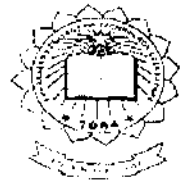




P- 3459



**FPGA BASED CONVOLUTIONAL ENCODER AND VITERBI  
DECODER**

**By**

**SANKARI. K. N.**

**Reg. No. 0920106012**

**of**

**KUMARAGURU COLLEGE OF TECHNOLOGY**

**(An Autonomous Institution affiliated to Anna University of Technology, Coimbatore)**

**COIMBATORE - 641049**

**A PROJECT REPORT**

*Submitted to the*

**FACULTY OF ELECTRONICS AND COMMUNICATION  
ENGINEERING**

*In partial fulfillment of the requirements*

*For the award of the degree*

**of**

**MASTER OF ENGINEERING**

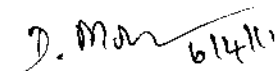
**IN**

**APPLIED ELECTRONICS**

**APRIL 2011**

## BONAFIDE CERTIFICATE

Certified that this project report entitled "**FPGA BASED CONVOLUTIONAL ENCODER AND VITERBI DECODER**" is the bonafide work of **Sankari.K.N.** [Reg. No. **0920106012**] who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

  
Project Guide

(D. Mohana Geetha M.E.,)

  
Head of the Department

(Dr. Rajeswari Mariappan Ph.D.,)

The candidate with university **Register No. 0920106012** is examined by us in the project viva-voce examination held on ..... 21.04.2011 .....

  
Internal Examiner

  
External Examiner

## ACKNOWLEDGEMENT

A project of this nature needs co-operation and support from many for successful completion. In this regards, we are fortunate to express our heartfelt thanks to **Padmabushan Arutselvar Dr.N.Mahalingam B.Sc.,F.I.E.,Chairman and Co-Chairman Dr.B.K.Krishnaraj Vanavarayar B.com.,B.L.**, for providing necessary facilities throughtout the course.

I would like to express my thanks and appreciation to many people who have contributed to the successful completion of this project. First I thank **Dr.J.Shanmugam Ph.D.,Director**, for providing us an opportunity to carry out this project.

I wish to acknowledge my sincere gratitude and thanks to our beloved Principal **Dr.S.Ramachandran Ph.D.**, for having given me the adequate support and opportunity for completing this project work successfully.

I express my sincere thanks to **Dr.Rajeswari Mariappan Ph.D.**, the ever active. Head of the Department of Electronics and Communication Engineering, who rendering me all the time by helps throughout this project.

In particular, I wish to thank and everlasting gratitude to the project coordinator **Ms.R.Latha M.E.**, Asso.Professor. Department of Electronics and Communication Engineering for her expert counseling and guidance to make this project to a great deal of success. With her careful supervision and ensured me in the attaining perfection of work.

I extend my heartfelt thanks to my internal guide **Ms.D.Mohana Geetha M.E.**, Asso.Professor. Department of Electronics and Communication Engineering for her ideas and suggestion, which have been very helpful for the completion of this project work.

Last, but not the least, I would like to express my gratitude to my family members, friends and to all my staff members of Electronics and Communication Engineering department for their encouragement and support throughout the course of this project.

## ABSTRACT

Convolutional codes are widely used in many digital communication systems for increasing the reliability of transmission due to the fact that they have regular trellis structure. It is designed to detect and correct some of the occurred errors with reliable data transfer. Viterbi Algorithm is a decoding algorithm for convolutional codes. For viterbi decoder, high throughput is achieved by applying look - ahead technique in Add - Compare - Select (ACS) unit, which is the system speed bottle neck. As the look - ahead level increases, the Branch Metric Precomputation (BMP) dominates the complexity and delay of the architecture. So, the optimal branch metric computation scheme with minimal complexity and latency is implemented.

Here, the layered architecture other than the look - ahead is proposed for reducing the latency at low complexity. The design is implemented in Very high speed integrated circuits Hardware Description Language (VHDL), logic simulation is done using the Modelsim XE III 6.2g and synthesis is done using the Xilinx ISE 9.2i. The results show that the complexity and latency is minimized. It was also found that the look - ahead level  $M$  is also relaxed for all the values.

# TABLE OF CONTENT

<b>CHAPTER NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
	<b>ABSTRACT</b>	iv
	<b>LIST OF FIGURES</b>	vii
	<b>LIST OF TABLES</b>	ix
	<b>LIST OF ABBREVIATIONS</b>	x
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 Project Goal	2
	1.2 Software's Used	3
	1.3 Organisation of the Chapter	3
<b>2</b>	<b>CHANNEL CODING</b>	4
	2.1 Need to use	4
	2.2 Channel code types	4
	2.3 Linear Block Codes	4
	2.3.1 Cyclic Codes	5
	2.3.2 Repetition Code	5
	2.3.3 Parity Codes	6
	2.3.4 Polynomial Codes	7
	2.3.5 Reed - Solomon Codes	7
	2.3.6 Algebraic Geometric Code	7
	2.3.7 Reed Muller Code	8
	2.3.8 Perfect Code	8
	2.4 Basic notions of Convolutional codes	8

<b>3</b>	<b>CONVOLUTIONAL ENCODER</b>	<b>9</b>
	3.1 Overview and Introduction	9
	3.2 Convolutional encoding	10
	3.3 Encoder structure	10
	3.4 Encoder representation	11
	3.4.1 Generator representation	12
	3.4.2 Tree-diagram representation	12
	3.4.3 State-diagram representation	14
	3.4.4 Trellis diagram representation	15
<b>4</b>	<b>VITERBI DECODER</b>	<b>19</b>
	4.1 Overview and Introduction	19
	4.2 Processing Technique	19
	4.3 Block Diagram	20
	3.3.1 Previous work	21
	3.3.2 Proposed work	21
	4.4 Branch Metric Unit (BMU)	22
	4.5 Add - Compare - Select Unit (ACS)	23
	4.6 Branch Metric Precomputation	24
	4.6.1 Conventional Look - Ahead Architecture	24
	4.6.2 Layered Look - Ahead Architecture	26
	4.7 Survivor Path Memory Unit	29
<b>5</b>	<b>RESULTS AND DISCUSSION</b>	<b>31</b>
	5.1 Simulation result	32
	5.2 Synthesis report	42
<b>6</b>	<b>CONCLUSION AND FUTURE SCOPE</b>	<b>48</b>
	<b>REFERENCES</b>	<b>49</b>

