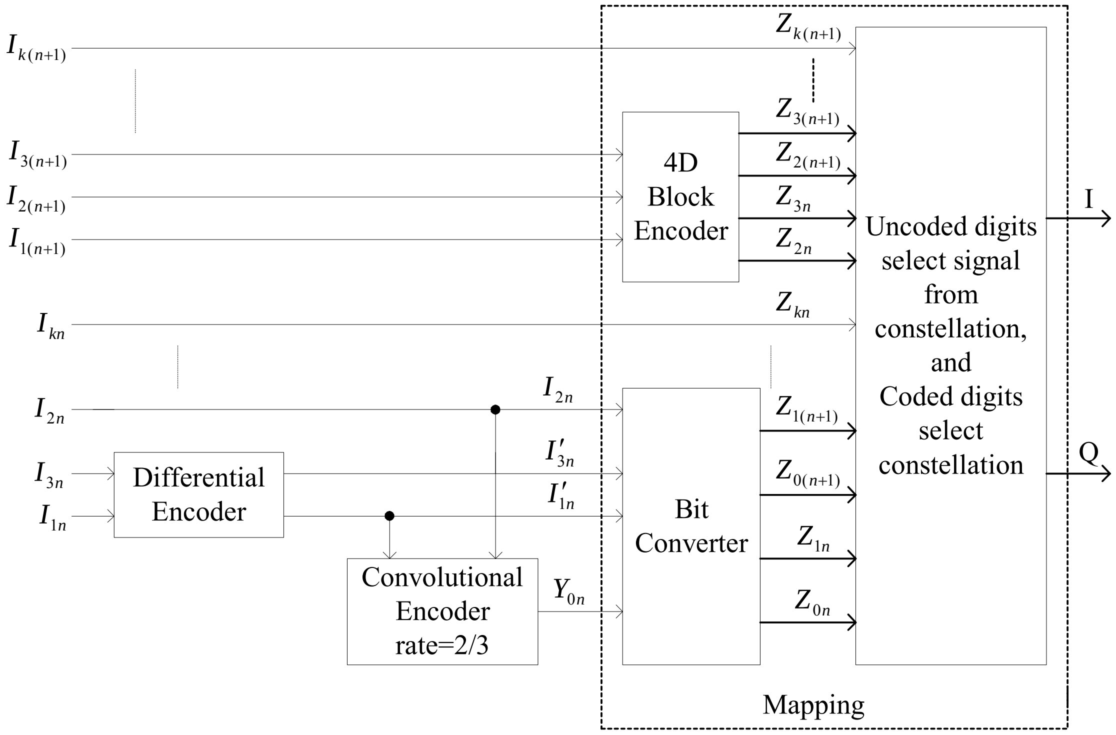


Figure 3.4 A 16-state 4-dimensional TCM encoder

The differential encoder encodes two (*I*$_{1n}$ and *I*$_{3n}$) out of three most-significant bits (*I*$_{1n}$, *I*$_{2n}$, and *I*$_{3n}$) from time interval n symbol. The differential encoder uses the Boolean expressions:

$$I'_{1n}(n') = I_{1n}(n') \oplus I'_{1n}(n'-1) \tag{3.11}$$

$$I'_{3n}(n') = [I_{1n}(n') \otimes I'_{1n}(n'-1)] \oplus I'_{3n}(n'-1) \oplus I_{3n}(n') \tag{3.12}$$

where n′ and n′-1 are new continual time intervals (i.e., half baud rate). The two differentially coded bits, I_{2n} and I'_{1n}, are convolutional encoded through a rate=2/3 encoder. The convolutional encoder has a same architecture and logic function as described in Section 3.1.1 for the 2D TCM scheme. After convolutional encoding, the extra redundant parity bit (Y_{0n}) is added, while other bits ($I_{4n} \ldots I_{kn}$, $I_{1(n+1)} \ldots I_{k(n+1)}$) in two 2D symbols remain unchanged. All the coded and uncoded bits are finally mapped into the two consecutive I-Q symbols by a mapping block.

The mapping consists of three sub-blocks 4D Block Encoder, Bit Converter and constellation subsets selection block. The bit converter is used to select a subset D_i, respective D_j in the constellation that depends on $(Y_{0n}, I'_{1n}, I_{2n}, I'_{3n})$ as defined in Table 3.1, and each signal in these subsets is given by the uncoded bits.

Table 3.1 4D subsets allocation

$Y_{0n}, I'_{1n}, I_{2n}, I'_{3n}$	4D types (D_i, D_j)	$Z_{0n}, Z_{1n}, Z_{0(n+1)}, Z_{1(n+1)}$
0 0 0 0	(D_0, D_0)	0 0 0 0
0 0 0 1	(D_2, D_2)	0 1 0 1
0 0 1 0	(D_0, D_2)	0 0 0 1
0 0 1 1	(D_2, D_0)	0 1 0 0
0 1 0 0	(D_1, D_1)	1 0 1 0
0 1 0 1	(D_3, D_3)	1 1 1 1
0 1 1 0	(D_1, D_3)	1 0 1 1
0 1 1 1	(D_3, D_1)	1 1 1 0
1 0 0 0	(D_0, D_1)	0 0 1 0
1 0 0 1	(D_2, D_3)	0 1 1 1
1 0 1 0	(D_0, D_3)	0 0 1 1
1 0 1 1	(D_2, D_1)	0 1 1 0
1 1 0 0	(D_1, D_2)	1 0 0 1
1 1 0 1	(D_3, D_0)	1 1 0 0
1 1 1 0	(D_1, D_0)	1 0 0 0
1 1 1 1	(D_3, D_2)	1 1 0 1

The D_i subsets, i = 0, 1, 2, 3, are given for each set of q uncoded bits shown (LSB to

MSB) in the QAM constellation. The combinations of $Z_{0n}Z_{1n}$ and $Z_{0(n+1)}Z_{1(n+1)}$ are used to select the pair of 2D subsets corresponding to the 4D type. Table 3.2 shows the correspondence between the bit pair $Z_{0k}Z_{1k}$, k = n, n+1, and 2D subsets D_0, D_1, D_2, D_3.

Table 3.2 Correspondence between $Z_{0k}Z_{1k}$ and four 2D subsets

$Z_{0k}Z_{1k}$	2D Subset
00	D_0
01	D_1
10	D_2
11	D_3

The Bit Converter logic function can be analyzed based on Table 3.1 as:

$$Z_{0n} = I'_{1n} \tag{3.13}$$

$$Z_{1n} = I'_{3n} \tag{3.14}$$

$$Z_{0(n+1)} = Y_{0n} \oplus I'_{1n} \tag{3.15}$$

$$Z_{1(n+1)} = [Y_{0n} \otimes I'_{1n}] \oplus I'_{3n} \oplus I_{2n} \tag{3.16}$$

The 4D-block encoder takes three of the remaining uncoded information bits $I_{1(n+1)}$, $I_{2(n+1)}$ and $I_{3(n+1)}$ to generate two pairs of selection bits, $Z_{2n}Z_{3n}$ and $Z_{2(n+1)}Z_{3(n+1)}$ in accordance to Table 3.3 listed on next page.

The pair $Z_{2n}Z_{3n}$ will be used to select the inner group or the outer group of the selected 2D subset by $Z_{0n}Z_{1n}$. The other pair $Z_{2(n+1)}Z_{3(n+1)}$ will be used to select the inner group or the outer group of the selected 2D subset by $Z_{0(n+1)}Z_{1(n+1)}$. The inner group is organized into two halves. If the bit pair $Z_{2k}Z_{3k}$ (k = n, n+1) is 00, one-half of the inner group is selected, and if the bit pair is 01, the other half of the inner group is selected; otherwise the outer group is selected. The inner group and the outer group are the two groups in the 2D constellation, as explained in Chapter 2, Section 2.3.3.

Table 3.3 4D block encoder

$I_{1(n+1)}$	$I_{2(n+1)}$	$I_{3(n+1)}$	Z_{2n}	Z_{3n}	$Z_{2(n+1)}$	$Z_{3(n+1)}$
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	1	0
0	1	1	0	1	1	0
1	0	0	1	0	0	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	1	0	1	0	1

The 4D-block encoder logic function can be obtained based on Table 3.3 as:

$$Z_{2n} = I_{1(n+1)} \otimes \overline{I_{2(n+1)}} = I_{1(n+1)} \otimes (I_{1(n+1)} \oplus I_{2(n+1)}) \tag{3.17}$$

$$Z_{3n} = I_{2(n+1)} \otimes (I_{1(n+1)} + I_{3(n+1)}) \tag{3.18}$$

$$Z_{2(n+1)} = \overline{I_{1(n+1)}} \otimes I_{2(n+1)} = I_{2(n+1)} \otimes (I_{1(n+1)} \oplus I_{2(n+1)}) \tag{3.19}$$

$$Z_{3(n+1)} = I_{3(n+1)} \otimes (I_{1(n+1)} + \overline{I_{2(n+1)}}) \tag{3.20}$$

The TCM encoder is easy to implement into hardware using the above information. The issue of TCM codec is how to implement the decoder in hardware to achieve high speed with a minimum silicon area. The next section explains in detail the TCM decoder implementation.

3.2 Decoder Implementation

In this thesis, the demodulation process is assumed to be separated from the decoder process and the detection process involves hard decisions. Each single point from the constellation has been selected. The free Hamming distance of the convolutional code is used in Viterbi Algorithm to calculate the cost of each branch and path. The cost values are also called branch metric and path metric.

The TCM technique recommends using soft decoding, which means the demodulation process is incorporated into the decoding process and the Euclidean distance is used as the cost function quantity in the Viterbi Algorithm. As mentioned in Chapter 1, the soft decoding obtains approximate 2dB performance over the hard decoding. Due to the effect on the decoding performance is not significant, using hard decoding has a tremendous benefit in hardware implementation. The reason for this benefit is when the Euclidean distance is increased by set partitioning mapping method; any noise perturbation is less likely to affect the estimation of the points to be detected using the hard decision detection.

In order to increase the speed of the decoder, two pre-calculated look-up tables (LUTs) are used. One LUT is used to obtain the Hamming distance of each branch metric when the branch cost is required to be calculated. The other LUT is used to make decisions on the output of convolutional decoder when two continual time interval delay states are known. The rest of this chapter describes in detail of the constructing of the Hamming Distance Look-Up Table (HDLUT), the Output Look-Up Table (OLUT) and the decoder implementation architecture.

3.2.1 **Structure of the HDLUT and OLUT**

Based on the 16-state TCM encoder illustrated in Section 3.1, both 2D and 4D TCM schemes have the same convolutional encoder structure, except they work at the different baud rates. According to equations (3.6) to (3.10), the 16-state Trellis diagram for both TCM schemes is constructed and shown in Figure 3.5.

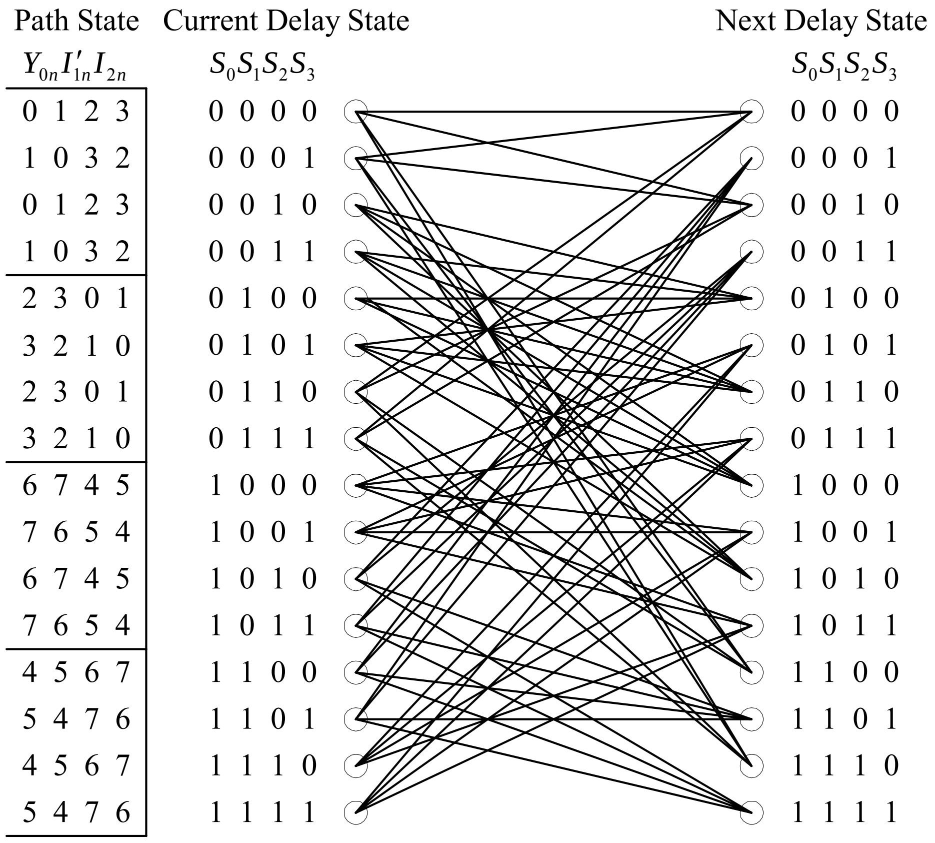


Figure 3.5 Trellis diagram of the 16-states TCM encoder

In this 16-state Trellis diagram, given a particular set of delay states ($S_0S_1S_2S_3$), not all eight path states ($Y_0I'_{1n}I_{2n}$) are possible to be used in that time interval, and not all 16 delay states are possible to be reached in the next delay state. The particular path chosen at that time interval depends on the current path state of the encoder, which means the current path state controls which successor delay state to be reached from that particular current delay state. For instance, if the current delay state is "0000", the successor delay state ($S_0S_1S_2S_3$) will be "0000", "0010", "1100" and "1110" when corresponding to the path state ($Y_0I'_{1n}I_{2n}$) are "000", "001", "010" and "011". For one specific current delay state, there are only four successor delay states being reached out of sixteen delay states, and each branch corresponds with one specific path state out of the eight possible path states.

From the trellis encoder, the path state also represents the coded bits for the output of the convolutional encoder at the current time interval, so it reflects the true part bits out of a symbol for the decoder input symbols. Each path state is one 3-bit binary data of "000", "001",…… "111" for the 2/3 convolutional encoder. In the TCM convolutional decoder, the input symbol should be one of eight possible three coded bits from "000" to "111" combined with the other uncoded bits at each time interval. Comparing each possible symbol with the truth symbol, the Hamming distance is easy to be calculated. This distance is computed by simply counting how many bit differences between the received channel symbol pairs and the possible channel symbol pairs. Table 3.4 shows a pre-calculated HDLUT.

Table 3.4 A Hamming Distance table

Truth Path State	The corresponding HD with the possible part bit out of received data sequence of the decoder							
	000	001	010	011	100	101	110	111
000 (P0)	0	1	1	2	1	2	2	3
001 (P1)	1	0	2	1	2	1	3	2
010 (P2)	1	2	0	1	2	3	1	2
011 (P3)	2	1	1	0	3	2	2	1
100 (P4)	1	2	2	3	0	1	1	2
101 (P5)	2	1	3	2	1	0	2	1
110 (P6)	2	3	1	2	1	2	0	1
111 (P7)	3	2	2	1	2	1	1	0

In this table, P0 ~ P7 represent truth path state from “000” to “111”. If the current delay state ($S_0S_1S_2S_3$) is “0001”, the next delay state will be "0000", "0010", "1100" and "1110" when corresponding to the path state ($Y_0 I'_{1n} I_{2n}$) are P1, P0, P3 and P2 as shown in Table 3.5. At this situation, if the received symbol is “00110110”, the three LSB bits are “110”. It can be seen from Table 3.4 that the cost for each branch will be 3, 2, 2, and 1. Table 3.5 shows the relationship between the delay states and the path states.

Table 3.5 HDLUT using in the TCM decoder

Current Delay State	Next Delay State	The Path State	Current Delay State	Next Delay State	The Path State	Current Delay State	Next Delay State	The Path State	Current Delay State	Next Delay State	The Path State
0000	0000	P0	0100	1000	P0	1000	1001	P4	1100	0001	P4
	0010	P1		1010	P1		1011	P5		0011	P5
	1100	P2		0100	P2		0101	P6		1101	P6
	1110	P3		0110	P3		0111	P7		1111	P7
0001	0000	P1	0101	1000	P1	1001	1001	P5	1101	0001	P5
	0010	P0		1010	P0		1011	P4		0011	P4
	1100	P3		0100	P3		0101	P7		1101	P7
	1110	P2		0110	P2		0111	P6		1111	P6
0010	1000	P2	0110	0000	P2	1010	0001	P6	1110	1001	P6
	1010	P3		0010	P3		0011	P7		1011	P7
	0100	P0		1100	P0		1101	P4		0101	P4
	0110	P1		1110	P1		1111	P5		0111	P5
0011	1000	P3	0111	0000	P3	1011	0001	P7	1111	1001	P7
	1010	P2		0010	P2		0011	P6		1011	P6
	0100	P1		1100	P1		1101	P5		0101	P5
	0110	P0		1110	P0		1111	P4		0111	P4

In both Figure 3.1 and Figure 3.4, after convolutional encoder, only one redundant bit Y_0 is generated. This bit is controlled by the delay states $S_0 S_1 S_2 S_3$ and by a pair bits I'_{1n} and I_{2n}. The two bits I'_{1n} and I_{2n} are the inputs of the convolutional encoder and they are the outputs of the convolutional decoder. $Y_0 I'_{1n} I_{2n}$ is the path state shown in Figure 3.5 the trellis diagram. Hence, if the current Delay State and the next

Delay State are determined, the particular path $Y_0 I'_{1n} I_{2n}$ will be determined, then I'_{1n} and I_{2n} will be recovered. For example, after tracing back, two delay states "0000" and "1100" are determined to be the continual two time interval delay states in survival path. Table 3.5 shows the path state is P2 ("010") corresponding to this branch. As the result, a pair bit "10" will be the decoded I'_{1n} and I_{2n} bits. Based on Table 3.5 and Figure 3.5, the OLUT is constructed and shown in Table 3.6.