## LIST OF FIGURES

FIGURE NO	CAPTION	PAGE NO
3.1	Convolutional encoder with $k=1, n=2, r=1/2, m=2, K=3$	11
3.2	Tree diagram representation of encoder	13
3.3	State diagram representation of encoder	14
3.4	The state transitions (path) for input sequence {1011}	15
3.5	Trellis diagram representation of encoder for 4 input bit intervals	16
3.6	Trellis path for the state transitions	16
3.7	Convolutional encoder with $k=2, n=3, r=2/3, m=3, K=4$	17
3.8	Convolutional encoder with $k=2, n=5, r=2/5, m=8, K=9$	18
4.1	The General Processing Technique	20
4.2	Block Diagram	20
4.3	Branch Metric Block	22
4.4	Add – Compare – Select Unit	23
4.5	A conventional Viterbi decoder architecture	25
4.6	A Layered Viterbi decoder architecture	27
4.7	Survivor Path Memory Unit	29
5.1	Simulated result of rate 1/2 convolutional encoder	32
5.2	Simulated result of rate 2/3 convolutional encoder	33
5.3	Simulated result of rate 2/5 convolutional encoder	34

5.4	Simulated result of Branch Metric Block	35
5.5	Simulated result of Add – Compare – Select Block	36
5.6	Simulated result of conventional precomputation	37
5.7	Simulated result of layered precomputation	38
5.8	Simulated result of survivor path memory unit	39
5.9	Simulated result of viterbi decoder in conventional method	40
5.10	Simulated result of viterbi decoder in layered method	41



## LIST OF TABLE

<b>TABLE NO</b>	<b>CAPTION</b>	<b>PAGE NO</b>
5.1	Comparison of conventional and layered method	47

## LIST OF ABBREVIATIONS

ASIC	-----	Application Specific Integrated Circuits
CDMA	-----	Code Division Multiple Access
EDA	-----	Electronic Design Automation
FEC	-----	Forward Error Correction
FPGA	-----	Field-Programmable Gate Arrays
IS	-----	Interim Standard
ISE	-----	Integrated System Environment
LTI	-----	Linear Time Invariant
PAR	-----	Place And Route
SNR	-----	Signal to Noise Ratio
SOC	-----	System-On-Chip
VHDL	-----	Very high speed integrated circuits Hardware Description Language
VLSI	-----	Very Large Scale Integration

# CHAPTER 1

## INTRODUCTION

Convolutional coding has been used in communication systems including wireless communications and deep space communications. It offers an alternative to block codes for transmission over a noisy channel. An advantage of Convolutional coding is that it can be applied to a continuous data stream as well as to blocks of data. IS-95, a wireless digital cellular standard for CDMA (code division multiple access), employs Convolutional coding. A third generation wireless cellular standard, under preparation, plans to adopt turbo coding, which stems from Convolutional coding.

The Viterbi decoding algorithm, proposed in 1967 by Viterbi, a decoding process for trellis structure in less noisy channel can be applied to a host problem encountered in the design of communication system. The Viterbi decoding algorithm provides both a maximum – likely hood and maximum posterior algorithm. A maximum posterior algorithm identifies a code word that maximize the conditional probability of the decoding code word against the receiving code word, in contrast maximum likely hood algorithm identifies a code word that maximize the conditional probability of the received code word against the decoded code word.

Traditionally, performance and silicon area are the most important concern in VLSI design. Recently area optimization is become an important concern, especially in transmitting application, such as cellular phones, laptop computers. In this, Viterbi decoder at the gate level in the standard cell design environment which reduces the complexity and latency in FPGA is the proposed system.

In the standard cell design environment, the behavior of a design is described in a high level hardware description language, such as VHDL. The behavioural design is synthesized to generate a gate level design. The advantage of the standard cell based design over full custom design is faster turnaround time for the easy design in verification and more accurate modeling of the circuit.

The Viterbi algorithm is a maximum likely hood decoding scheme that is used to recover encoded information corrupted during transmission over noisy channel. Trellis code modulation

is considered as means of transmitting coded information at high data rates. The number of trellis structure of the branch metric are used to find the shortest path for the area optimization. The trellis diagram can be decomposed into butterfly state interconnect structure. To reduce bandwidth expansion, higher rate codes upto rate 1 are obtained by puncturing the basic codes, which preserved the structure of trellis. The trellis consider the subset selector which consists of radix 4 butterflies, where four branches are leaving and entering each state. However, flexibility is also required in other processing blocks, namely branch metric and surviving path.

Another application of codes, used in some mobile phone systems, is Code Division Multiple Access (CDMA). Each phone is assigned a code sequence that is approximately uncorrelated with the codes of other phones. When transmitting, the code word is used to modulate the data bits representing the voice message. At the receiver, a demodulation process is performed to recover the data. The properties of this class of codes allow many users to use the same radio channel at the same time. At the receiver, the signals of other users will appear to the demodulator only as a low level noise.

Another general class of codes are the Automate Repeat Request codes. In these codes the sender adds redundancy to each message for error checking, usually by adding check bits. If the check bits are not consistent with the rest of the message when it arrives, the receiver will ask the sender to retransmit the message. All but the simplest wide area network protocols use ARQ. There is an extensive field of research on this topic because of the problem of matching a rejected packet against a new packet.

## **1.1 PROJECT GOAL**

The goal of the project is to implement optimal branch metric computation scheme for decoding the convolutional codes by viterbi algorithm with minimal complexity and latency.