Table 3.6 An OLUT using in the TCM decoder

Current Delay State	The output ($I'_{1n}I_{2n}$) of convolutional decoder when Next Delay State ($S_0S_1S_2S_3$ in decimal) is given															
($S_0S_1S_2S_3$)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000 (0)	00		01										10		11	
0001 (1)	01		00										11		10	
0010 (2)					00		01		10		11					
0011 (3)					01		00		11		10					
0100 (4)					10		11		00		01					
0101 (5)					11		10		01		00					
0110 (6)	10		11										00		01	
0111 (7)	11		10										01		00	
1000 (8)						10		11		00		01				
1001 (9)						11		10		01		00				
1010 (10)		10		11										00		01
1011 (11)		11		10										01		00
1100 (12)		00		01										10		11
1101 (13)		01		00										11		10
1110 (14)						00		01		10		11				
1111 (15)						01		00		11		10				

*Note: Empty cells in the table represent no relationship between two delay states

The table shows a regular pattern between output and delay states. For example, if output is "00, 01, 10, 11", the corresponding current delay states ($S_0S_1S_2S_3$) are "0000" and "0010", or "1100" and "1110". According to the complementary law "$x + \overline{x} = 1$", states $S_0S_1S_2S_3$ ("0000" + "0010") and ("1100" + "1110") can be combined to ($S_0S_1S_3$) "000" or "110". If consider from next delay state, for output "00, 01, 10, 11", the corresponding next delay states ($S_0S_1S_2S_3$) are four respectively groups: (0000, 0100, 0001, 0101), (0010, 0110, 0011, 0111), (1000, 1100, 1001, 1101) and (1010, 1110, 1011, 1111). Using the same law, they can be combined to (S_0S_2) “00”, “01”, “10” and “11”. Followed this pattern, Table 3.6 is simplified to Table 3.7. This simplification saves memory space or storage cells required in the hardware implementation.

Table 3.7 The simplified OLUT using in TCM decoder

Three Bits of Current Delay State ($S_0S_1S_3$)	The output ($I'_{1n}I_{2n}$) of convolutional decoder when the two bits of Next Delay State (S_0S_2) is given			
	00	01	10	11
000	00	01	10	11
001	01	00	11	10
010	10	11	00	01
011	11	10	01	00
100	10	11	00	01
101	11	10	01	00
110	00	01	10	11
111	01	00	11	10

From this result, a decoding look-up-table is created and used in conjunction with the Viterbi algorithm to decode the convolutional code. Next section will go into detail of the architecture in implementation of the 2-dimensional and 4-dimensional TCM decoders.

3.2.2 A Novel TCM Decoder Architecture Implementation

The TCM decoder includes three processes: de-mapping, convolutional decoding, and differential decoding. The implementation algorithm of this convolutional decoding is mainly based on the Viterbi algorithm given in Chapter 2. Parallel processing is used at each node (i.e., state) to trace the history delay states of the sixteen paths and the Hamming Distance is used as the cost function. The distance calculation circuit is omitted because the cost of each branch is obtained from HDLUT. This technically reduces complexity of the decoder architecture. Utilizing the characteristics of shift registers, such as timing delay and memory, the real time tracing back process of the Viterbi algorithm is accomplished. In addition, in order to shorten the time of the critical path in the decoder, a pipelining technique is used; this technique increases the decoding speed.

When the receiver receives the signal, after de-modulation, the sampled signal enters the de-mapping process. The difference between 2-dimensional and 4-dimensional TCM scheme is in the processing of signal de-mapping. The convolutional decoding and the differential decoding processes have the same methodology and structure, except they work at the different baud rates. The next two subsections describe more details of the decoder for 2D and 4D TCM scheme respectively.

3.2.2.1 A Decoder for 2-Dimensional TCM Scheme

For 2D TCM scheme, the coded bits after convolutional encoder combined with the remaining uncoded bits directly mapped into the 256-point constellation as shown in Figure 3.3. The signal from the demodulation is sampled. The de-mapping is simply dividing the sampled signal into two parts: 3-bits go through the second step of TCM decoder for the convolutional decoding process, the remaining 5-bits is delayed using a group of shift registers. Figure 3.6 shows the structure of 2D 16-state TCM decoder.

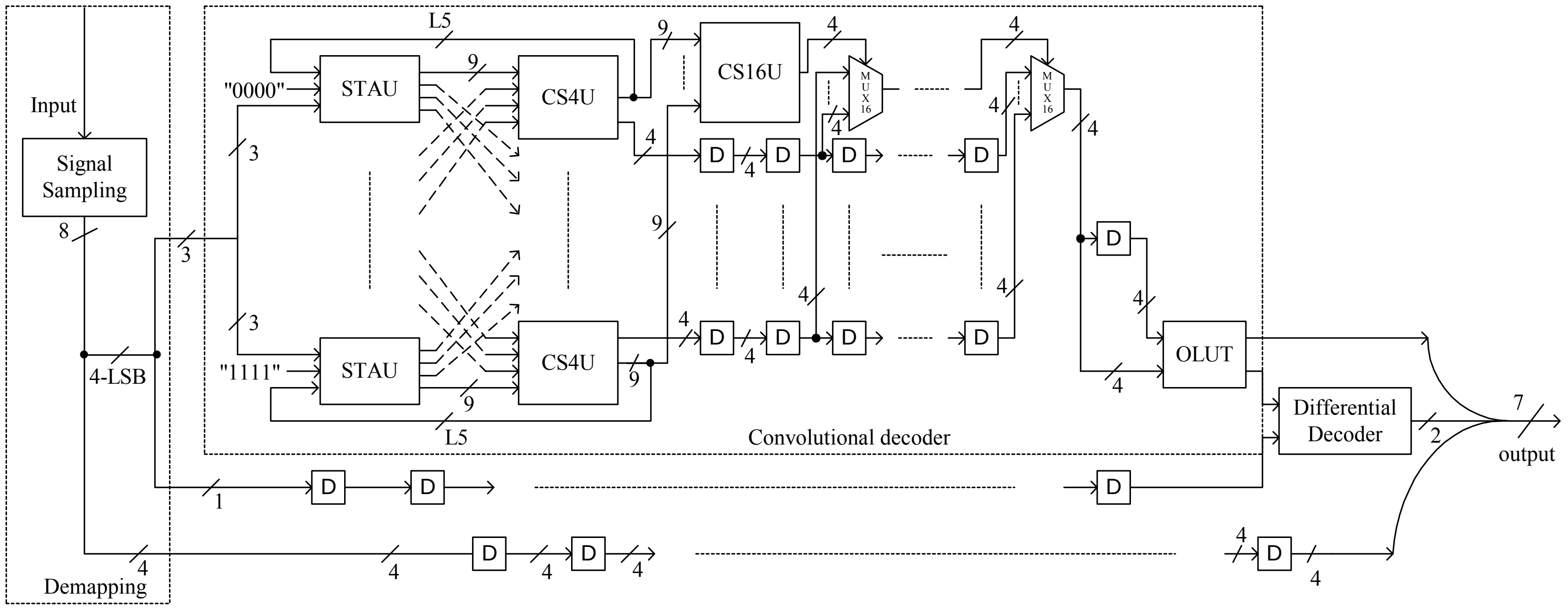


Figure 3.6 A 2-dimensional 16-state TCM decoder

As shown in Figure 3.6, 3-bits out of four LSB bits go through convolutional decoding process directly; the remaining five bits enter into five groups shift register. The number of each group shift registers is determined by the latency of the convolutional decoder. After convolutional decoding process completes, 1-bit output from the convolutional decoder will combine with the 1-bit out of 5-bits from shifter register to execute the differential decoding process.

After differential decoding, 1-bit from the output of the convolutional decoder, 2-bits from the output of the differential decoder and 4-bits from the output of de-mapping are combined to form a 7-bit output symbol of the TCM decoder.

In the convolutional decoder, several steps are sequentially accomplished based on the Viterbi algorithm. They are represented by blocks such as State-Transition and Add Unit (STAU), Compare-Selection 4 Unit (CS4U), Compare-Selection 16 Unit (CS16U), 16-to-1 multiplexer (MUX16) and OLUT. Since this thesis focuses on 16-state TCM codec, sixteen parallel paths are traced. The algorithm of both CS4U and CS16U relies heavily on the cost function. The minimum cost branch or path is selected.

The State-Transition and Add Unit is based on the Trellis diagram shown in Figure 3.5. The current delay states split into four new delay states; each branch cost is obtained from the HDLUT. The current cost adds to the previous accumulative path cost. The accumulative cost combines with the current delay state to be the output of the STAU. Figure 3.7 shows the architecture of the STAU block.

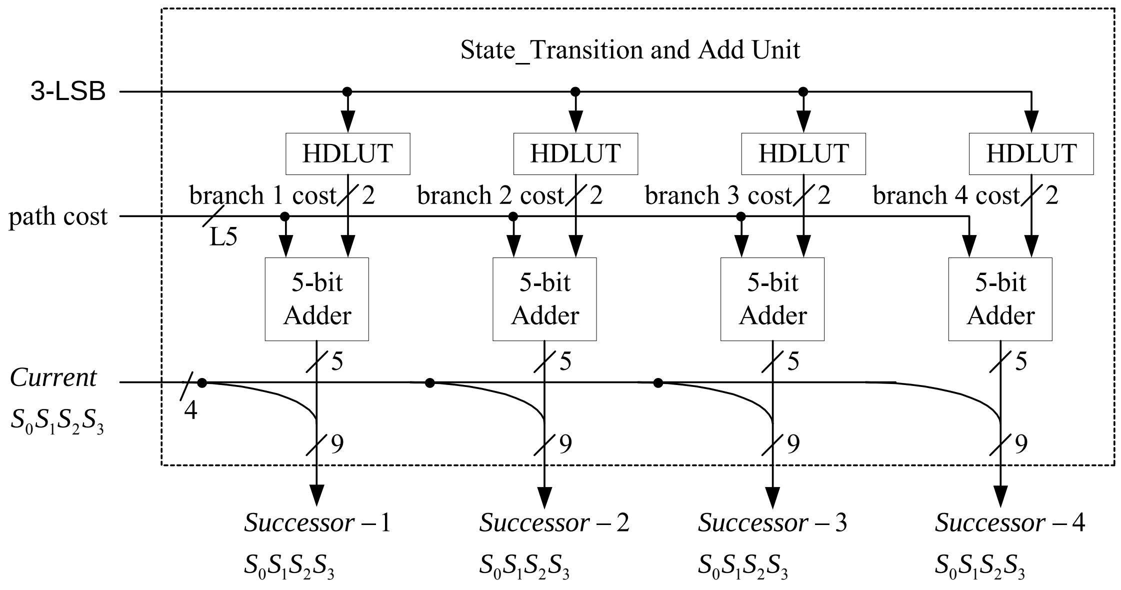


Figure 3.7 Architecture of the STAU block

The outputs from the STAU will input into the CS4U to select the survivor branch based on the Trellis diagram. The same successor (i.e., next delay state) branches are input into a same CS4U block. Sixteen CS4U blocks are in path order parallel arranged from “0000” to “1111”. Since each output of the STAU includes accumulative cost and current delay state, the CS4U selects the minimum accumulative cost branch as the new path accumulative cost. For this minimum accumulative cost, the corresponding current delay state is chosen to be the predecessor delay state. The predecessor of the chosen survivor branch will be one output of the CS4U. The other output of the CS4U is the new path accumulative cost combined with the path label (i.e., sixteen paths “0000”, “0001” … “1111”).

The predecessor output of the CS4U is put into a shift register. The groups of shift registers are used to store the history delay states of the 16 paths. A minimum number of 16 shift registers is used to store the past delay states. This numbers is 4 or 5 times the code constraint length [26]; this length is the number of memory bits in the convolutional encoder. A constraint length of 4 is used in this 16-state TCM scheme.

The new path accumulative cost is taken from the other output of the CS4U and sent back to the STAU. This path cost will be used in the STAU block when the branch accumulative cost is calculated at the next time interval. Simultaneously, the accumulative path cost inputs to the CS16U to select the survivor path, which is the path with minimum accumulative cost.

Once the survivor path is chosen, sixteen sequential MUX16 logics perform the delay states trace back. Once the foremost two delay states are produced, they will be sent to the OLUT. The OLUT is a pseudo-ROM and used to reconstruct the data I'_{1n} and I_{2n} which was sent to convolutional encoder. The address to this ROM is the combination of two consequence delay states as illustrated in Table 3.7 in Section 3.2.1.

Finally, the LSB bit I'_{1n} of the convolutional decoder and the delayed bit I'_{3n} out from the 4-LSB are differential decoded. The differential decoder performs the reverse function of the differential encoder in 16-states TCM encoder as follows:

$$I_{1n}(n') = I'_{1n}(n') \oplus I'_{1n}(n'-1) \tag{3.21}$$

$$I_{3n}(n') = [I'_{1n}(n') \otimes I'_{1n}(n'-1)] \oplus I_{sn}(n'-1) \oplus I_{3n}(n') \tag{3.22}$$

where n′ and n′-1 are continual half baud rate time intervals.

3.2.2.2 A Decoder for 4-Dimensional TCM Scheme

For the 4D TCM encoder scheme, the coded bits after convolutional encoded are converted through a Bit Converter to generate selection pair bits for the 4-dimensional mapping as shown in Figure 3.4. Hence for 4D TCM decoder, the signal from the demodulation is sampled and the de-mapping process performs the reverse of the encoder Bit Converter block and the reverse of the 4D-block encoder.

The logic function of the Bit Converter and the 4D-block decoder in the 4D TCM decoder are the reverse functions of the 4D TCM encoder. The logic relationship of these two blocks was described in Section 3.1.2.

The Bit Converter logic function in decoder can be analyzed based on Table 3.1 as following:

$$Y_{0n} = Z_{0n} \oplus Z_{0(n+1)} \qquad (3.23)$$

$$I'_{1n} = Z_{0n} \qquad (3.24)$$

$$I_{2n} = Z_{1(n+1)} \oplus Z_{1n} \oplus Z_{0n} \oplus (Z_{0n} \otimes Z_{0(n+1)}) \qquad (3.25)$$

$$I'_{3n} = Z_{1n} \qquad (3.26)$$

The 4D-block decoder logic function can be analyzed based on Table 3.3 as:

$$I_{1(n+1)} = \overline{Z_{2(n+1)}} \otimes (Z_{2n} \oplus Z_{3n}) \qquad (3.27)$$

$$I_{2(n+1)} = Z_{3n} + Z_{2(n+1)} \qquad (3.28)$$

$$I_{3(n+1)} = Z_{3(n+1)} + Z_{3n} \otimes Z_{2(n+1)} \qquad (3.29)$$

Equations (3.23) ~ (3.29) can be easily converted into a schematic diagram. Figure 3.8 shows a schematic diagram of the Bit Converter used in the 4D TCM decoder. Figure 3.9 shows the schematic diagram of the 4D-block decoder.

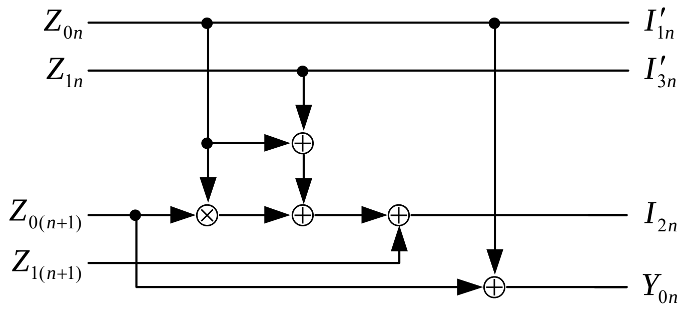


Figure 3.8 Schematic diagram of the bit converter

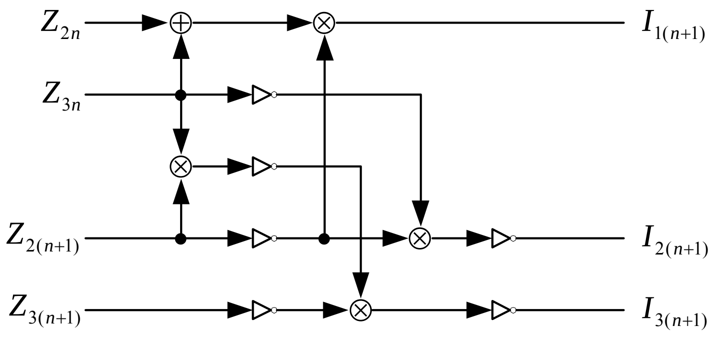


Figure 3.9 Schematic diagram of the 4D-block decoder

Figure 3.10 shows the structure of a 16-state 4-dimensional TCM decoder. In this structure, after de-mapping, three LSB bits out of four bits from the Bit Converter are sent to the convolutional decoder. The other bit is send to the shift register and will be combined with the output bit from the convolutional decoder. These two bits are then sent to a differential decoder. The remaining eight uncoded bits and three bits from 4D-block decoder are also sent to the shift registers. The number of shift registers is determined by the latency of the convolutional decoding process. This latency is the same as the 2D scheme, except the clock rate is different. If the clock of the 2D scheme is T, the clock of the 4D scheme is 2T.

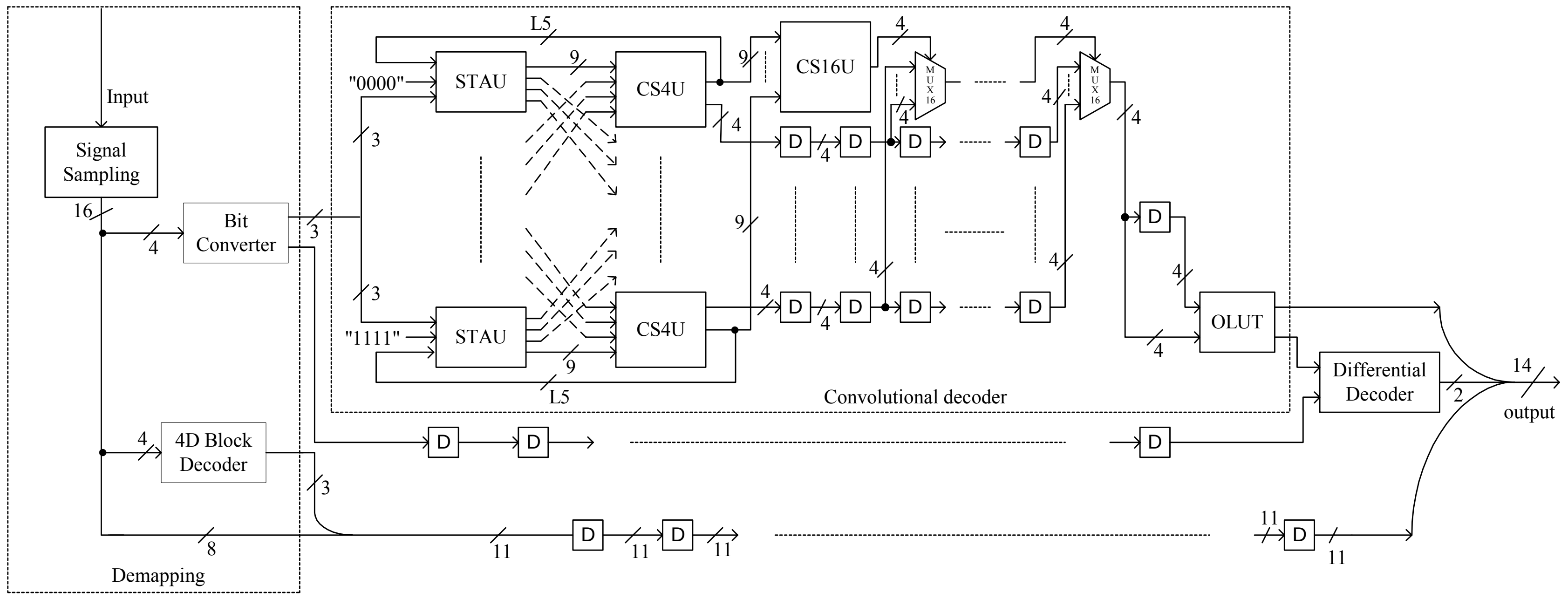


Figure 3.10 A 16-state 4-dimensional TCM decoder

After the differential-decoding completes, the 14-bit symbol will be the output of the TCM 4D decoder. This symbol includes 1-bit from the output of the convolutional decoder, 2-bits from the output of the differential decoder and 11-bits from the output of the de-mapping. Details of the convolutional decoder and the differential decoder are the same as in the 2D TCM scheme, which are described in Section 3.2.2.1.

For the 4D TCM scheme, two more blocks are required to accomplish the conversion between two symbols from two continual clock cycles of full baud rate to one symbol for half baud rate. Since two continual symbols are coded simultaneously for 4D TCM encoder as illustrated in Figure 3.4, the system input works at a full baud rate while the 4D TCM encoder and decoder work at a half baud rate. The block diagram explaining this idea is shown in Figure 3.11. The detailed architecture of the blocks will be discussed in the next chapter.

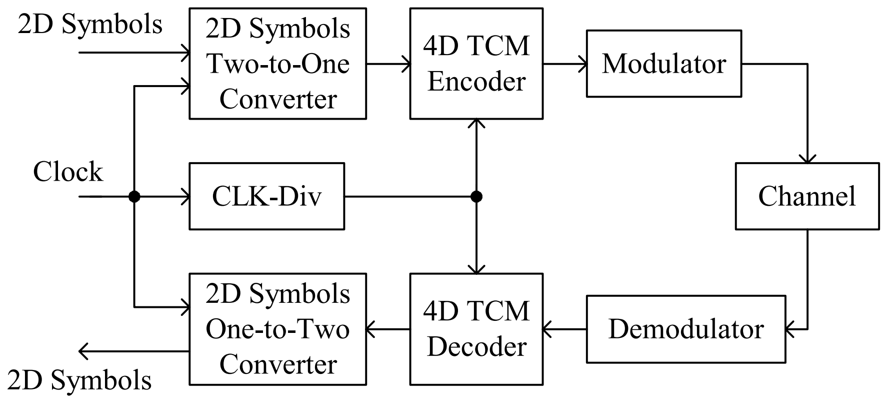


Figure 3.11 The 4-dimensional TCM coding system

CHAPTER 4

TCM CODEC ASIC IMPLEMENTATION

This chapter introduces the concept and design flow of the ASIC; centering on the TCM codec ASIC implementation, this chapter also illustrates the methodology used to increase the speed of the codec such as pipelining techniques, and other chip architectures such as the clock divider and the Built-In-Self-Test (BIST).

4.1 ASIC Design Introduction

Application Specific Integrated Circuit (ASIC) design is an efficient way to implement specific system functions into a single chip [28]. The technique is widely used in high-speed internet chip design and telecommunication applications. ASICs offer some advantages such as high speed, small area and low cost manufactory in high volumes for industry specific products. A typical ASIC design process includes three basic stages: HDL design capture, HDL design synthesis, and design implementation. Figure 4.1 shows a flow chart of an ASIC design process.

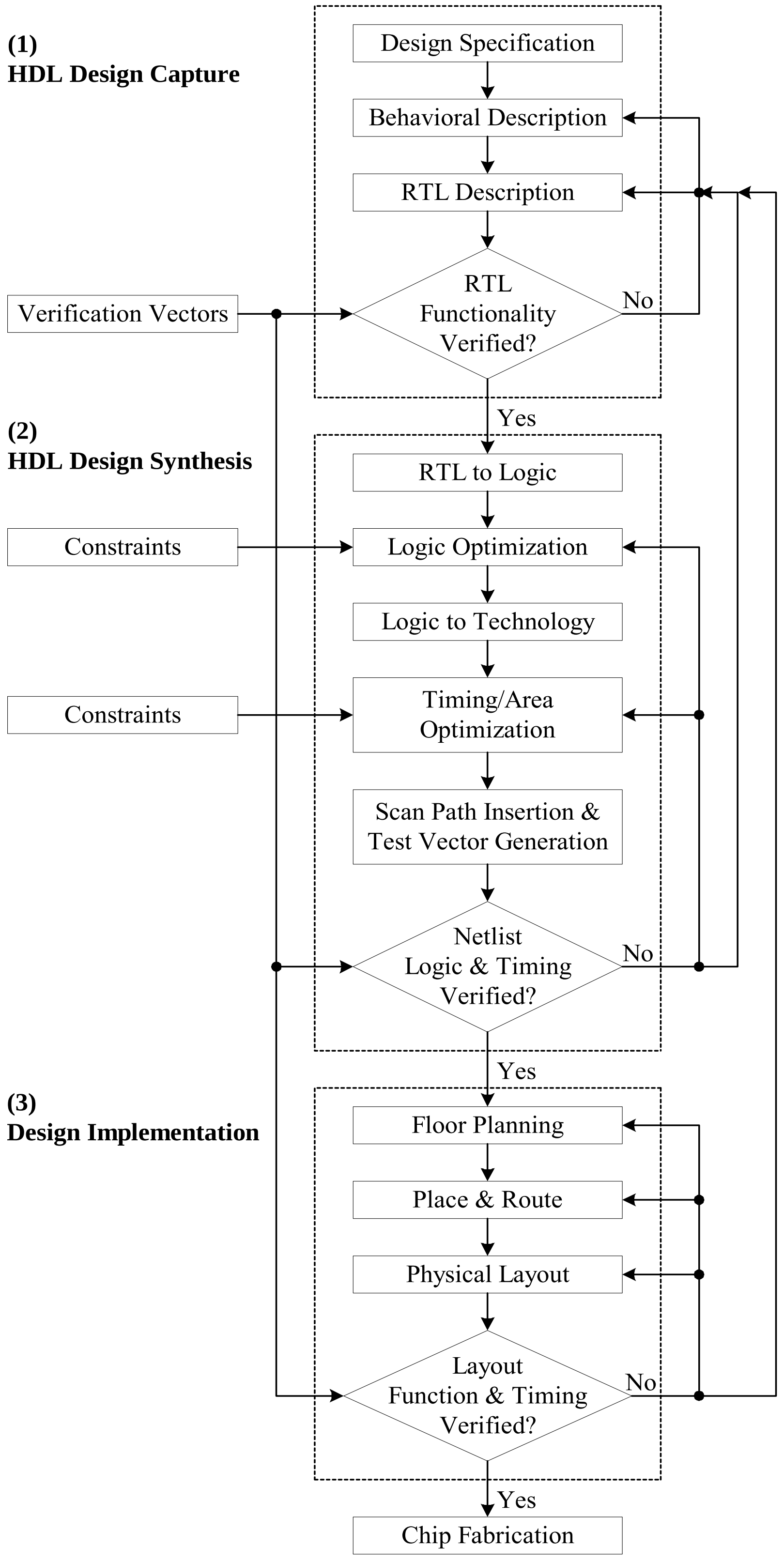


Figure 4.1 A flow chart of the ASIC design process

(1) **HDL design capture** implements the "top-down" design methodology from abstract concept or algorithm down to hardware in manageable and verifiable steps. This involves developing a design specification that will be used to create a high level behavioral abstraction with high level programming languages such as C or C++. Additionally, the behavioral abstraction may also be created using hardware description languages (HDL) such as VHDL or Verilog.

The behavioral abstraction should be simulated in order to verify that the desired functionality is captured completely and correctly. The behavioral abstraction is then used as a reference to create and refine a synthesizable register transfer level (RTL) abstraction. This RTL code captures desired functionality required by the design specification. The difference between a purely behavior abstraction model and a RTL abstraction model will be described later; both models are used in functionality verification, but slightly different in the hardware synthesis.

Generally, the designs are represented in HDL at three levels of abstraction:

- Behavioral level: a design is implemented in terms of the desired algorithm, much like software programming and without regard for actual hardware. For example, a Verilog HDL model written at the behavioral level is usually not synthesizable into hardware model by the synthesis tools.
- Register transfer level: a design is implicitly modeled in terms of hardware registers and combinational logic that exists between them to provide the desired data processing. The key feature of this level is that an RTL description can be translated into hardware model by the synthesis tools.
- Structural level: a design is realized through explicit instances of logic

primitives and interconnections between them. This level is also referred to as a "gate level" model. It is a hardware model and is used to create the floorplan of the circuit layout. Most synthesis tools can generate the gate-level model after synthesizing the RTL code.

HDL Design Capture is completed with "pre-synthesis" simulations to verify that the RTL abstraction fully provides the desired function. The functional verification in the design process that occurs at this point must be as complete and thorough as possible. The test vectors used during simulation should provide the fault coverage necessary to ensure the design meet specifications.

(2) **HDL design synthesis** involves steps using a synthesis tool to:

- Translate the abstract RTL design description to register elements and combinational logic.
- Optimize the combinational logic by minimizing and flattening the resultant Boolean equations.
- Translate the optimized logic level description to a gate level description using logic cells from the specified technology library.
- Optimize the gate level description using cell substitution to meet the specified area and timing constraints.
- Produce a gate-level netlist of the optimized circuit with accurate cell timing information.

HDL design synthesis finishes with "post-synthesis" simulations to verify that the gate level circuit fully provides the desired functionality and meets the appropriate timing requirements.

(3) **Design implementation** involves steps using layout tools [31] to:

- Create a floorplan for the IC from the gate-level netlist for the design including a default group of cells, I/O ring connected, and defined placement sites for all the cells.
- Place core cells by using forward-annotated timing constraints information from the synthesis step.
- Add clock buffer cells and nets to create a balanced clock tree, which exceeds the parameters specified in synthesis.
- Generate a “golden” netlist of the design to be used for final verification.
- Verify the functionality of the “golden” netlist.
- Route the power, clock, and regular nets of the design.
- Functionally verify the physical (placed and routed) layout of design that contains the same instances, nets, and connectivity as the verified “golden” netlist. Alternatively, do the layout-versus-schematic (LVS) verification.
- Execute the design rules check (DRC) of the layout and fix DRC errors. In the VLSI design rules, circuit geometry is specified based on methodology. The unit of measurement can easily be scaled to different fabrication processes as semiconductor technology advances. Each design has a technology-code associated with the layout file. Each technology-code may have one or more associated options added for the purpose of specifying either (a) special features for the target process or (b) the presence of novel devices in the design. This ASIC implementation is based on the 0.18μm CMOS technology.
- Identify nets with antenna rule check (ARC) and perform the advanced DRC

under manufactory requirements. Antenna rules deals with the processes which may cause gate oxide damage such as: (a) expose polysilicon and metal structures, (b) connect to a thin oxide transistor, (c) collect charge from the processing environment (e.g., reactive ion etch), and (d) develop potentials sufficiently large current to flow through the thin oxide. Failing to consider antenna rules in the design may lead to either reduce performance or induce damage in the transistors exposed process. The chip may be totally failure if the antenna rules are seriously violated.

Design implementation is completed with the physical verification on the chip layout which should have DRC and ARC error free. The final chip can now be fabricated. This stage is the black box process in the digital flow for 0.18μm CMOS technology.

The TCM codec chip implemented in this project follows the above ASIC design digital flow. Next section describes the overall architecture of this chip.

4.2 Top Module Architecture of the TCM Codec ASIC Chip

Figure 4.2 shows the overall architecture of the TCM encoder/decoder chip implemented in this project which includes 2D and 4D TCM schemes. In the encoder, the block labelled “tcmencoder2d” performs the 2D TCM encoder architecture described in Section 3.1.1. The block labelled “tcmencoder4d” performs the 4D TCM encoder architecture described in Section 3.1.2.

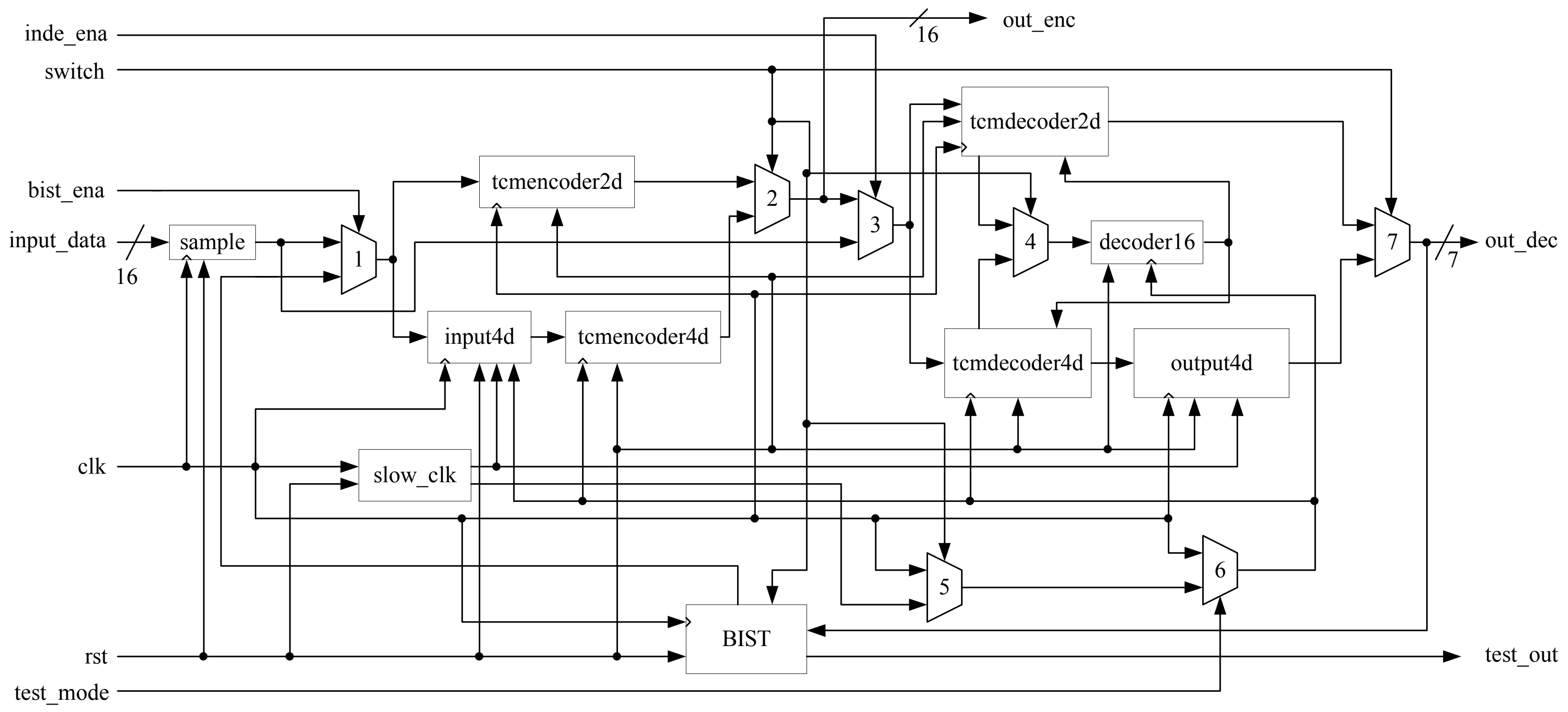


Figure 4.2 Overall ASIC chip architecture of the TCM codec

In the decoder, both 2D and 4D share the same block labelled "decoder16", which executes the convolutional decoding process described in Section 3.2.2. This block works at different timing cycle controlled by a "switch" signal. Meanings of all the I/O signals used in the codec will be described in Section 4.2.1. The block labelled "tcmdecoder2d" performs the de-mapping and the differential decoding processes for the 2D TCM decoder, while the block labelled "tcmdecoder4d" performs the de-mapping and the differential decoding processes for the 4D TCM decoder. Details of these two blocks are described in Section 3.2.2. Section 4.2.2 will describe the remaining blocks in detail.

4.2.1 I/O Signal Description

Figure 4.3 shows the input and output signals for the TCM chip designed in this thesis. Some of the IO signals are not implemented in the finally chip due to the limitation in silicon area provided by CMC.

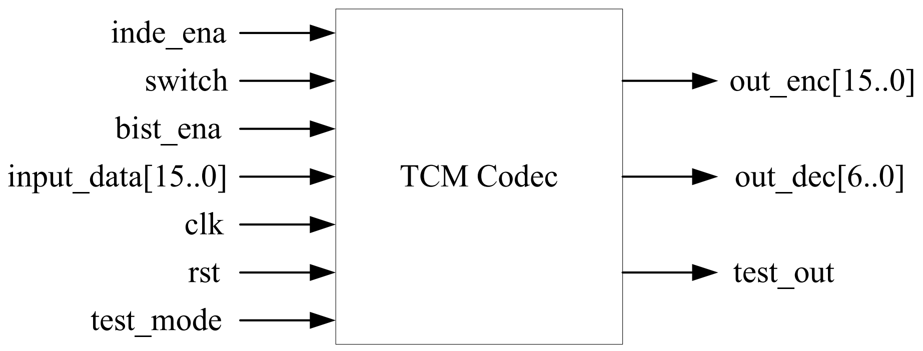


Figure 4.3 I/O signals illustration of the TCM codec

The "inde_ena" is a decoder enable signal which controls the input symbol for TCM decoder. When the control signal "inde_ena" is set to '1', the TCM decoder will decode the symbol sequence coming from external input "input_data", which is a 16-bit data sequence for “tcmdecoder4d” and a low 8-bit data sequence for “tcmdecoder2d”. When “inde_ena” is set to ‘0’, the TCM decoder will decode the symbol sequence coming from the internal signal, which is the output of multiplexer 2 in Figure 4.2.