## **1.2 SOFTWARES USED**

- Modelsim XE III 6.2g
- Xilinx ISE 9.2i

## **1.3 ORGANISATION OF THE REPORT**

- Chapter 2 explains the Channel coding types.
- Chapter 3 deals with the Convolutional encoder structure.
- Chapter 4 deals with the viterbi decoder and functional units.
- Chapter 5 gives the simulation results and synthesis report for three types of encoder structure and Viterbi decoder.
- Chapter 6 deals with the conclusion.

## **CHAPTER 2**

### **CHANNEL CODING**

Channel coding is the process of adding redundant information to the data being transmitted. It improves the capacity of a channel. The simplest error correcting code is the repetition code. Instead of transmitting one binary symbol once, we transmit it for three times. If one of the transmission is erroneously received, we can detect that and still be able to say what was transmitted.

Channel codes that are used to detect errors are called error detection codes, while codes that can detect and correct errors are called error correction codes. The introduction of the redundant increases the data rate and bandwidth.

#### **2.1 NEED TO USE**

The channel is not ideal. The transmitted signals are often corrupted by channel noise and can use channel coding to correct these errors so as to achieve the reliable transmission of information over a large variety of channels. Propagation can introduce errors to the signal caused by path loss and path fading. So, channel coding is introduced to overcome these problems.

#### **2.2 CHANNEL CODE TYPES**

A channel coder operates on digital message data by encoding the source information into a code sequence for transmission through the channel. The term algebraic coding theory denotes the sub - field of coding theory where the properties of codes are expressed in algebraic terms. Algebraic coding theory is basically divided into two major types of codes.

- Linear block codes.
- Convolutional codes

#### **2.3 LINEAR BLOCK CODES**

Linear block codes have the property of linearity, i.e. the sum of any two code words is also a code word and they are applied to the source bits in blocks, hence the name is linear block

codes. There are block codes that are not linear, but it is difficult to prove that a code is a good one without this property. Linear block codes are summarized by their symbol alphabets and parameters  $(n, m, d_{\min})$  where.

- $n$  is the length of codeword in symbols.
- $m$  is the number of source symbols that will be used for encoding at once.
- $d_{\min}$  is the minimum hamming distance for the code.

There are many types of linear block codes, such as

- Cyclic codes.
- Repetition codes.
- Parity codes.
- Polynomial codes.
- Reed - Solomon codes.
- Algebraic geometric codes.
- Reed - Muller codes.
- Perfect codes.

### 2.3.1 CYCLIC CODES

Cyclic codes are linear block error correcting codes that have convenient algebraic structures for efficient error detection and correction. Let  $C$  be a linear code over a finite field of block length  $n$ .  $C$  is called a cyclic code obtained by a cyclic right shift of components against a code word. The ideal is generated by the unique element in  $C$  of minimum degree, the generator polynomial  $g$ . If the generator polynomial  $g$  has degree  $d$  then the rank of the code  $C$  is  $n - d$ .

### 2.3.2 REPETITION CODE

Repetition code is an  $(r, 1)$  coding scheme that repeats the bits across a channel to achieve error free communication. Repetition code is generally a very naïve method of encoding data across a channel, and it is not preferred for Additive White Gaussian Noise channels, due to its worse than the present error performance. Repetition codes generally offer a poor compromise

between data rate and bit error rate. Other forms of error correcting codes can provide superior performance in these areas. The chief attraction of the repetition code is the ease of implementation. There are two parts of the repetition code, as for any other code: The encoder and decoder.

### 2.3.3 PARITY CODES

A parity bit is a bit that is added to ensure that the number of bits with the value one in a set of bits is even or odd. Parity bits are used as the simplest form of error detecting code. There are two variant of parity bits: even parity bit and odd parity bit. When using the even parity, the parity bit is set to 1 if the ones in a given set of bits is odd, making the entire set of bits even. When using the odd parity, the parity bit is set to 1 if the number of ones in a given set of bits is even, making the entire set of bits odd. Even parity is a special case of a Cyclic Redundancy Check, where the 1 – bit is generated by the polynomial  $x + 1$ . If the parity bit is present but not used, it may be referred to as mark parity or space parity. In Telecommunications and Computing, parity refers to the evenness or oddness of the number of bits with value one within a given set of bits and thus determined by the value of all the bits.

If an odd number of bits are transmitted incorrectly, the parity bit will be incorrect and thus indicates that an error occurred in transmission. The parity bit is only suitable for detecting errors. It cannot correct any errors, as there is no way to determine which particular bit is corrupted. The data must be discarded entirely, re-transmitted from scratch. However, parity has the advantage that it uses only a single bit and requires only a number of XOR gates to generate. Parity bit checking is used occasionally for transmitting ASCII characters, which have 7 bits, leaving the 8<sup>th</sup> bit as a parity bit.

Because of its simplicity, parity is used in many hardware applications where an operation can be repeated in case of difficulty, or where simply detecting the error is helpful. For example, PCI buses use parity to detect transmission errors and many microprocessor instruction caches include parity protection. Because, 1 – cache data is just a copy of main memory, it can be thrown away and re-fetched if it is found corrupted. In serial data transmission, a common format is 7 data bit, an even parity bit and one or two stop bits.



### 2.3.4. POLYNOMIAL CODES

A polynomial code is a type of linear code whose set of valid code words consists of those polynomials that are divisible by a given fixed polynomial. Fix a finite field, whose elements are called symbols. The polynomial code generated by  $g(x)$  is the code whose code words are precisely the polynomials of degree less than  $n$  that are divisible by  $g(x)$ . An erroneous message can be detected in a straight-forward way through polynomial division by the generator polynomial resulting in a non-zero remainder. Assuming that the code word is free of errors, a systematic code can be decoded by stripping away the  $m$  checksum digits. If there are errors, then error correction should be performed before decoding. Efficient decoding algorithms exist for specific polynomial codes.

### 2.3.5. REED – SOLOMON ERROR CODES

Reed-Solomon (RS) codes are non-binary cyclic error-correcting codes invented by Irving S. Reed and Gustave Solomon. They described a systematic way of building codes that could detect and correct multiple random symbol errors. By adding  $t$  check symbols to the data, an RS code can detect any combination of up to  $t$  erroneous symbols, and correct up to  $t/2$  symbols. Furthermore, RS codes are suitable as multiple-burst bit-error correcting codes, since a sequence of  $b+1$  consecutive bit errors can affect at most two symbols of size  $b$ . The original concept of Reed-Solomon coding describes encoding of  $k$  message symbols by viewing them as coefficients of a polynomial  $p(x)$  of maximum degree  $k-1$ .