The “switch” signal controls the function of the TCM codec chip. If the “switch” is ‘0’, the chip works in the 2D TCM scheme; if the “switch” is ‘1’, the chip works in the 4D TCM scheme.

The “bist_ena” is a BIST enable signal which enables or disables the chip in BIST mode. If it is “0”, the output of multiplexer 1 (i.e. input of the encoder) takes 7 LSBs out of the 16 bits “input_data”. If it is “1”, the output takes the pseudo input generated from the BIST block.

The “input_data” is a 16-bit input signal of this chip. For general TCM encoder/decoder application, the external input symbol sequence for TCM encoder is the LSB 7-bits out of 16-bits “input_data” for both the 2D and 4D encoders. In the layout floorplan creating process, since the chip size is decided by PAD limited policy, which means optimizing based on the number of all I/O PAD and power PAD. In order to minimize the number of I/O PAD, the “input_data” signal used in the chip is 16 bits only when the 4D TCM decoder part requires to be tested.

The “clk” is the global clock signal of this TCM codec chip. The “rst” is the reset signal of the chip; this is an active low signal, the system will reset if “rst” is ‘0’.

The “test_mode” signal controls the chip transform its working status between normal codec function and scan test. When the chip is under scan test, all blocks related to the 4D TCM scheme will be forced to work at high frequency. That means if “test_mode” is ‘1’, the clock of all blocks related to the 4D TCM scheme will use an external system clock; if “test_mode” is ‘0’, the clock of all these blocks will use the clock selected by the “switch” control signal. This signal is used to keep the scan test simple. Scan test is used to detect problems occurred at the time of fabricating and packaging.

The “out_enc” is a 16-bit output signal of TCM encoder blocks. It is a mapping signal ready for 2D or 4D modulation. This “out_enc” signal is the output from multiplexer 2 in Figure 4.2 controlled by the “switch” signal. When “switch” is 0, only the 8-LSB out of “out_enc” is available for 2D modulation.

The “out_dec” is a 7-bit output signal of the TCM decoder block. It is the recovered original input signal. This signal sequence should be same as the 7-LSB of “input_data” signal sequence if the chip functions properly.

The “test_out” is an output signal indicating the result of BIST. If it is ‘1’, it shows the TCM codec functional self-test passed. If “test-out” is ‘0’, there are problems existing in the system, which could be design and/or fabrication faults. Section 4.2.2.3 will describe in detail the BIST of this design.

4.2.2 Individual Block Description

For the 2D TCM scheme, the 7-bits input data is directly coded through “tcmencoder2d” block at full-baud rate. For the 4D TCM scheme, the input of encoder spans over two consecutive symbol periods. These two-symbol bits are converted into parallel m-tuples at half of the baud rate. As described in Chapter 3, Section 3.2.2.2, the 4D TCM coding system requires more logic blocks compared to the 2D TCM system. The block “input4d” performs two 2D signals to one 4D signal conversion. The "output4d" performs the reverse process. The "clk_div" implements the half-baud rate clock conversion. This clock is used for all the blocks within the 4D TCM scheme. The next subsections describe these blocks in detail.

4.2.2.1 The Multiplexers

There are seven 2-to-1 multiplexers (MUX) used in this chip top module:

- MUX 1 selects input signal for TCM encoder. It is controlled by the signal "bist_ena".
- MUX 2 selects the output from TCM encoder blocks "tcmencoder2d" and "tcmencoder4d". It is controlled by the signal "switch".
- MUX 3 selects the input signal for TCM decoder. It is controlled by the signal "inde_ena".
- MUX 4 selects de-mapping symbol bits from "tcmdecoder2d" and "tcmdecoder4d", which will be used in the convolutional decoding process. It is controlled by the signal "switch".
- MUX 5 selects the different clock for each TCM scheme, the external input clock (i.e. full-baud rate) is for 2D TCM scheme, test mode and input/output of the 4D scheme; the internal divided clock (i.e. half-baud rate) is used in the 4D TCM scheme. It is also controlled by the signal "switch".
- MUX 6 selects the clock for block "decoder16", "tcmencoder4D" and "tcmdecoder4D". It is controlled by the signal "test_mode". In the case when the system is under the scan test, all the blocks of the chip work at the same clock rate.

- MUX 7 selects the output of TCM decoder blocks “tcmdecoder2d” and “output4d”. It is controlled by the signal “switch”.

4.2.2.2 The Clock Divider

Clock divider is the “div_clk” block shown in Figure 4.2. For the 4D TCM scheme implementation, the encoder and decoder work at half-baud rate, which means the data transmission rate is only the half of the 2D TCM scheme codec. Since the trellis states and structure are same in both 2D and 4D TCM scheme, if the data transmission rates are the same in both TCM codec systems, the throughput of the 4D TCM scheme will be double of the one in the 2D TCM scheme.

If the input clock period is T, in order to get the new clock period 2T to control the 4D TCM codec, a clock divider is required. Figure 4.4 illustrates the signals of clock divider.

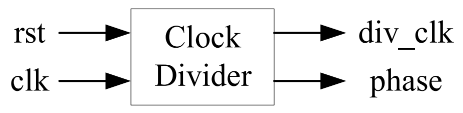


Figure 4.4 Clock divider signals illustration

There is a D-Flip-Flop used to generate “div_clk” and “phase” signals inside the “Clock Divider” block. When “rst” is ‘0’, the DFF is initialized. Else when “rst” is ‘1’, the DFF works at positive edge of clock “clk” to invert the input of DFF. The signal “phase” is used for correcting the half phase deviation occurring when the two consecutive symbols are converted to one symbol in the “input4d” block and one symbol is converted to two consecutive symbols in the “output4d” block. Figure 4.5 shows the waveforms of the clock divider.

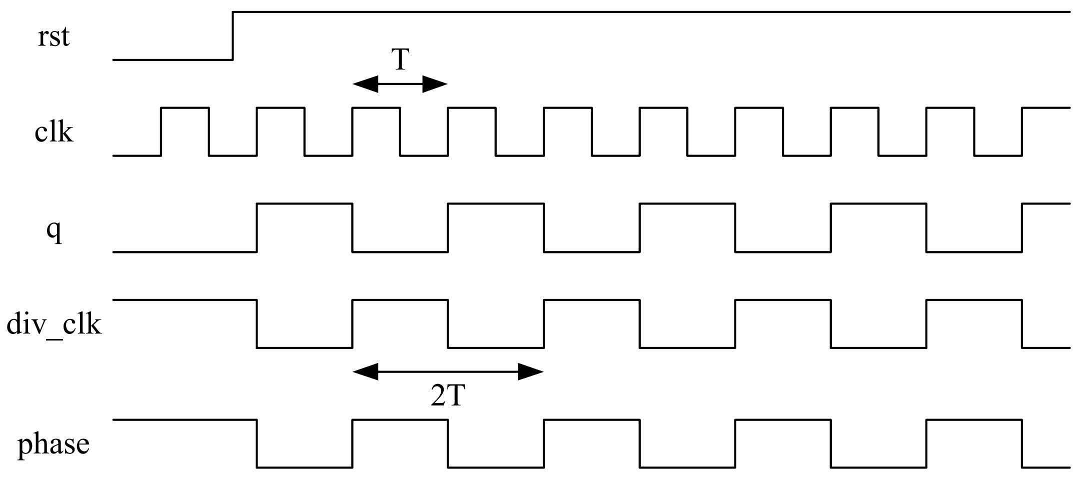


Figure 4.5 The waveform of clock divider

The “div_clk” and “phase” signals are the same but they are used in different locations of the chip. The “div_clk” signal is required to be “clean” to generate the clock tree in the chip layout process.

4.2.2.3 The Built-In Self Test

The “BIST” block is used for functionality self-testing of the chip, including both 2D and 4D TCM schemes. BIST is defined as a design-for-test (DFT) technique in which testing (test generation and test application) is accomplished through the built-in hardware features [28-29].

The BIST technique offers a number of advantages in chip design:

- Fast and efficient: same hardware is capable of testing chips, boards and systems. At the system level, BIST is a cheap testing solution.
- Testing during operation and maintenance.
- Uniform technique for production, system and maintenance tests.
- Dynamic properties of the circuit can be tested at speed.

- Support concurrent testing.
- Can be used for delay testing as it can be used in real time.

However, BIST technique also exposes some disadvantages:

- Silicon area overhead: additional silicon area of the chip must be reserved for BIST circuitry. This increases the cost of the IC.
- Access time: by adding additional test circuitry, it is necessary to add at least one extra level of logic operation between inputs and design chip circuitry, as well as a level logic operation at the output of the chip.
- Requires extra input/output connection: BIST circuitry may require I/O pins to communicate with the outside.
- Correctness is not assured: it is difficult to test the chip hardware under all conditions. Verifying correct operation of such hardware is a difficult issue.

A BIST test structure used in this chip is a Linear Feedback Shift Register (LFSR). The LFSR produces a pseudo random symbol sequence. Therefore it is also called a Pseudo Random Pattern Generator (PRPG). The polynomial used in this BIST is $1+x^{14}+x^{15}$. Figure 4.6 shows a PRPG based on this polynomial.

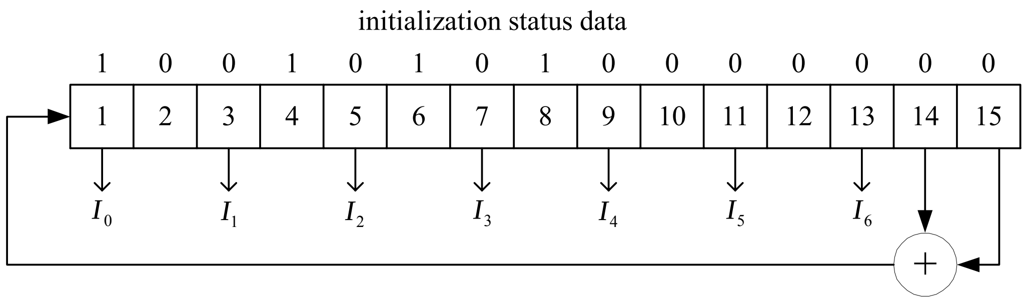

initialization status data
1 0 0 1 0 1 0 1 0 0 0 0 0 0 0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
I_0 I_1 I_2 I_3 I_4 I_5 I_6
+

Figure 4.6 The structure of LFSR

The 7-bit LFSR scrambler can be used to create a pseudo random 7-bit signal sequence. In order to create a longer non-repeat random sequence, a 15-bit LFSR scrambler is used in this project. The length of this random pattern is 2^{15}. The 7 bits test vector, $I_0 I_1 I_2 I_3 I_4 I_5 I_6$ is randomly selected from the 15-bit LFSR and it is fixed on this selection. It will be used as the test input signal sequence of the TCM codec system. The output of the TCM codec will be fed back to the BIST to be compared with the pseudo random symbols created by another randomizer LFSR-2D or LFSR-4D dependent on the "switch" signal. Figure 4.7 shows the architecture of the BIST block.

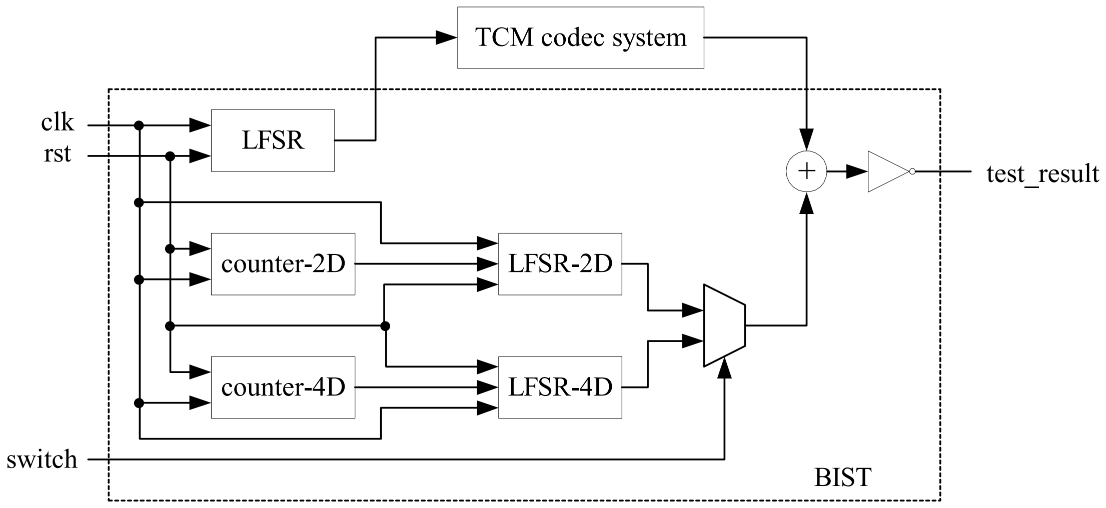


Figure 4.7 The architecture of the BIST block on the TCM codec chip

LFSR-2D and LFSR-4D have the same structure as LFSR. They are initialized at different clock cycle based on the result of the counter-2D and counter-4D, which are the latency of the TCM codec for the 2D TCM scheme and the 4D TCM scheme, respectively. If the output signal "test_result" is "1", the TCM codec chip functions properly; otherwise, the chip fails on its functional self-test.

4.3 Register Transfer Level Code in VHDL

In this research, all register transfer level codes are written in VHDL. VHDL stands for VHSIC (Very High-Speed Integrated Circuit) Hardware Description Language [30]. It can be used for behavioral modeling of designs or for logic synthesis using either behavioral or structural descriptions. Since writing structural circuit descriptions is like trying to describe a circuit using text instead of a schematic editor, the advantage of VHDL is in its behavioral synthesis potential. For complex circuit designs like this TCM codec, writing RTL code in description language is more simple and convenient than drawing schematic circuits.

Based on the architecture shown in Figure 4.2 and the behavior described in Chapter 3 and Section 4.2.2, the RTL codes of the top module and each block of the chip were written in VHDL. All the RTL codes were compiled and simulated using Cadence NC-Sim. Functional simulation result of the design performed as expected and will be described in Chapter 5.

Increasing the speed of the chip is equivalent to shorten the length of its critical signal path. This can be achieved by using a technique called pipelining.

4.4 Pipelining

Pipelining refers to the partitioning of a process into successive, synchronized stages such that multiple processes can be executed in parallel. Depending on the granularity of the process, three types of pipelining techniques can be identified [28]:

- Instruction pipelining partitions processes into stages of instruction fetch-decode, operand-fetch, and execution.

- Intra-functional unit pipelining divides the execution unit (usually a combinational circuit) into several segments of equal delay time.
- Inter-functional unit pipelining involves predefining a sequence of frequently encountered primitive operations such as multiplier or accumulator structure.

In this TCM codec system, the intra-functional pipelining method is used. Extra registers are inserted to divide the execution path into several short paths. Pipelining techniques aim at improving system throughput. It allows digital synchronous system to be clocked at a higher rate.

In digital communication systems, the throughput of the chip is defined as the number of bits that can be transferred by the system per second. Normally, it is the number of bit in data symbols times the system frequency. For example, if 8-bit symbols are processed through a 100MHz communication system, then the throughput of the system is 800MHz (i.e., 800Mbits/second). Generally, system frequency is determined by the propagation delay of the longest path in the pipeline segments (i.e., the critical timing path in the chip). The path starts from one register and ended at the next register. By using pipelining technology, extra registers are inserted into the critical path. The insertion shortens the path and decreases timing of the critical path; this results in an improvement of the system frequency and throughput.

One drawback of the pipelining technique is, while the path gets shorter, the system latency gets longer because extra delays are introduced due to the increase in the number of shift registers. Latency refers to the number of clock cycles that the system takes to respond to the input, which means the time taken when the first input symbol entering the decoder to be restored at the output.

4.5 System Synthesis

The Synopsys design compiler is used to perform the synthesis of the system RTL code. Design compiler is the core of the Synopsys synthesis software products. It provides constraint-driven sequential optimization and supports a wide range of design styles. The design compiler synthesizes a HDL description into a technology-dependent, gate-level design. Using design compiler defines the environmental conditions, constraints, compile methodology, design rules, and target libraries to achieve design goals. The TSMC 0.18μm CMOS technology library is used in this research.

The synthesis process follows these general steps:

- Read in the design and its sub-designs.
- Set design attributes on the top-level design.
- Set realistic timing or area goals for the design.
- Run check-design to verify the design. Identify and correct any errors.
- Perform design compiler optimization.
- Run area and constraint reports to determine whether design goals are met.
- Insert scan circuitry.
- Perform final check and generate gate-level Verilog netlist.

During the synthesis process, the scan chain is added into the chip. Scan chains are routes included on a chip for testing purposes. The Synopsys test compiler substitutes all sequential devices (i.e. flip-flops) with scan equivalents, and then connects them together to form a scan chain. Each scan chain will be reordered during layout place and route process to minimize routing based on layout floorplan placement.

The test compiler will then be used to create a set of test vectors which can detect “Stuck-At-1” and “Stuck-At-0” faults in the chip.

“Stuck-At-1” and “Stuck-At-0” are two models of stuck-at faults for each cell pin in ASIC cell pin test. The measure of test quality is often based on the percentage of ASIC cell pin stuck-at faults that are detected. Cell output faults are interpreted as the output is stuck-at either the logical one or the logical zero state independent of the input condition.

Other features such as vector compaction and fault coverage estimation are also performed. If basic design methods are followed, the fault coverage should obtain above 90%. In industry, the higher the fault coverage, the fewer defective chips will be packaged and placed in products. In this designed chip, the fault coverage achieved to 96%. Reducing the number of defective chips used at the time of initial testing (with the scan-based test) reduces the time and money spent on defective devices. Purpose of scan-based test is to detect problems created at the time of fabricating and packaging; the scan test does not intend to find design faults.

The synthesis process also generates timing constraints of the design and a gate-level netlist; these constraints information are used in Cadence layout tools to produce a chip layout. All the results of simulation and chip layout will be provided in Chapter 5.

CHAPTER 5

RESULTS

This chapter provides the simulation and hardware implementation results of the TCM 2D and 4D codec system. The research was accomplished in four stages:

1. Using MATLAB to simulate the Viterbi algorithm combining with the use of Hamming distance and output look-up table.
2. Using an Altera FPGA device to implement and simulate the codec in both functionality and timing analysis.
3. Using Cadence NC-Sim to simulate system functionality, using Synopsys to synthesize hardware language code, and using Cadence tools to finish chip layout.
4. Using lab equipment to test the final ASIC upon receiving. In addition, hardware implementation results in FPGA and ASIC are also compared.

5.1 MATLAB System Simulation Results

MATLAB is a simulation tool. It stands for "matrix laboratory" because the program is based on matrices to perform all the calculations in the simulation process. MATLAB is an integrated technical computing environment that combines numeric computation, advanced graphics and visualization, and a high-level programming language. It has widely application areas as technical computing, digital signal processing and communication design, control design, image processing, test and measurement, financial modeling and analysis.

In this research, Hamming Distance and output look-up tables are introduced into the Viterbi algorithm to simplify calculation and complexity in the decoding process as shown in Section 2.2.6 and Section 3.2.1. In order to test the feasibility of the algorithms and techniques, MATLAB code was written for the convolutional codec using the methods described in previous chapters. In this code, the Hamming distance and the decoder output look-up tables were built into matrices. The input sequence initialized into two matrices represented the two-bit input of the convolutional encoder. The convolutional code rate was 2/3. Logic functions from the MATLAB library were used in the code.

MATLAB simulation was run using the written code. Figure 5.1 shows the input sequence of the TCM encoder. Figure 5.2 shows the codeword sequence at the output of the encoder. This codeword sequence was then entered to the decoder to decode the sequence and retrieve the original input. The output sequence from the TCM decoder is shown in Figure 5.3.

The written MATLAB code was simulated to verify the use of look-up table in the Viterbi algorithm to decode the convolutional code in the TCM codec. The simulation stored data in one-dimensional matrix and did not consider any timing requirement therefore there were no delays shown in simulation result graphs. This MATLAB simulation is different from hardware description language simulation in which HDL simulation shows timing and latency of the system. MATLAB simulation simply verifies the feasibility of the algorithm and functionality of the codec.

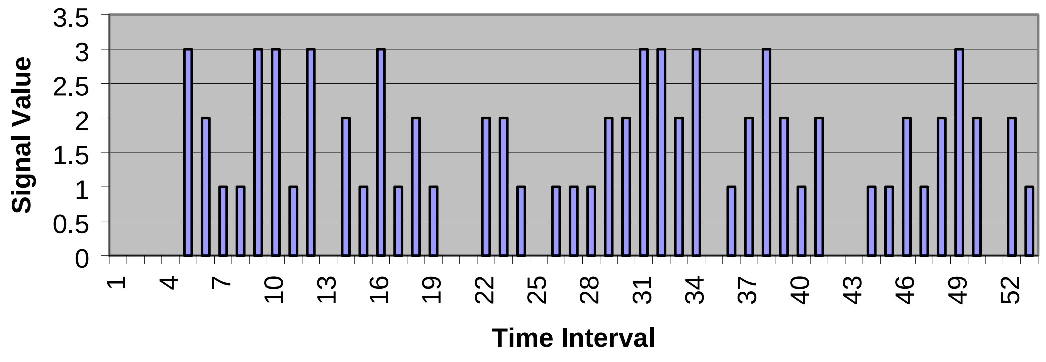


Figure 5.1 Input sequence of the convolutional encoder

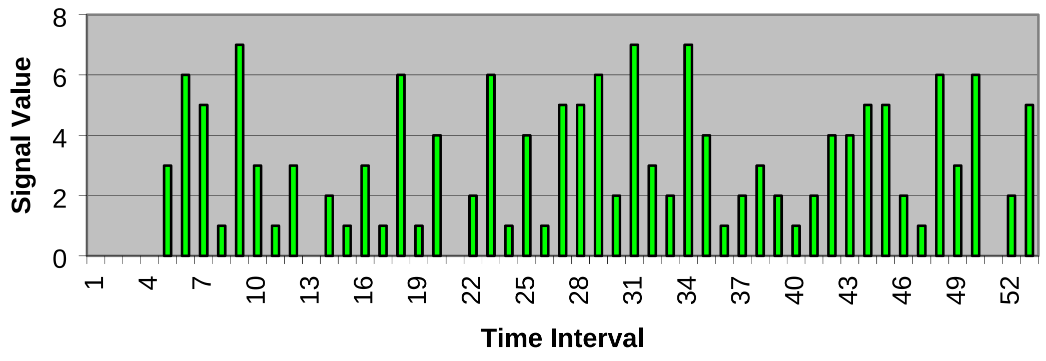


Figure 5.2 Codeword sequence of the convolutional coding

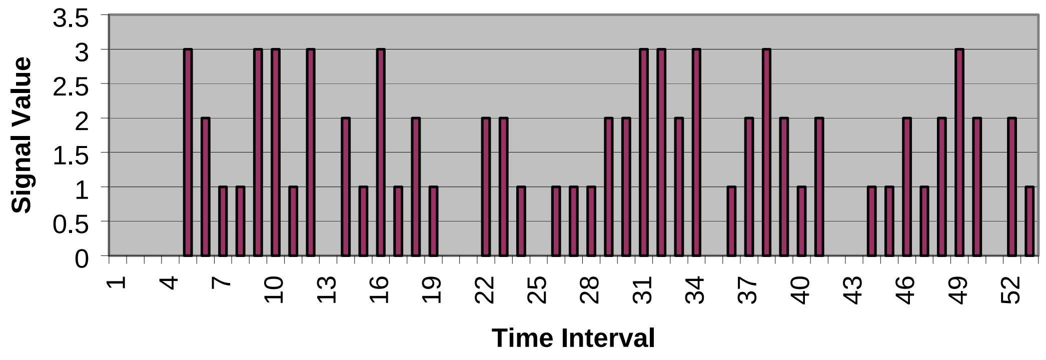


Figure 5.3 Output sequence of the convolutional decoder

The input used two-bit symbol sequence. The output result in Figure 5.3 was the same as the input signal sequence in Figure 5.1. This indicated that the decoder successfully recovered the original signal sequence. It also demonstrated that the simplified algorithm is practical. This simulation confirmed the method and correctness of the look-up tables described in Chapter 3.

Next section describes the implementation results in FPGA, which further confirms the feasibility of hardware implementation of the algorithm and the technique developed in this research.

5.2 FPGA Prototype Implementation Results

As the simplified algorithm was verified through MATLAB simulation, a synthesizable VHDL code was written in particular for the Altera FPGA device compiler. The hardware language code for the TCM encoder/decoder was based on the architecture described in Chapter 3 for 2-dimensional and 4-dimensional TCM codec.

The VHDL code was imported into the Altera MAXPLUS II and compiled. After successfully compiled, a netlist was generated and the design was fit into an automatically chosen FPGA device. The compiler generated a report to indicate the amount of hardware required to implement the codec. Timing analysis was performed to determine the operation frequency (i.e., data rate) of the codec for that particular FPGA device. Finally, a simulation was run based on that device to perform functionality and timing analysis. MAXPLUS II provides a powerful simulation tool which allows the user generate input data and change value of the signal at any time interval of the simulated waveforms. Table 5.1 shows the overall results for 2D and 4D

TCM codec system on Altera FPGA device. The operation frequencies are based on the fastest simulation clock using MAXPLUS II.

Table 5.1 FPGA implementation results

	2D TCM codec		**4D TCM codec**	
	Encoder	Decoder	Encoder	Decoder
Altera FPGA device name	EPF10K20 RC240-3	EPF10K10L C84-3	EPF10K10L C84-3	EPF10K10L C84-3
Number of Logic Cells	21	162	38	432
Amount of Memory	0	64	0	64
Operation frequency	100MHz	20MHz	125MHz	33MHz
Input pins	9	14	16	22
Output pins	8	10	16	17
Data bits	7	8	14	16
Throughput (Bit rate)	700Mbps	160Mbps	1.75Gbps	528Mbps

After confirming the feasibility of the algorithm and architecture from software simulation and FPGA implementation, the TCM codec was implemented into an ASIC using the ASIC design process described in Chapter 4. Next section shows the details of the implementation results.

5.3 ASIC Implementation Results

The TCM codec was implemented into an ASIC using TSMC 0.18µm CMOS technology. This TSMC technology is a single poly, six metal layers process. The technology not only achieves minimum drawn gate length of 0.18µm, but also layout and interconnects design rules that are appropriate to the new generation of chip design and fabrication. This technology has the tightest metal pitches with 0.46µm contacted metal layer 1, 0.56µm contacted metal layers 2 through 5, and 0.90µm on metal layer 6. These pitches provide a higher gate density and more die per wafer, which leads to a lower cost per chip. The 0.18μm CMOS technology offers the optimal combination of density, speed and power to serve a broad range of computing, communications and consumer electronics applications. This technology is suitable for IC design in various microelectronics areas such as analog, low power, RF, and full custom digital circuits. The recommended nominal supply voltages for this technology are 1.8 and 3.3 volts.

In the ASIC implementation, the time delay controls the overall operation frequency of the device. This delay is determined by the longest time of the critical path in the design. Pipelining is a technique to reduce delays of these critical paths. Pipelining was inserted in this design to improve decoding speed and data integrity.

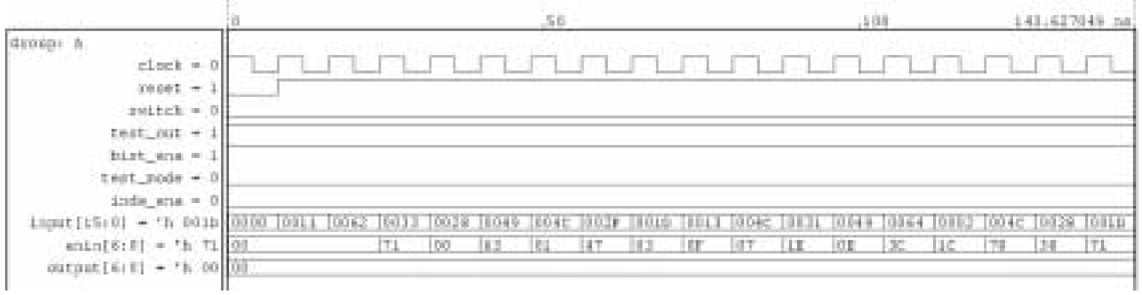
As shown in the ASIC design flow (Figure 4.1, Section 4.1), Cadence NC-Sim was used to complete the RTL code simulation in the first step. Next, Synopsys synthesis tool was used to synthesize the RTL code and generate a Verilog gate-level netlist. This netlist was then used in the layout creation. Cadence physical design planner (PDP or DP) was used to accomplish the floorplan initialization, I/O cells creation, group's placement definition, power planning, and clock tree generation. After the clock tree was successfully generated, another Verilog netlist was generated and this netlist was used to simulate the functionality of the codec again. Both Verilog netlist simulations passed and verified that the design function properly.

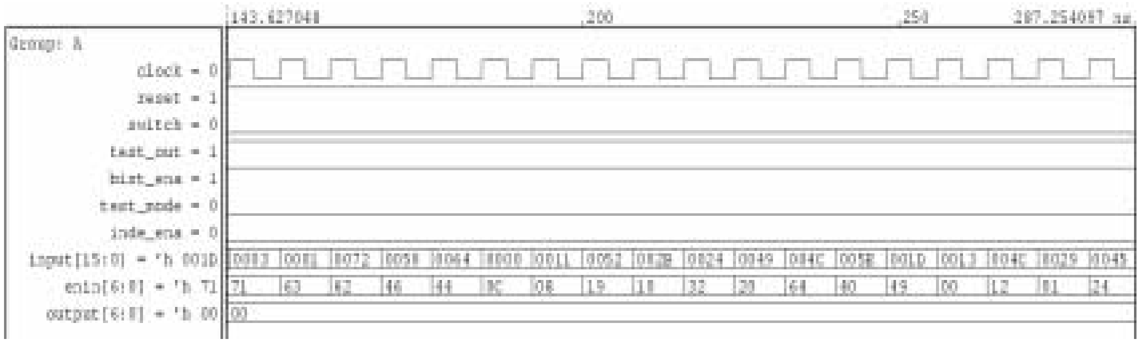
Cadence silicon ensemble (SE) interfaces was used as a tool to route power, clock, and regular nets of the design. After routing, SE exported the design to a design exchange format (DEF) file. This file was then imported into the Cadence design framework II (DFII) environment to perform layout-versus-schematic (LVS) and design rule check (DRC) verifications. The final step on the chip layout is metal slotting, logo adding, and streaming out the design into a GDSII database for fabrication in 0.18μm CMOS. The GDSII stream format is the standard file format for transferring/archiving 2D graphical design data. The file contains a hierarchy of the design structures; each structure contains layout elements such as boundary/polygon, path/plotline, textbox, structure references, and structure array references. These elements are situated on different layers. The stream is a binary format that is platform independent because it uses internally defined formats for its data types. The next sections provide results of each stage in the ASIC implementation flow.

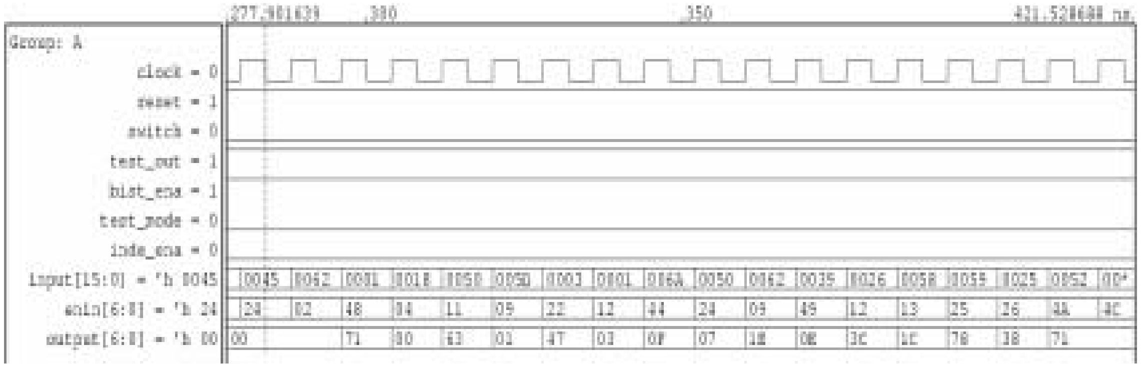
5.3.1 Register Transfer Level Code Simulation

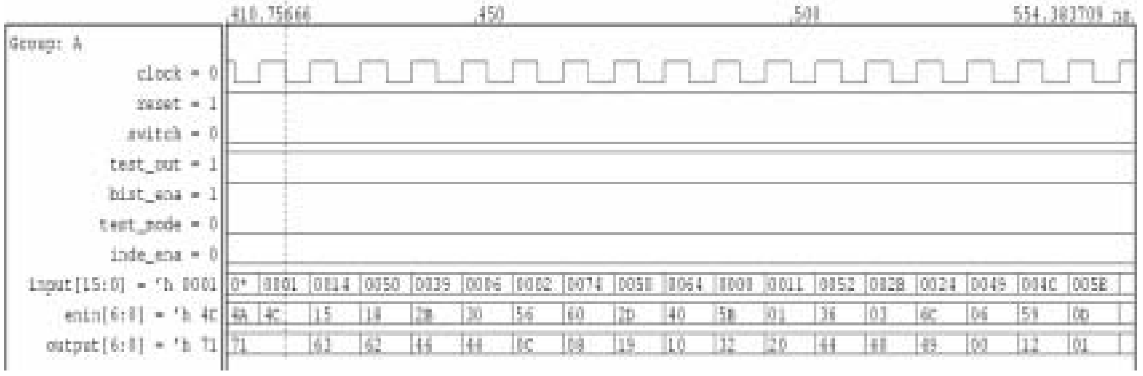
The RTL code for the TCM codec was written based on the architecture described in Section 4.2. The code was simulated using Cadence® NC-Sim. This simulator is an optimum verification solution for system-on-a-chip (SOC) design. It is a flexible and adaptable simulator because it provides the freedom to transparently mix Verilog® and VHDL, and other interface standards. A test bench provided the system clock and reset signals. Inside the test bench code, values of input signals shown in Figure 4.3 were changed each time before compiling the testbench, NC-Sim executed the testbench and provided the following simulation results:

1. **2D BIST simulation** (switch = ’0’, bist_ena = ‘1’)