### 2.3.6 ALGEBRAIC GEOMETRIC CODE

In mathematics, an algebraic geometric code (AG code), otherwise known as a Goppa code, is a general type of linear code constructed by using an algebraic curve  $X$  over a finite field  $F$ . Such codes were introduced by Valerii Denisovich Goppa. In particular cases, they can have interesting external properties. They should not be confused with binary Goppa that are used for instance in the McEliece cryptosystem. Traditionally, an AG code can be constructed from a non-singular projective curve  $X$  over a finite field  $F$ .

### 2.3.7 REED MULLER CODE

Reed-muller codes are a family of linear error-correcting codes used in communications. They are named after their discoverers, Irving S. Reed and D. E. Muller. Muller discovered the codes and Reed proposed the majority logic decoding for the first time. A first order Reed-Muller code is equivalent to a Hadamard code. RM codes are related to the binary functions. Reed-Muller codes are listed as  $RM(d, r)$ , where  $d$  is the order of the code and  $r$  is the parameter related to the length of the code.

### 2.3.8 PERFECT CODE

In mathematics and computer science, in the field of coding theory, the Hamming bound is a limit on the parameters of an arbitrary block code: it is also known as the sphere packing bound or the volume bound from an interpretation in terms of packing balls in the Hamming metric into the space of all possible words. It gives an important limitation on the efficiency with which any error correcting code can utilize the space in which its code words are embedded. A code which attains the Hamming bound is said to be a perfect code. An original message and an encoded version are both composed in an alphabet of  $q$  letters. Each code word contains  $n$  letters. The original message (of length  $m$ ) is shorter than  $n$  letters. The message is converted into an  $n$ -letter code word by an encoding algorithm, transmitted over a noisy channel and finally decoded by the receiver. The decoding process interprets a garbled code word as the valid code word "nearest" the  $n$ -letter received string.

## 2.4 BASIC NOTIONS OF CONVOLUTIONAL CODES

- Convolutional code adds redundancy to a message sequence in order to form code sequence.
- Input: a sequence of  $k$ -tuples over some field.
- Output: a sequence of  $n$ -tuples ( $n > k$ ) over same field.
- Transition from input to output occurs over one clock period.
- Total memory ( $m$ ) of an encoder is number of consecutive  $k$ -tuples that it can store.
- Constraint length ( $K$ ) of a code is the longest number of non-zero output blocks produced by a single non-zero input block.

## CHAPTER 3

### CONVOLUTIONAL ENCODER

#### 3.1 OVERVIEW AND INTRODUCTION

Convolutional codes are used in voice band modems (V.32, V.17, and V.34), in GSM mobile phones, as well as in satellite and military communication devices. Here, the idea is to make every codeword symbol be the weighted sum of the various input message symbols. This is like convolution used in LTI systems to find the output of a system, when the input and impulse response are known. Fundamentally, convolutional codes do not offer more protection against noise than an equivalent block code. So, generally the output of the system convolutional encoder is found, which is the convolution of the input bit, against the states of the convolutional encoder and registers.

In many cases, they generally offer greater simplicity of implementation over a block code of equal power. The encoder is usually a simple circuit which has state memory and some feedback logic, normally XOR gates. The Viterbi algorithm is the optimum algorithm used to decode convolutional codes. There are simplifications to reduce the computational load. They rely on searching only the most likely paths. Although not optimum, they have generally found to give good results in the lower noise environments.

In Telecommunications, a convolutional code is a type of error-correcting code in which

- Each  $m$  – bit information symbol to be encoded is transformed into an  $n$  – bit symbol, where  $m/n$  is the code rate.
- The transformation is a function of the last  $k$  information symbols, where  $k$  is the constraint length of the code.

Convolutional codes are used extensively in numerous application in order to achieve reliable data transfer, including digital video, radio, mobile communication and satellite communication. These codes are often implemented in concatenation with a hard – decision code, particularly Reed Solomon. Prior to Turbo codes, such constructions were the most efficient, coming closest to the limit.

## 3.2 CONVOLUTIONAL ENCODING

To convolutionally encode data, start with  $k$  memory registers, each holding 1 input bit. Unless otherwise specified, all memory registers start with a value of 0. The encoder has  $n$  modulo-2 adders (A modulo 2 adder can be implemented with a single Boolean XOR gate, where the logic is:  $0+0 = 0$ ,  $0+1 = 1$ ,  $1+0 = 1$ ,  $1+1 = 0$ ), and  $n$  generator polynomials one for each adder. An input bit  $m_1$  is fed into the leftmost register.

Using the generator polynomials and the existing values in the remaining registers, the encoder outputs  $n$  bits. Now bit shift all register values to the right ( $m_1$  moves to  $m_0$ ,  $m_0$  moves to  $m_{-1}$ ) and wait for the next input bit. If there are no remaining input bits, the encoder continues output until all registers have returned to the zero state. The encoder can be represented in several different but equivalent ways.

## 3.3 ENCODER STRUCTURE

The information bits are input into shift registers and the output encoded bits are obtained by modulo-2 addition of the input information bits and the contents of the shift registers. The connections to the modulo-2 adders were developed heuristically with no algebraic or combinatorial foundation. The code rate  $r$  for a convolutional code is defined as

$$r = k / n$$

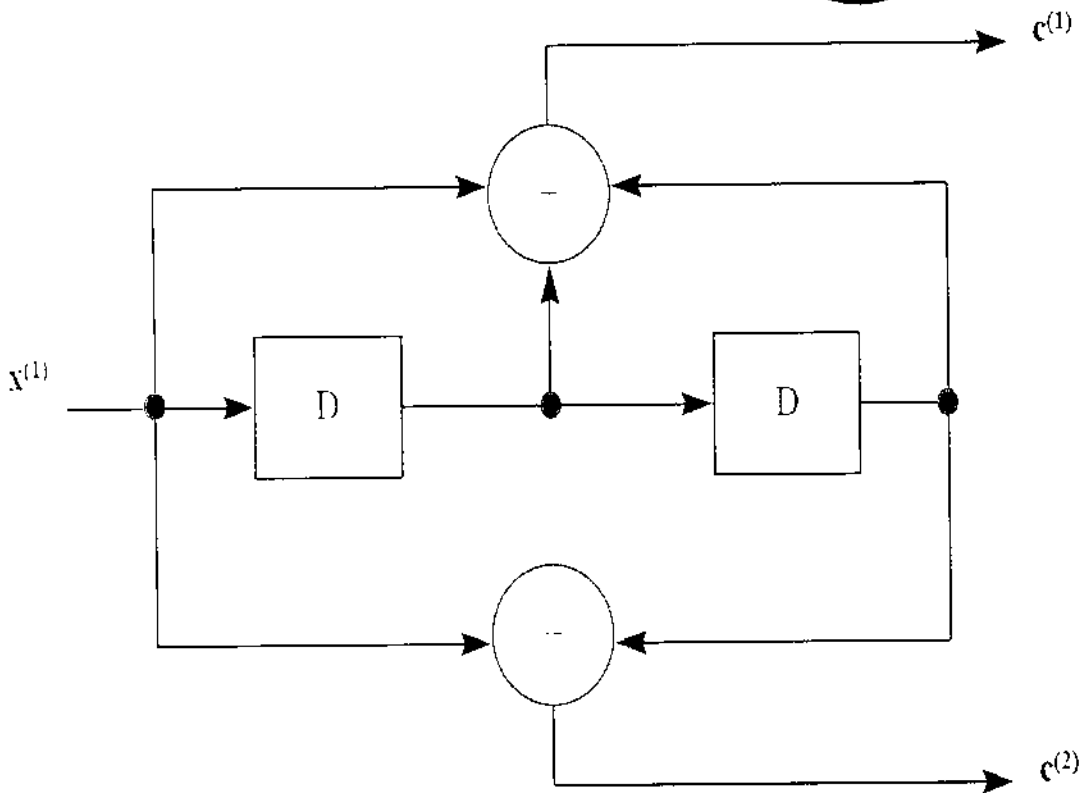
$k$  - Number of inputs of the encoder

$n$  - Number of outputs of the encoder

The constraint length  $K$  for a convolutional code is defined as

$$K = m + 1$$

$m$  - Maximum number of stages (memory size) in any shift register.



**Figure 3.1: Convolutional encoder with  $k=1$ ,  $n=2$ ,  $r=1/2$ ,  $m=2$  and  $K=3$ .**

The encoder in figure 3.1 produces two bits of encoded information for each bit of information, so it called a rate 1/2 encoder. A convolutional encoder is generally characterized in  $(n,k,m)$  format. The encoder shown in the figure is a  $(2, 1, 2)$  encoder with rate 1/2.

### 3.4 ENCODER REPRESENTATIONS

The encoder can be represented in several different but equivalent ways. They are

- Generator Representation
- Tree Diagram Representation
- State Diagram Representation
- Trellis Diagram Representation

### 3.4.1 GENERATOR REPRESENTATION

Generator representation shows the hardware connection of the shift register taps to the modulo-2 adders. A generator vector represents the position of the taps for an output. A "1" represents a connection and a "0" represents no connection. For example, the two generator vectors for the encoder in figure 3.1 are  $g_1 = [111]$  and  $g_2 = [101]$  where the subscripts 1 and 2 denote the corresponding output terminals.

### 3.4.2 TREE DIAGRAM REPRESENTATION

The tree diagram representation shows all possible information and encoded sequences for the convolutional encoder. Figure 3.2 shows the tree diagram for the encoder for four input bit intervals. In the tree diagram, a solid line represents input information bit 0 and a dashed line represents input information bit 1. The corresponding output encoded bits are shown on the branches of the tree. An input information sequence defines a specific path through the tree diagram from left to right.

For example, the input information sequence  $x = \{1011\}$  produces the output encoded sequence  $c = \{11, 10, 00, 01\}$ . Each input information bit corresponds to branching either upward (for input information bit 0) or downward (for input information bit 1) at a tree node. To generate the output, there is no theoretical basis for the optimal location of the shift register to be connected to XOR gates. It is based on empirical approach. The location of the stages is determined by the interconnection functions, the location of stages as well as the number of memory elements determine the minimum hamming distance. Minimum hamming distance determines the maximal number of correct bits interconnected for different rates.

Constraint length represents the number of bits in encoder memory that affect the generation of  $n$  output bits. The selection of output bits is called as generator polynomial. In encoder section, all the inputs are EXOR to get the output in that section. The polynomial gives a code of error protection quality. After all the encoder symbols of the information bits are transmitted, the encoder is usually forced back into the initial state by applying a fixed input sequence called reset sequence. The fixed input sequence reduces the possible transmission.