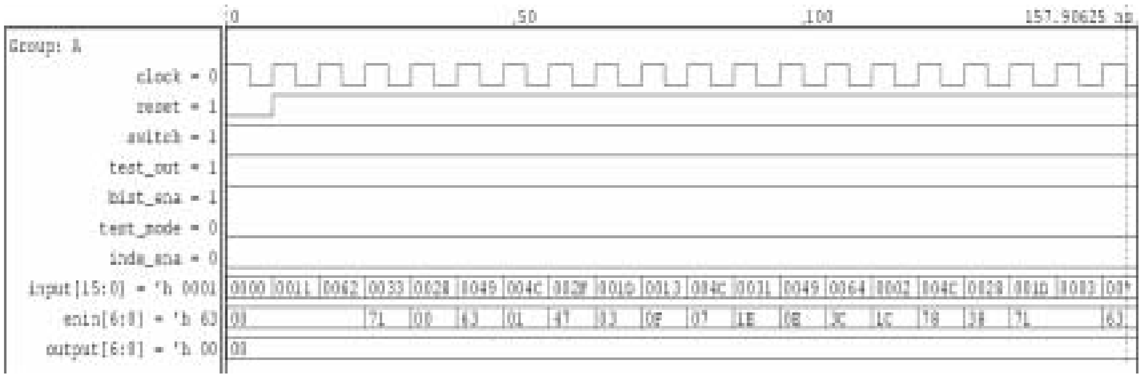


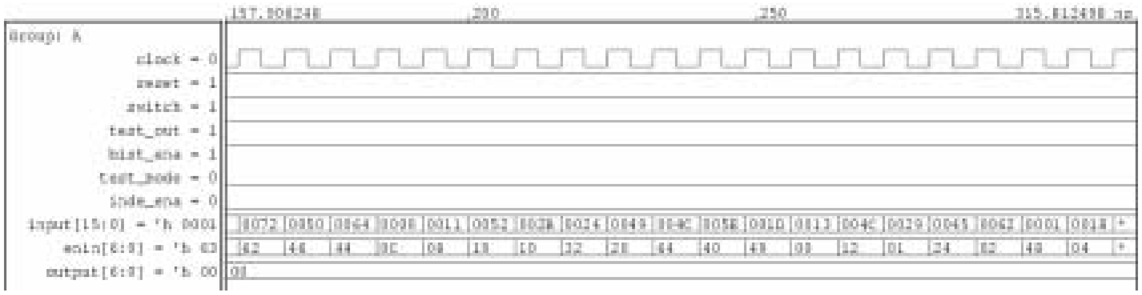


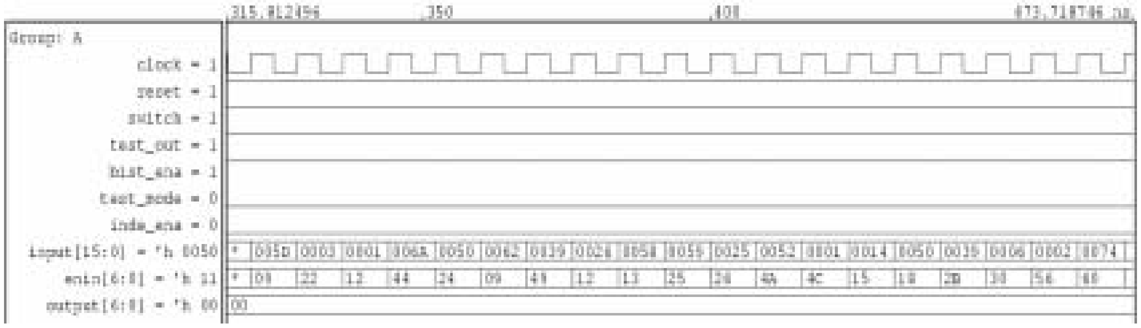


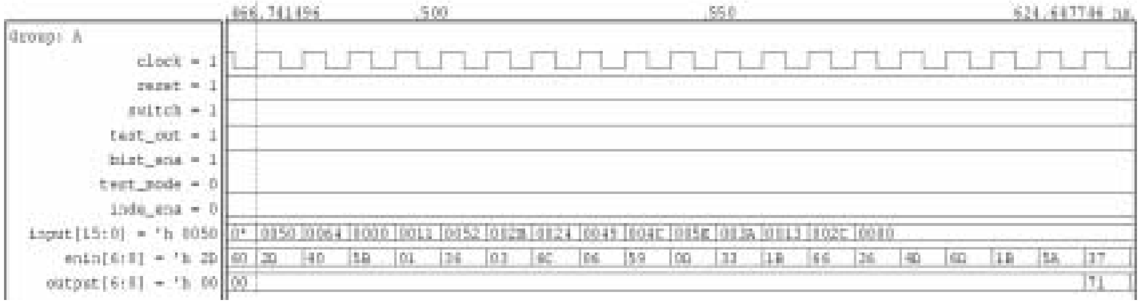


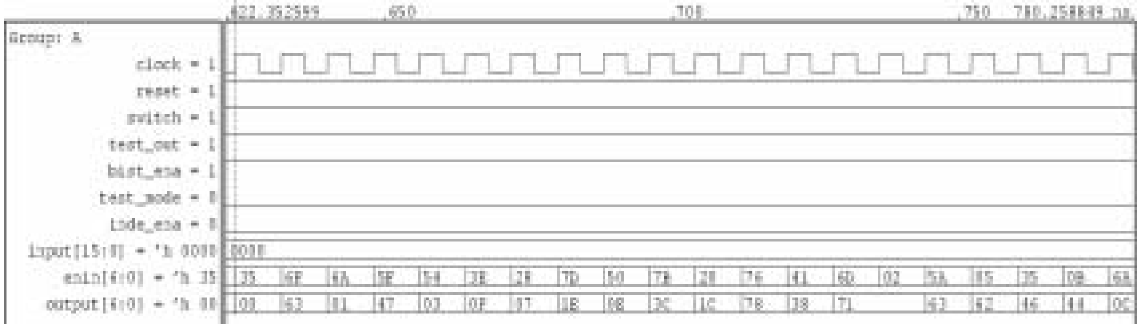
2. 4D **BIST simulation** (switch = '1' , bist_ena = '1')

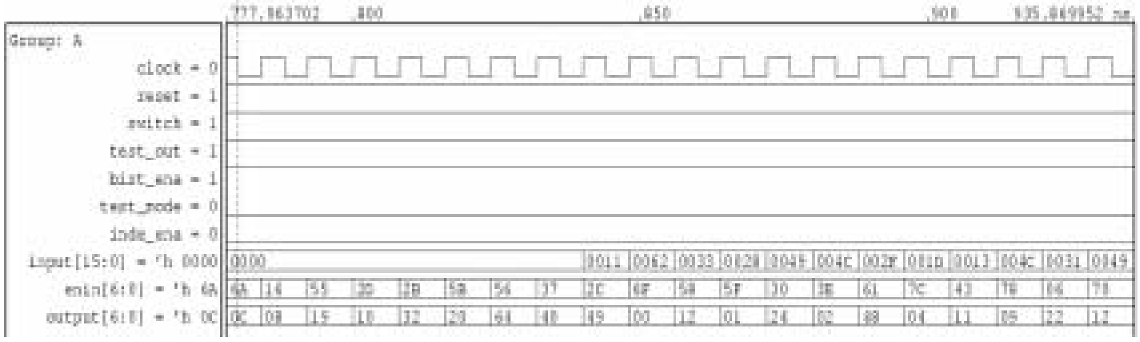



157.908248
200
250
315.812498 ns
Group: A
clock = 0
reset = 1
switch = 1
test_out = 1
bist_ena = 1
test_mode = 0
inde_ena = 0
input[15:0] = 'h 0001
enin[6:0] = 'h 63
output[6:0] = 'h 00

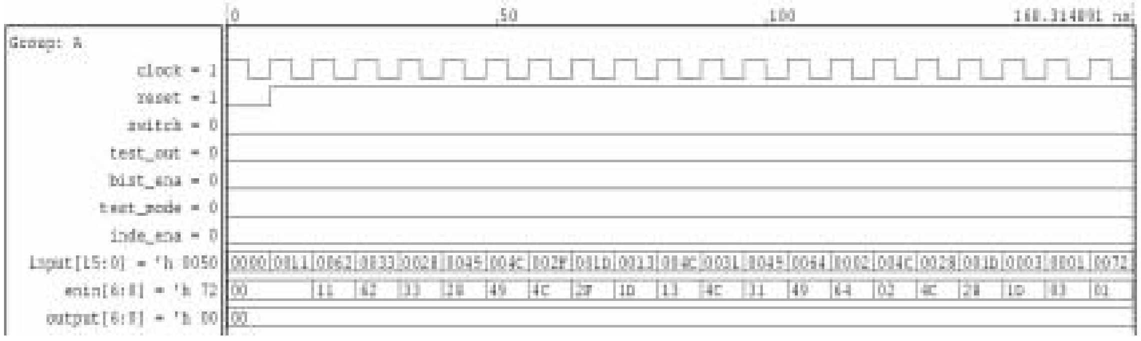

315.812496
350
400
473.718746 ns
Group: A
clock = 1
reset = 1
switch = 1
test_out = 1
bist_ena = 1
test_mode = 0
inde_ena = 0
input[15:0] = 'h 0050
enin[6:0] = 'h 11
output[6:0] = 'h 00

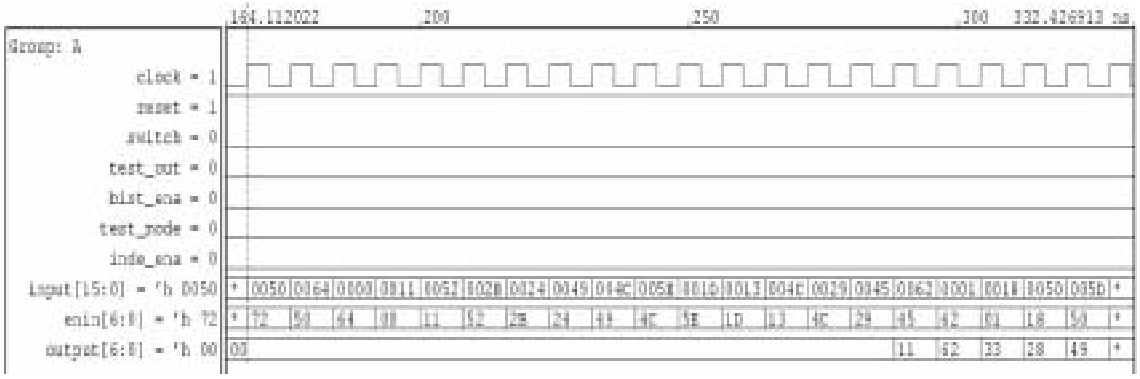

466.741496
500
550
624.647746 ns
Group: A
clock = 1
reset = 1
switch = 1
test_out = 1
bist_ena = 1
test_mode = 0
inde_ena = 0
input[15:0] = 'h 0050
enin[6:0] = 'h 2D
output[6:0] = 'h 00

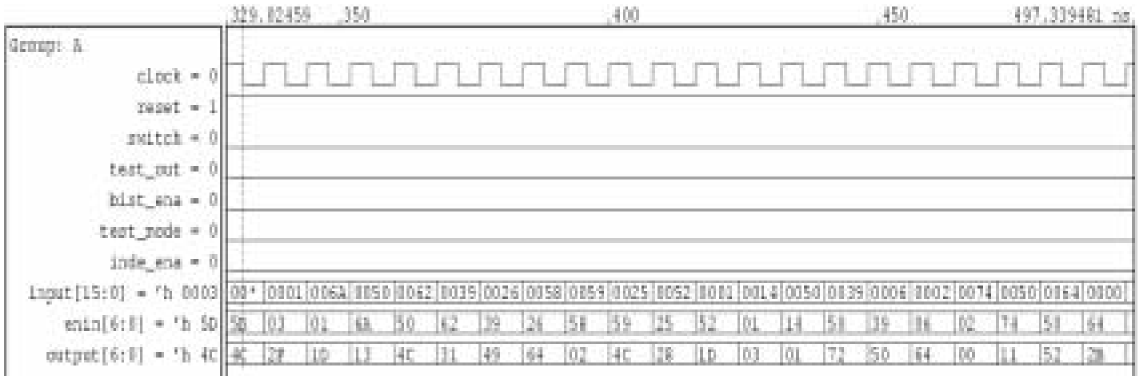

622.352599
650
700
750
780.258849 ns
Group: A
clock = 1
reset = 1
switch = 1
test_out = 1
bist_ena = 1
test_mode = 0
inde_ena = 0
input[15:0] = 'h 0000
enin[6:0] = 'h 35
output[6:0] = 'h 00

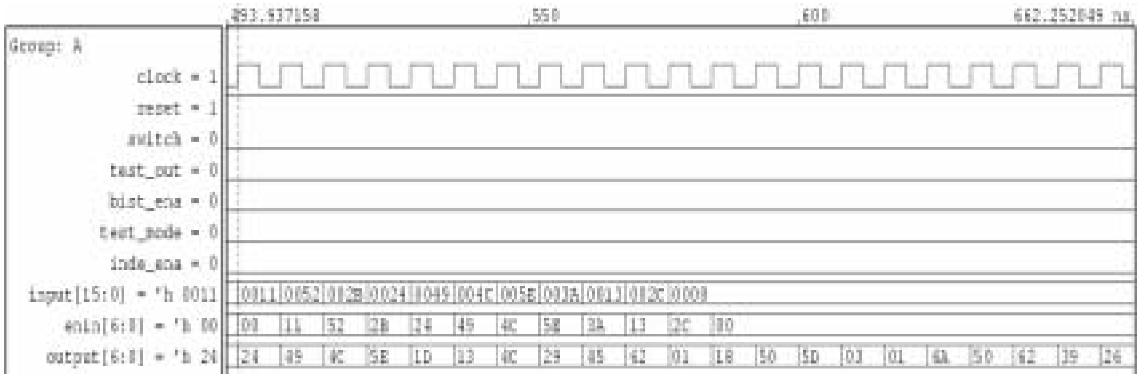

777.863702
800
850
900
935.849952 ns
Group: A
clock = 0
reset = 1
switch = 1
test_out = 1
bist_ena = 1
test_mode = 0
inde_ena = 0
input[15:0] = 'h 0000
enin[6:0] = 'h 6A
output[6:0] = 'h 0C

3. **2D real input simulation** (switch = ’0’ , bist_ena = ‘0’)

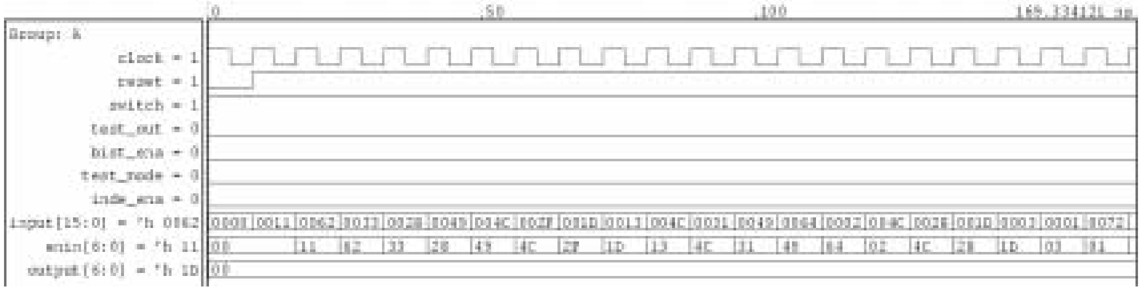


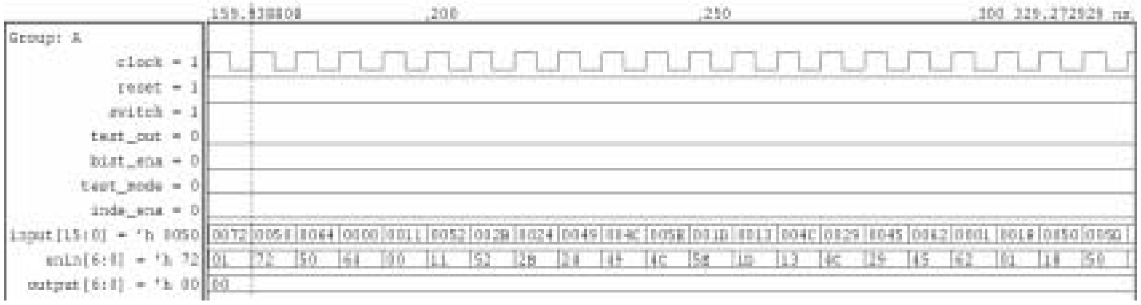


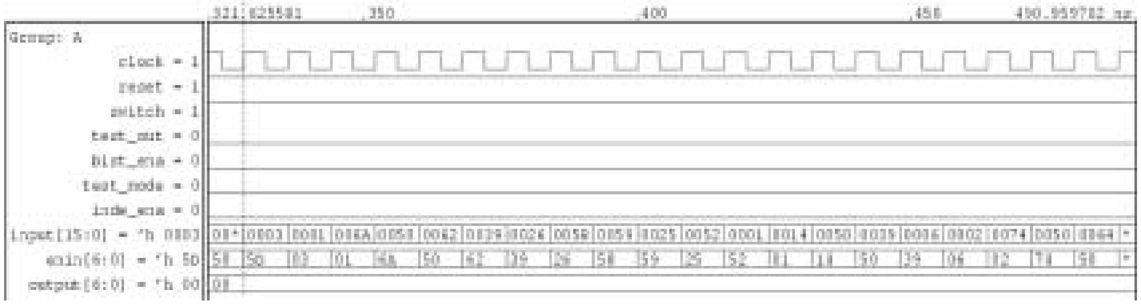


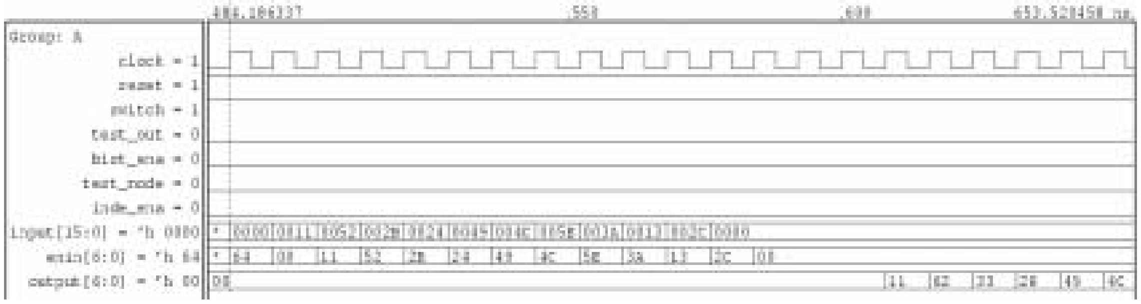


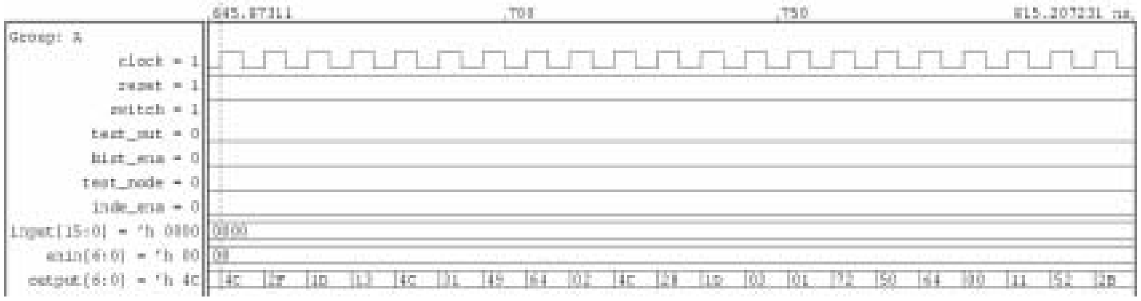
4. **4D real input simulation** (switch = ’1’ , bist_ena = ‘0’)

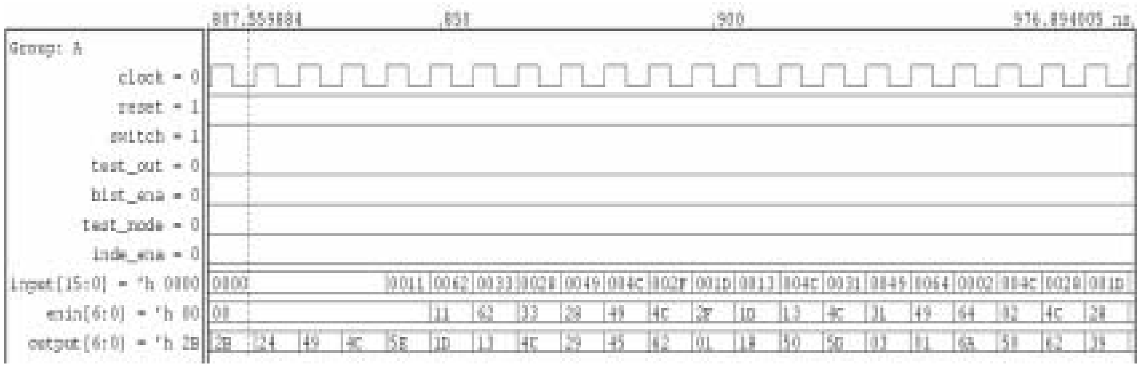












In the signal waveforms from the simulation results, the “enin” signal indicated the output of the MUX 1 in Figure 4.2 described in Chapter 4. The LSB 7-bits out of 16-bit external input data were taken by “enin” when “bist_ena” was ‘0’. When “bist_ena” was ‘1’, “enin” took the random data generated from “BIST” block. All values shown for the “input”, “enin” and “output” were hex numbers.

These simulation results verified the design functionality. The output data sequence (i.e., “output” signal) was the same as the input data sequence (i.e., the “enin” signal) after an encoding/decoding latency. The RTL functional simulation results are summarized in Table 5.2. When BIST is active and functions properly, the “test_out” should be always ‘1’. If “Stuck-At-1” error happened on this signal during fabrication, the scan test should detect the error. As shown in Figure 4.7, the BIST block generates a random symbol sequence therefore it has another extra clock cycle to input the sequence into the encoder.

Table 5.2 RTL code simulation results

Functional simulation	Result	Latency
2D BIST simulation	Pass, test_out = ‘1’	36 clock cycles
4D BIST simulation	Pass, test_out = ‘1’	76 clock cycles
2D real input simulation	Pass, signal recovered after latency	35 clock cycles
4D real input simulation	Pass, signal recovered after latency	75 clock cycles

5.3.2 Register Transfer Level Code Synthesis

After RTL functionality verification, the HDL code for the system was synthesized using Synopsys. Synopsys synthesis tool includes features such as synthesis for virtually any clocking scheme, automatic constraint derivation, embedded point-to-point timing analysis, and industry-supported links to layout. Section 4.5 introduced some information of the synthesis.

As a core of Synopsys synthesis tool, design compiler optimizes logic designs for speed, area, and routability. In the TCM codec synthesis process, this optimization was performed for hierarchical combinational or sequential circuit design descriptions. The design compiler synthesized the circuit and put it in the TSMC 0.18μm CMOS technology. Table 5.3 shows the physical results of the chip top module synthesis before scan insertion.

Table 5.3 Synopsys synthesizes physical results of the chip

Parameters	Value	Remark
Number of ports	30	Not including power and scan pins
Number of nets	1887	Wire
Number of cells	1026	Total logic cell
Number of references	38	Single logic cell
Total cell area (μm^2)	337,265.3125	Combinational and non-combinational area without optimization.

The synthesis area shown in Table 5.3 is for the core area only (i.e., without considering area required for the I/O pads). This area is an estimation result without considering optimization of the cell placement and route. The finalized chip area will be report in next section after the chip layout is generated.

From the synthesis timing report, the critical path of this codec chip was found in the decoder block. A critical path is the longest path between two shift registers. Timing of the critical path decides operation frequency of the entire chip. A 6.9ns data arrival time was reported after synthesis. This time indicated that the highest frequency of 2D TCM codec is 144MHz and 289MHz of 4D TCM codec without any the I/O pad delay (i.e., I/O pad delay is extra). These operating frequencies result in a decoding throughput of 1.008Gbps for a 7-bit symbol sequence in the 2D codec and 2.023Gbps in the 4D codec. However, for the current 0.18μm CMOS technology, there is a delay on the I/O pad in the order of 10ns provided in the design library; therefore the optimal frequency of the chip will be in the range of 59MHz (2D) and 74MHz (4D). Delay of the I/O pads reduces the throughput of real chip to 472Mbps for the 2D codec and 1.184Gbps for the 4D codec. This data rate is still faster than FPGA synthesis results for the decoder shown in Table 5.1. Table 5.4 provides the comparison of the synthesis results between the FPGA and ASIC TCM decoders. The ASIC achieves a two fold improvement in the decoding speed. All these results are simulation results. FPGA device used here is the Altera EPF10K10LC84-3. ASIC implementation is using 0.18μm CMOS technology. The results provided here are FPGA and CMOS technology dependent.

Table 5.4 The synthesis results of FPGA and ASIC TCM decoder

TCM Decoder		ASIC		FPGA
		Without bonding pad	With bonding pad	EPF10K10LC84-3
Operation frequency	2D	144MHz	59MHz	20MHz
	4D	289MHz	74MHz	33MHz
Throughputs (Bit rate)	2D	1.008Gbps	472Mbps	160Mbps
	4D	2.023Gbps	1.184Gbps	528Mbps

5.3.3 Layout Generation

After synthesis, the design is converted to a gate-level Verilog netlist from the RTL code. A Verilog test bench was written to generate data and timing in order to run the gate-level netlist simulation. The netlist of this chip was tested with a Verilog test bench. Simulation results obtained were the same as the results in Section 5.3.1. This outcome verified functionality of the gate netlist.

The successfully simulated gate-level Verilog netlist was imported into Cadence PDP for physical layout. The physical placement started with the creation of a design floorplan. Creating a proper I/O floorplan is one of the most critical stages in the digital design process. At the floorplan creation stage, power pads require to be added to the design. As shown in Figure 4.2, a total of 46 Input/Output pins are required, not including scan/BIST I/O pins. After scan insertion, there are three more I/O pins added for scan test which are the scan test input (test_si), the scan test enable (test_se) and the scan test output (test_so). Considering VDD and VSS pins necessary to power the chip, 4 pairs of ring power pads and 4 pairs of core power pads were inserted. The number of ring power pads is determined by a rule of thumb of one pair of power ring for every 4-6 output pins. In addition, using the rule of thumb of 1mA per micron of metal width and 40-micron wide power connections are used, four pairs of core power pads should allow for an average flow of 160mA of current to the core. This comes to the total of 65 I/O pads required for the chip. Since the chip area was based on the pad-limited policy and the area granted by CMC was not sufficient for all these 65 pads, 16 output pins of the encoder were eliminated in the final chip lay out for fabrication. At the final stage, only 49 pins were put in the ASIC, including 30 I/O pins, 3 scan test pins and 16 power pins.

In the floorplanning, a default group of cells was created. An I/O ring is connected by abutment and the placement sites for all the cells are defined. After floorplanning, PDP uses forward-annotated timing information from Synopsys synthesis to place the core cells. This placement is optimized. Hence the core area of all the cells connection is smaller than the total cell area reported in Table 5.3. Once the cells were placed, a clock tree was created at this stage. Creating the clock tree is to add clock buffer cells and nets to create a balanced clock tree which meets timing parameters specified in the synthesis.

At this point, the PDP generated a “golden netlist”. The netlist was then simulated again to ensure the clock tree generation did not alter the netlist. This “golden netlist” was used as a schematic when the final layout was verified with automatic layout-versus-schematic (LVS) at the end of the design cycle. Then the placed design was imported into the silicon ensemble environment to route the power, clock, and regular nets of the design.

After placed and routed, the Cadence DFII tool was used to perform LVS and DRC check; both the LVS and DRC tests passed at this step. Then the GDSII format file was streamed out and sent to CMC to perform the ARC and DRC checks. Finally, bonding pads were added to the layout. Figure 5.4 shows the chip layout before sending to TSMC for fabrication.

The optimized IP core layout dimension is 971.423μm x 1138.56μm (an area of 1.1mm^2 shown in the centre of Figure 5.4). After adding bonding pads, the final chip layout has a dimension of 1591.6μm x 2091.6μm (or an area of 3.3mm^2). The core area is 33.22% of the final chip (i.e., a utilization of 33%). Most of the area in the layout is used for routing the 49 I/O and bonding pads.

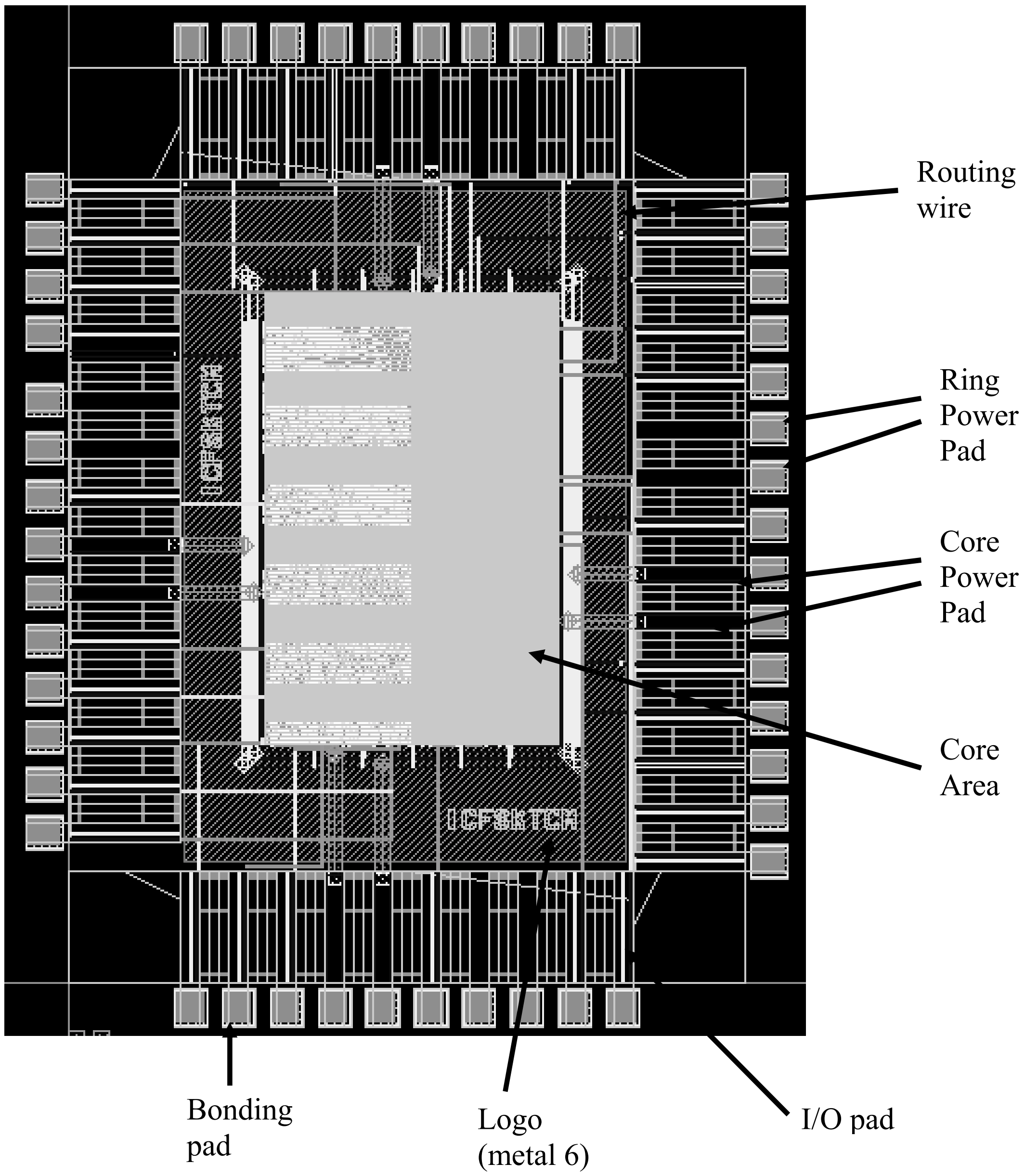


Figure 5.4 Final layout of the TCM codec ASIC

5.4 The Fabricated TCM Codec

Figure 5.5 is a micrograph of the fabricated ASIC received from CMC. This photograph was taken from a die. The picture shows the structure is similar to the final layout in Figure 5.4.

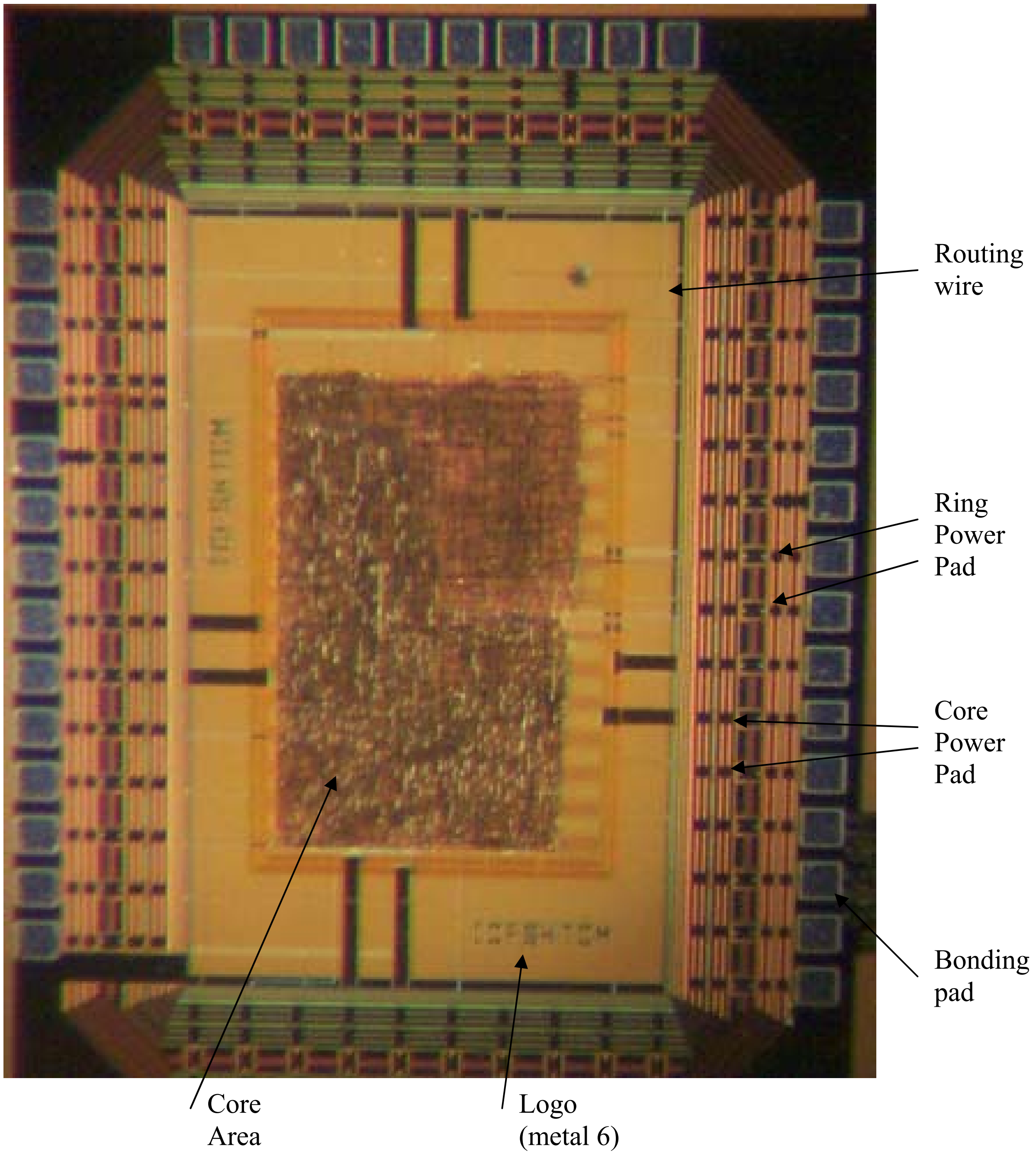
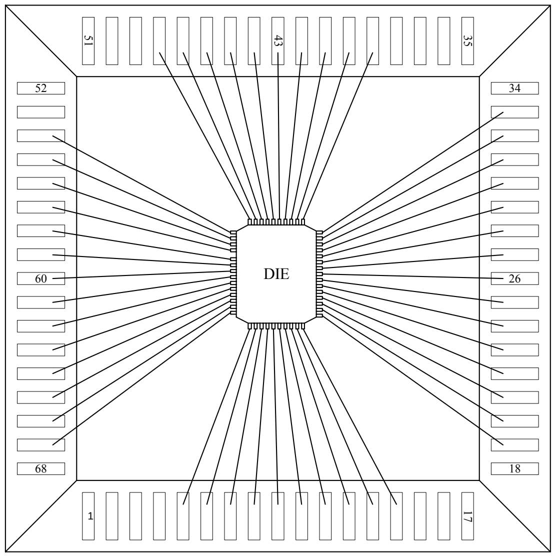
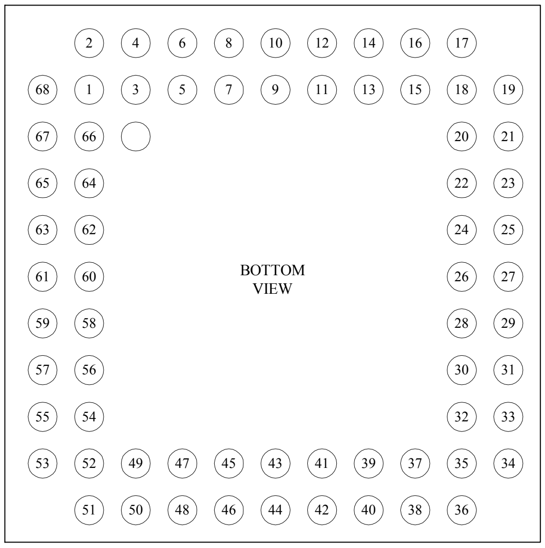


Figure 5.5 Photograph of the TCM codec chip with bonding pads

The fabricated TCM codec ASIC was packaged using the standard 68CPGA package. The 68CPGA package has 68 pins with through hole pin type. The cavity size is 8.89mm x 8.89mm or 350mils x 350mils. Figure 5.6 shows the 68CPGA-bonding diagram and pin out diagram. The third empty circle in the bottom view left top corner indicates pin #1 location.