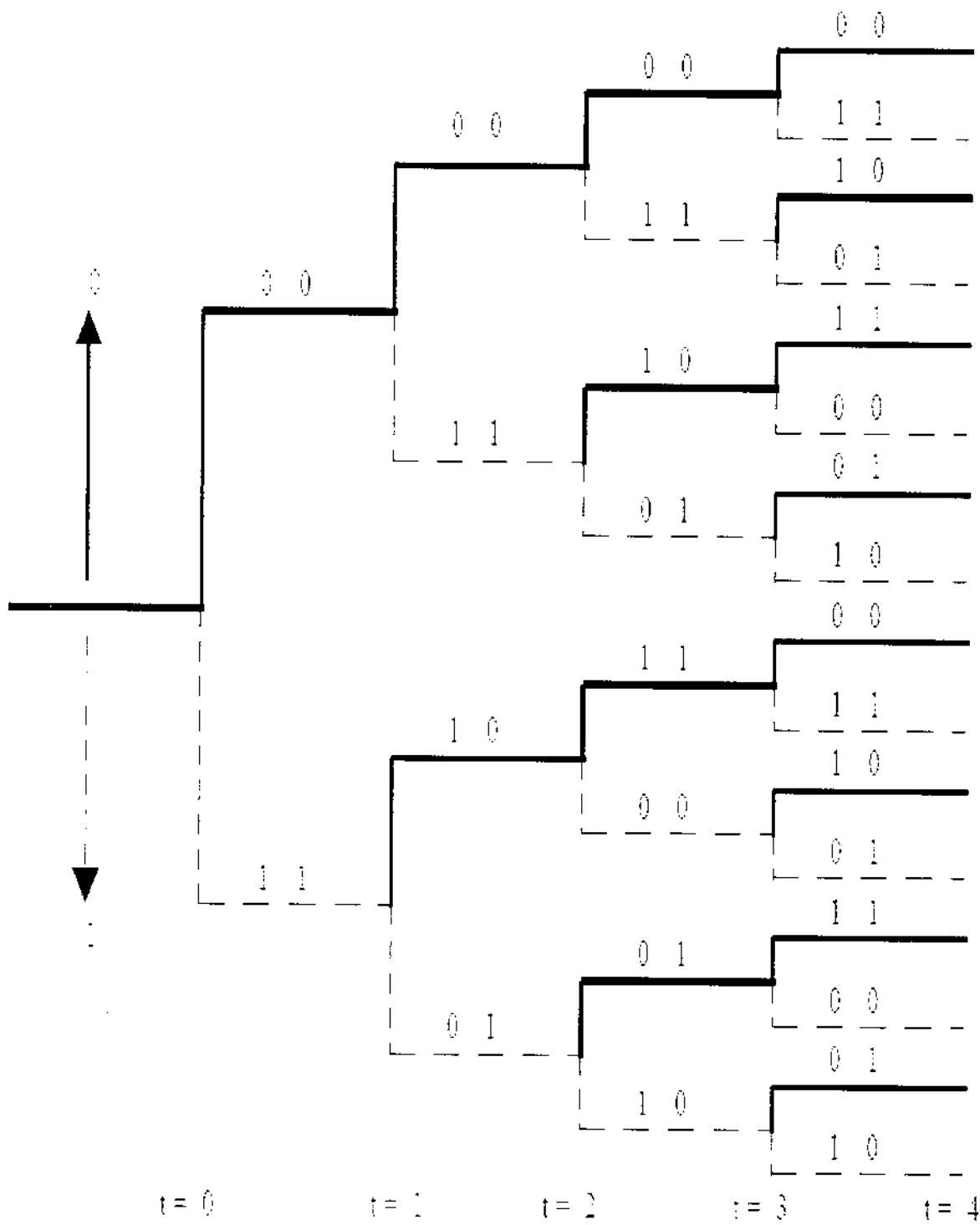
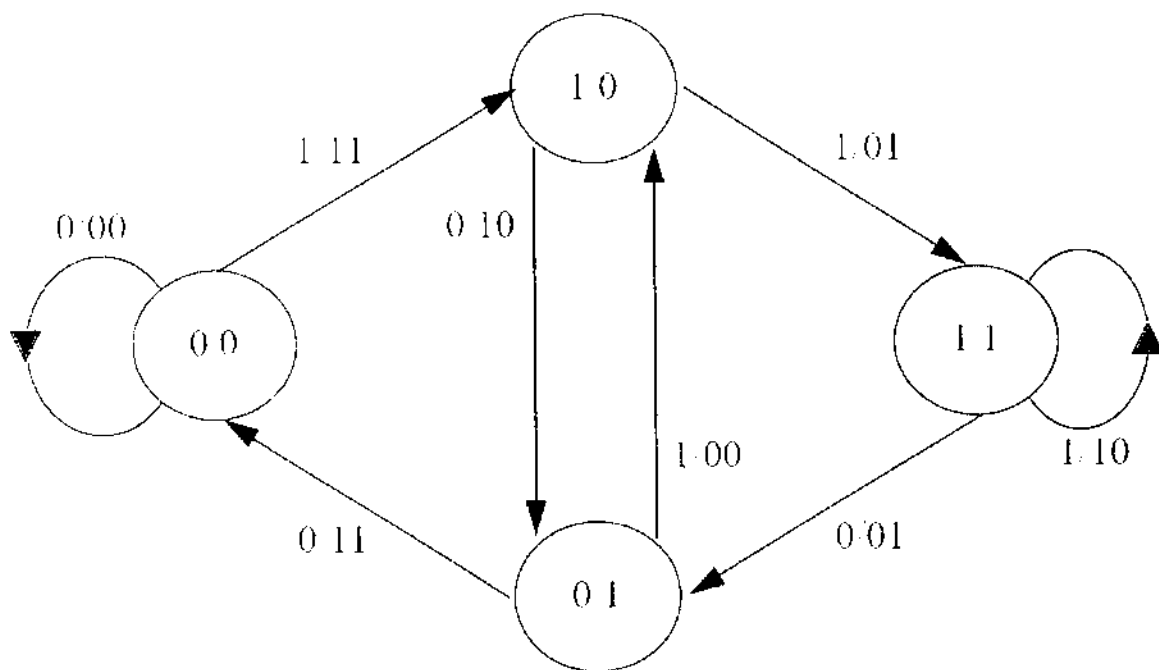


Figure 3.2: Tree diagram representation of the encoder

### 3.4.3 STATE DIAGRAM REPRESENTATION

The state diagram shows the state information of a convolutional encoder. The state information of a convolutional encoder is stored in the shift registers. Figure 3.3 shows the state diagram of the encoder. In the state diagram, the state information of the encoder is shown in the circles. Each new input information bit causes a transition from one state to another.

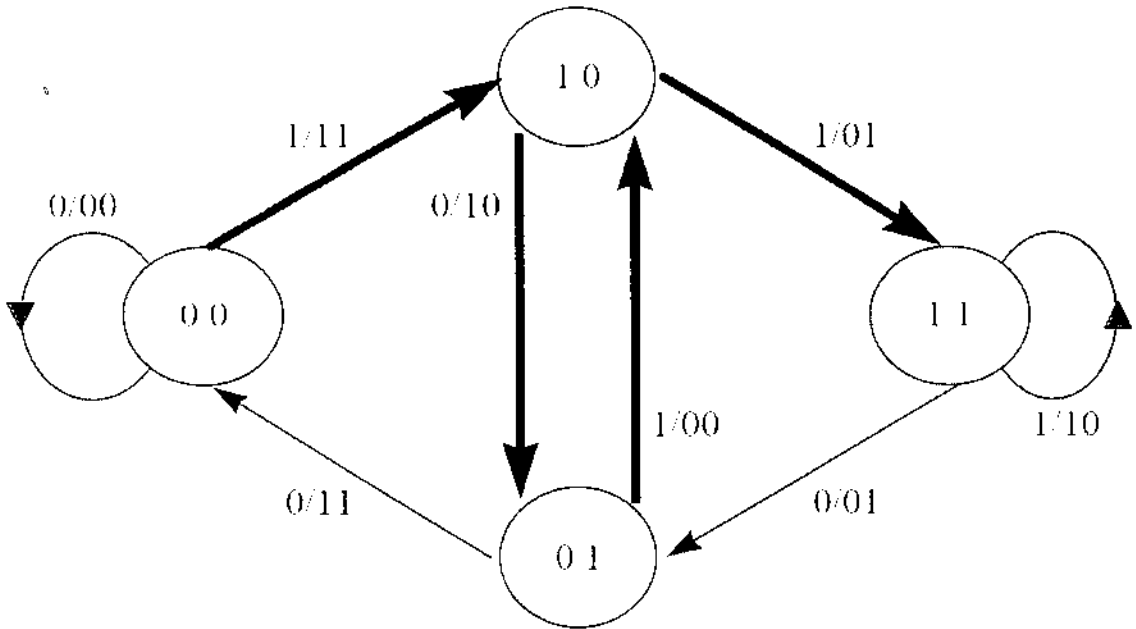


**Figure 3.3: State diagram representation of the encoder**

The path information between the states, denoted as  $x/c$ , represents input information bit  $x$  and output encoded bits  $c$ . It is need to begin convolutional encoding from the all zero state.

For example, the input information sequence  $x = \{1011\}$  (begin from the all zero state) leads to the state transition sequence  $s = \{10, 01, 10, 11\}$  and produces the output encoded sequence  $c = \{11, 10, 00, 01\}$ . Figure 3.4 shows the path taken through the state diagram for the encoder.





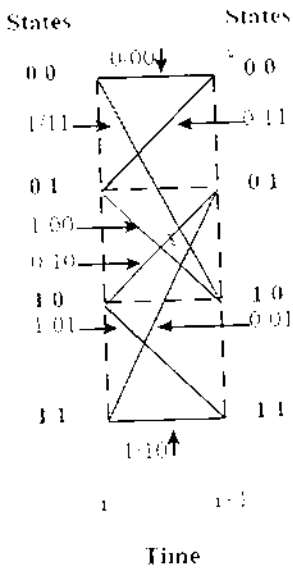
**Figure 3.4: The state transitions (path) for input information sequence {1011}.**

### 3.4.4 TRELLIS DIAGRAM REPRESENTATION

The trellis diagram is basically a redrawing of the state diagram. It shows all possible state transitions at each time step. Frequently, a legend accompanies the trellis diagram to show the state transitions and the corresponding input and output bit mappings ( $x/c$ ). Figure 3.5 shows the trellis diagram for the encoder. Trellis diagram is an extension of the state diagram that explicitly shows the passage of time. From an initial state, the trellis records the possible transition to the next states.

Hence, the trellis grows up to the maximum number of states or nodes, which is decided by the number of memory elements in the encoder. Consider a decoder that receives the transmitted signal 11 01 01 00 10 11 going from  $t = 0$  to  $t = 6$ . Assume the trellis was reset to state  $S_0$  ( 0 0 ) at the start. One goes through the trellis as before, only for decoding the numbers as output/input. So, the first input 11 gives a decoded output as 1 and takes the machine to state  $S_1$ . At  $t = 1$ , in state  $S_1$  the next input 01 causes a 1 output and a change to the state  $S_3$ .

**LEGEND**



**TRELLIS DIAGRAM**

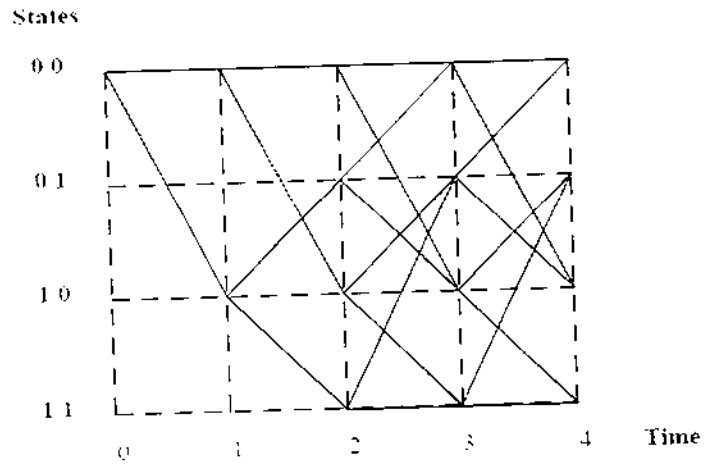


Figure 3.5: Trellis diagram representation of the encoder for four input bit Intervals.