Bonding diagram



Pin out diagram (bottom view)

Figure 5.6 68CPGA pin bonding and pin out diagram

5.5 Testing Results

Functional testing aimed to determine the ASIC functionality which includes power-up self-test (i.e., to verify the chip is working using BIST), decoder function test and encoder/decoder function tests. These tests were performed in the lower clock rates (less than 10MHz). The tests were set up using an FPGA and various testing equipment. Figure 5.7 shows the test set-ups. The FPGA generated data and signals required to operate the ASIC. The adaptor was used to convert the TTL outputs from the FPGA (i.e., 5V) to the 3.3V inputs of the ASIC. A logic analyzer was used to capture the input and output waveforms. Power consumption was measured to determine the ASIC power requirement. The average power consumption was 19.8μW when connected to a 3.3V power supply. However, this measurement does not indicate the actual power consumption because the faster the device run, the more power it draws.

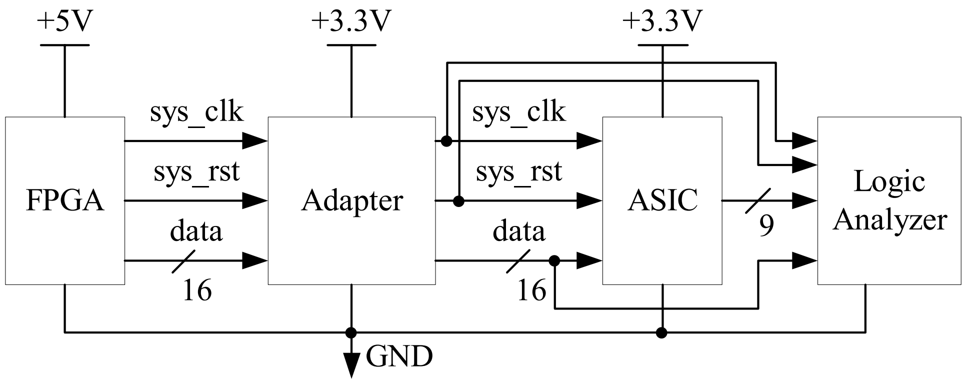


Figure 5.7 Functional test set-ups

The FPGA was used to generate the clock and reset signal for the ASIC. A 16-bit counter implemented in the FPGA was used to generate the data sequence. The data sequence was recovered from the output of the ASIC. This indicated that the data sequence go through the encoder and the decoder; the data was encoded and then decoded correctly by the codec. Unfortunately, at higher frequency, the output of the codec was not stable. In addition, the BIST was not passed when the chip was powered up and the switches were set properly for the built-in self-test. Figure 5.8 and Figure 5.9 show the signal waveforms captured by the logic analyzer. The waveforms show the results for both 2D and 4D functional testing when using the input data sequence generated by the FPGA.

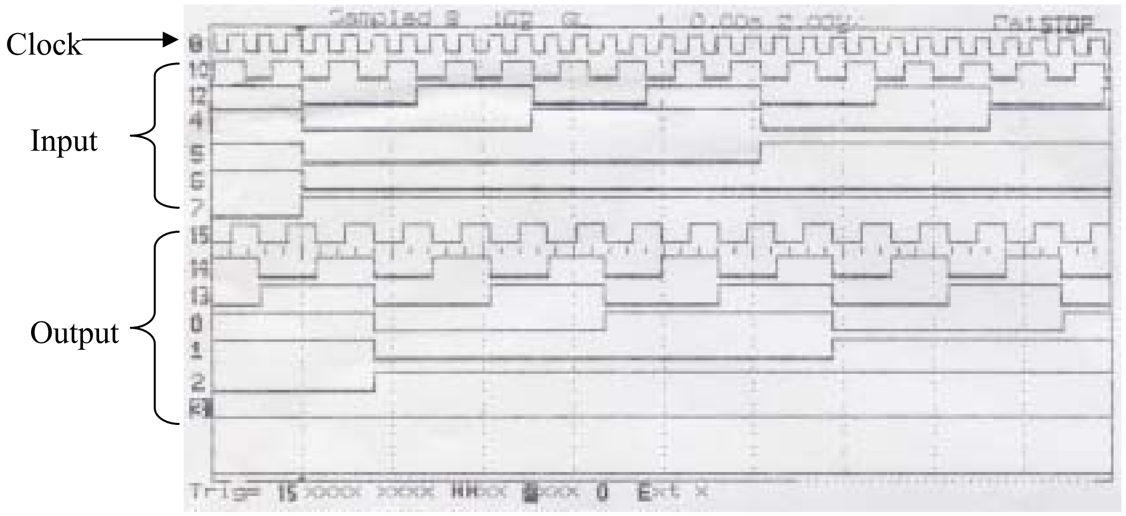

Clock
Input
Output
Sampled @ 1G2 GL 0.00s 5.00%/
Pat STOP
Trig= 15 XXXX XXXX HHXX XXXX 0 Ext X

Figure 5.8 Functional testing for 2D scheme

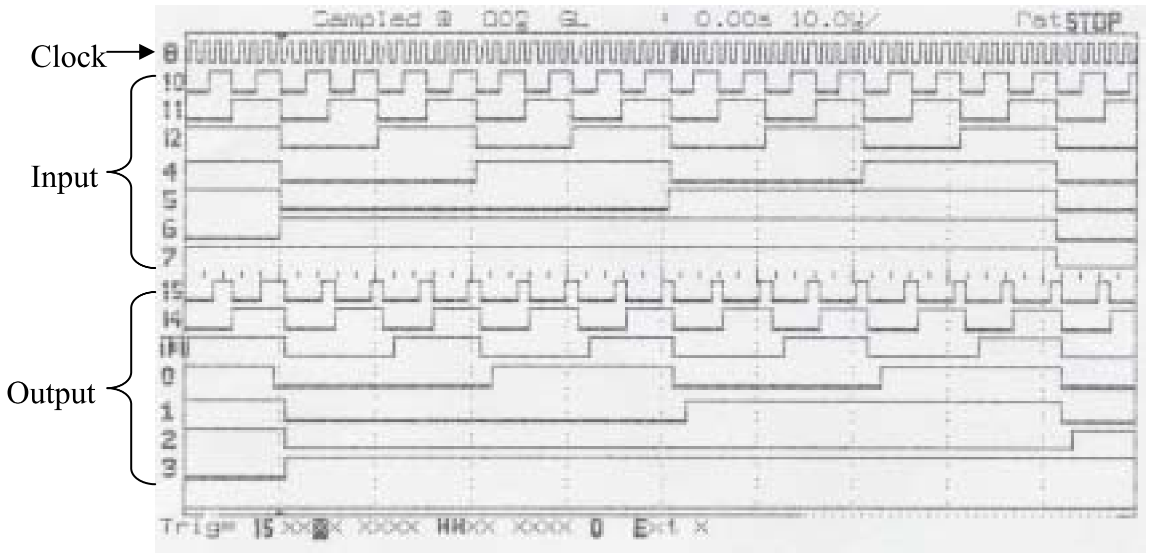

Clock
Input
Output
Sampled @ 002 GL 0.00s 10.0%/
Pat STOP
Trig= 15 XXXX XXXX HHXX XXXX 0 Ext X

Figure 5.9 Functional testing for 4D scheme

The functional testing of the ASIC was performed at a 3.125MHz clock. Because of the stability problem of the testing results, the chip could not be tested fully to provide timing results. The high frequency testing was not performed successfully at this time.

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

This chapter provides the research summary, concludes the thesis, and recommends future investigations for this project.

6.1 Research Summary

This research introduces concepts and applications of the trellis-coded modulation. The research focuses on algorithm simplification and hardware implementation of a high-speed TCM encoder/decoder; the device is an essential part to build a complete wireless communications system. The research is finalized with the accomplishment in the design of a TCM codec ASIC. The codec was fabricated using current available technology (the 0.18μm CMOS). This research studies methods that used in TCM codec system, introduces and develops the use of look-up tables in the Viterbi algorithm. The research also provides architecture of a high-speed TCM codec chip for both 2-dimensional and 4-dimensional mapping used in the 256QAM system. The architecture can be modified to extend to the other modulation techniques.

The research started with the developing of the look-up table concept and introduced it into Viterbi algorithm to decode the convolutional codes. The modified algorithm was then simulated in MATLAB to verify the algorithm feasibility. MATLAB simulation of the convolutional encoder/decoder successfully recovered input signals. Once the simulation results described in Chapter 5 confirmed the methods was feasible, hardware description language code was then written to describe the codec in order to implement the algorithm in hardware.

VHDL register transfer level codes were used to implement a standard TCM encoder and the novel architecture of the TCM decoder into an FPGA prototype. First, the TCM codec was designed for 2-dimensional modulation including a 16-state, radix-4, rate=2/3 convolutional codec and implemented using Altera FPGA developing tool, the MUXPLUS II. Using this FPGA developing tool, RTL code of the TCM codec was synthesized, compiled, fitted and simulated successfully into FPGA devices to verify the algorithm and show that the algorithm can be implemented successfully into hardware. The compilation and simulation results in Altera MAXPLUS II further confirmed the advantage of using look-up tables to increase speed of the decoding process, to increase decoding throughput, to reduce hardware complexity, and again, to verify the feasibility of the novel architecture to decode convolutional codes in TCM.

Finally, VHDL RTL code was optimized for ASIC implementation. At this time, a top level of the chip fulfilled the TCM codec including the 2-dimensional and 4-dimensional parts was synthesized; a built-in self-test block was also included into the chip for power-up testing purpose. Chapter 4 describes in details the overall architecture of the implemented TCM codec chip into ASIC. The RTL code was then simulated and synthesized in ASIC developing tools, the Cadence and Synopsys. After successfully simulated the chip functions and timing, an IC layout was created using Cadence digital layout tool for 0.18μm CMOS technology. All the necessary stages required in the design and verify layout were carefully considered to ensure a functional ASIC. The area of final chip layout is 3.33mm^2 including 49 bonding pads. Thank to the support from CMC, the chip was successfully designed and fabricated through TSMC. The final device can be tested to verify its timing and functionality before inserting into a complete communications system.

The functional testing of the fabricated chip was completed at a low data rate. The built-in self-test did not executed successfully. The input signal into the codec was recovered during the test; however the chip was not working stable at high frequencies. Due to the instability of the chip at high data rate, the timing testing has not been accomplished.

6.2 Conclusions

The TCM codec chip implemented in this research utilizes the advantages of simplification and speed enhancement through the use of look-up tables. In addition, parallel processing and pipelining are also used to further increase decoding throughputs. The final TCM codec includes a TCM encoder with mapping for 2-dimensional and 4-dimensional modulations and a TCM decoder with a 16-state, radix-4, rate=2/3 Viterbi decoder. The implementation in 0.18μm CMOS technology yields a simulation decoding iteration rate of 289MHz under mapping for 4-dimensional modulation condition. This operating frequency converts to a decoding data rate of over 1Gbs. The real speed of the chip is less than 289MHz due to the long delay of the available bonding and I/O pads in the fabrication and packaging processes at TSMC.

The research verifies that the use of look-up table will eliminate the circuits which are used to calculate branch cost and path cost in the Viterbi decoder. Look-up tables not only simplify the circuit but also reduce the core area of the device. The designed decoder employs parallel structure in its architecture. This structure requires higher silicon area than serial structure; however it efficiently increases decoding speed. In addition, the rational use of shift registers in memorizing the delay state conjunction and the realization of pipelining achieve algorithm optimization and increase decoding iteration rate.

As mentioned, one of the goals in the design of this codec is to integrate the codec into a single chip system (i.e., SOC). The core of the ASIC can be used as an IP core to incorporate into future development of any high-speed SOC applications using TCM.

6.3 Future Work

There is work to be considered in future research. The first thing is to further testing the timing of the fabricated ASIC. The timing testing is to determine the maximum optimum bit rate. More sophisticated equipments are required in this test; these testing equipment expect to operate at high speed which is in the range of 0.5Gbs to 1Gbs.

Furthermore, in this thesis, the Viterbi algorithm uses the free Hamming distance to calculate the cost function instead of the free Euclidean distance. The reason to use Hamming distance is already explained in the thesis. If a look-up table based on Euclidean distance is used, the architecture of this chip requires to be redesigned. For the use of Euclidean distance, received signal will be processed by the receiver which combines both demodulation and decoding in a single stage; the modulation architecture of TCM needs to be inserted into the codec. As the results, the choice of the code and the choice of the signal constellation must be considered together; the detection process will involve soft decisions rather than hard decisions. The use of Euclidean distance look-up table should yield a code performance of 2dB over the counterpart Hamming distance.

REFERENCES

[1] Stephen G. Wilson, “Digital Modulation and Coding,” Prentice Hall, Inc., 1996.

[2] G. Ungerboeck, “Channel Coding with Multilevel/Phase Signals”, IEEE Transactions on Information Theory, Vol. IT-28, pp. 55-67, No.1, January 1982.

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

[4] G. Ungerboeck, “Trellis Coded Modulation with Redundant Signal Sets, Part II: State of the Art”, IEEE Communication Magazine, Vol. 25, pp. 12-21, No.2 February 1987.

[5] Michael A. Bree, David E. Dodds, Ronald J. Bolton, Surinder Kumar, and Brain L. F. Daku, “A Modular Bit-Serial Architecture for Large-Constraint-Length Viterbi Decoding,” IEEE Journal of Solid-State Circuits, Vol. 27, No. 2, pp. 184-190, February 1992.

[6] Peter J. Black and Teresa H. Meng, “ A 140-Mb/s, 32-State, Radix-4 Viterbi Decoder,” IEEE Journal of Solid-State Circuits, Vol. 27-12, pp. 1877-1885, December 1992.

[7] V. S. Gierenz, O. Weiss, T. G. Noll, I. Carew, J. Ashley, R. Karabed, "A 550 Mb/s Radix-4 Bit-Level Pipelined 16-State 0.25-µm CMOS Viterbi Decoder," IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'00), p.195, July 10 - 12, 2000, Boston, Massachusetts.

[8] Olaf J. Joeressen and H. Meyr, “A 40 Mb/s Soft-output Viterbi Decoder,” IEEE Journal of Solid-State Circuits, Vol. 30, No. 7, pp. 812-818, July 1995.

[9] S. Bitterlich and H. Meyr, “Efficient Scalable Architectures for Viterbi Decoders,” in International Conference on Application Specific Array Processing (ASAP), Venice, Italy, October 1993.

[10] Stefan J. Bitterlich, Bert Pape, H. Meyr, “Area Efficient Viterbi-Decoder Macros,” Proceedings of the 1994 European Solid-State Circuits Conference (ESSCIRC'94), Ulm, Germany.

[11] Lee-Fang Wei, “Trellis-Coded Modulation with Multidimensional Constellations,” IEEE Transactions on Information Theory, Vol. IT-33, No. 4, July 1987.

[12] G. D. Forney, Jr., et al., “Efficient modulation for band-limited channels,” IEEE J. Select. Areas Commun., Vol. SAC-2, pp. 632-647, Sept. 1984.

[13] R. Fang and W. Lee, “Four-dimensionally coded PSK system for combating effects of severe ISI and CCI,” in Proc. IEEE Globe-com Conv. Rec., pp. 30.4.1-30.4.7, 1983.

[14] S. G. Wilson et al., “Four-dimensional modulation and coding: An alternate to frequency-reuse,” in Proc. IEEE ICC Conv. Rec., pp. 919-923, 1984.

[15] A. R. Calderbank and N. J. A. Sloane, “Four-dimensional modulation with an eight-state trellis code,” AT&T Tech. J., Vol. 64, pp. 1005-1018, May-June 1985.

[16] Digital Audio-Visual Council (DAVIC), “The DAVIC 1.2 Specifications,” DAVIC, Geneva, Switzerland, 1997.

[17] Claude E. Shannon, ``A mathematical theory of communication," Bell System Technical Journal, Vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[18] Man Young Rhee, "Error-correcting Coding Theory," McGraw-Hill Companies, Inc., 1989.

[19] Michelson, A. M., and Levesque, A. H., "Error-Control Techniques for Digital Communication," John Wiley & Sons, 1985.

[20] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," IEEE Trans. Info. Theory, 13(2), pp. 260--269, 1967.

[21] John G. Proakis, "Digital Communications," McGraw-Hill Companies, Inc., 2001.

[22] Bernard Sklar, "Digital Communications, Fundamentals and Applications," Prentice Hall, Inc., 2001.

[23] William G. Chambers, "Basic of Communications and Coding," Oxford University Press, New York, 1985.

[24] Ezio Biglieri, Dariush Divsalar, Peter J. McLane and Mavin K. Simon, "Introduction to Trellis-Coded Modulation with Applicatins," MacMillan Publishing Co, 1991.

[25] E. Zehavi and J. K. Wolf, "On the Performance Evaluation of Trellis Codes," IEEE Transactions on Information Theory, Vol. IT-32, No.2, pp. 196--202, March, 1987.

[26] Mansoor A. Christie, “Viterbi Implementation on the TMS320 C5x for V.32 Modems”, Digital Signal Processing Application-Semiconductor Group, Document #SPRA099.pdf, Texas Instruments Incorporated, Texas, 1996.

[27] A. Dinh, R. Mason and J. Toth, “High-speed V.32 Trellis Encoder/Decoder Implementation using FPGA,” IEEE International Symposium on Circuits and Systems (ISCAS’99) Proceedings, Orlando, Florida, pp. IV-295 to IV-298, May 30-June 2, 1999.

[28] Steven S. Leung, and Michael A. Shanblatt, “ASIC System Design with VHDL: A Paradigm”, Kluwer Academic Publishers, 1990.

[29] Texas Instruments SCTA043A, “Built-In Self-Test (BIST) Using Boundary Scan,” December 1996.

[30] Charles H. Roth, Jr., “Digital Systems Design Using VHDL,” PWS Publishing Company, 1998.

[31] CMC, “Tutorial on CMC’s Digital IC Design Flow”, document ICI-096, V1.3, May 2001.