**TRELLIS DIAGRAM**

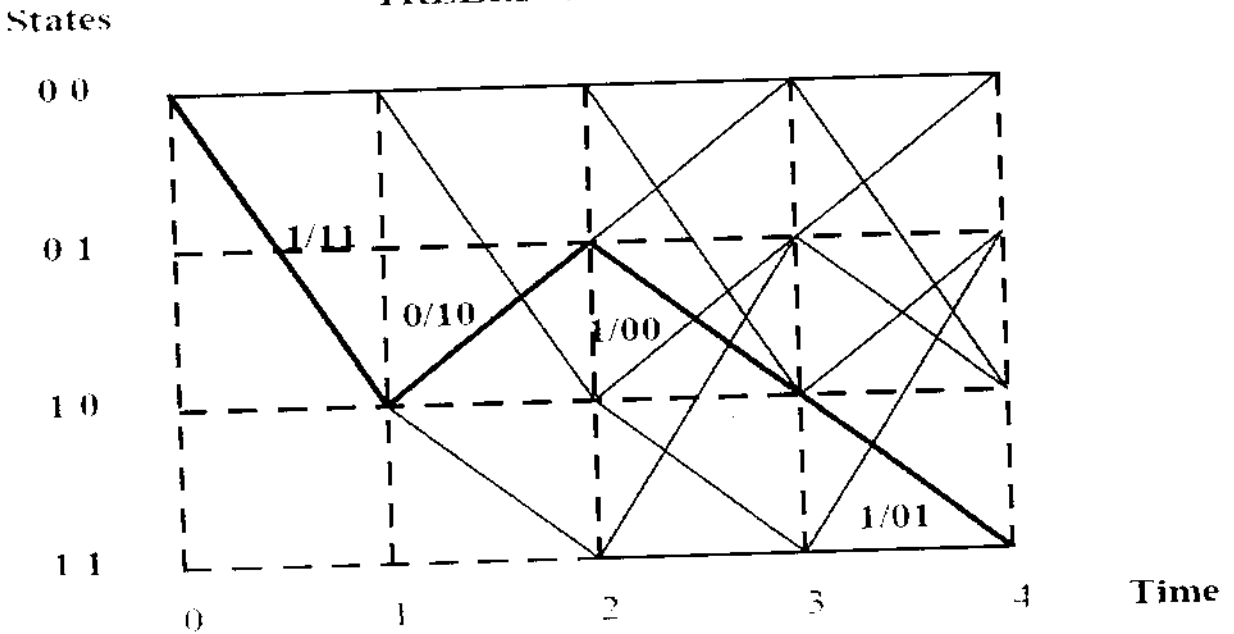
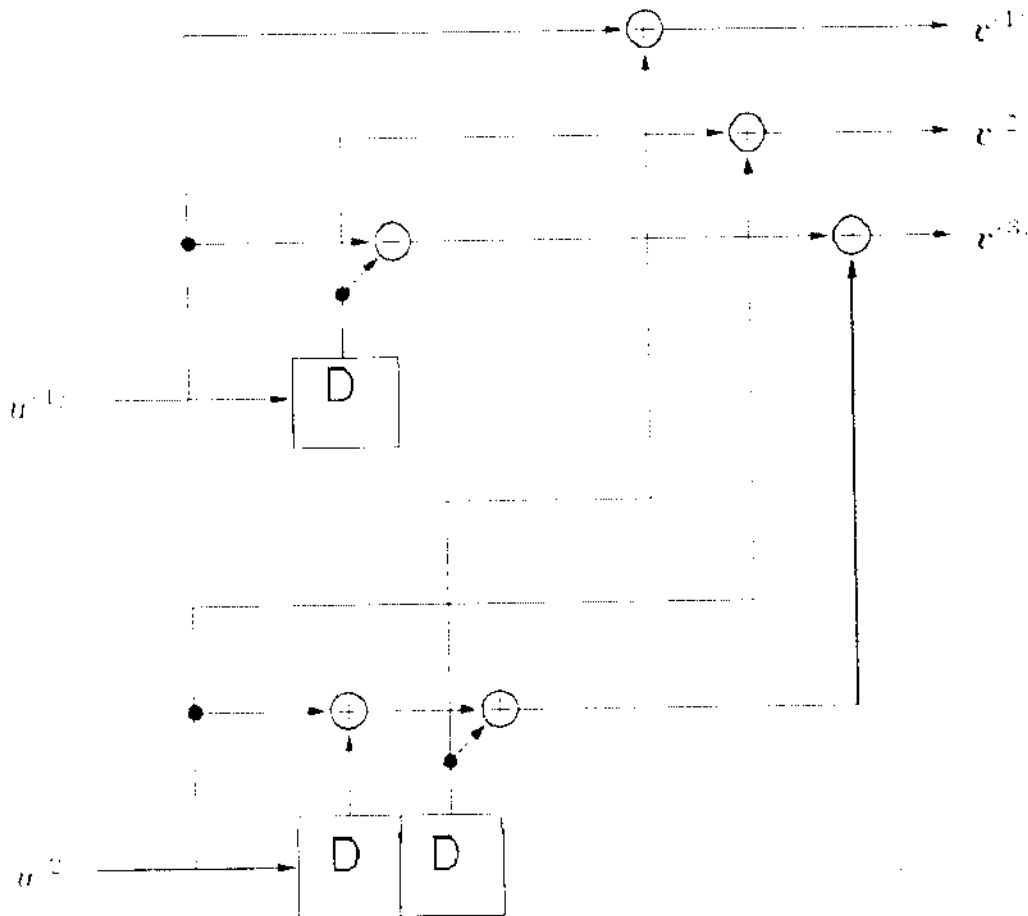


Figure 3.6: Trellis path for the state transitions.

## A RATE – 2 / 3 CONVOLUTIONAL ENCODER

The encoder in figure 3.7 produces three bits of encoded information for two bit of information, so it called a rate 2/3 encoder. The encoder shown in the figure is a ( 3, 2, 3 ) encoder with rate 2/3.



**Figure 3.7: Convolutional encoder with  $k=2$ ,  $n=3$ ,  $r=2/3$ ,  $m=3$  and  $K=4$ .**

The structure of this 2/3 convolutional encoder is implemented using VHDL code and the simulated results are shown.

## A RATE – 2 / 5 CONVOLUTIONAL ENCODER

The encoder in figure 3.8 produces five bits of encoded information for two bit of information, so it called a rate 2/5 encoder. The encoder shown in the figure is a ( 5, 2, 8 ) encoder with rate 2/5.

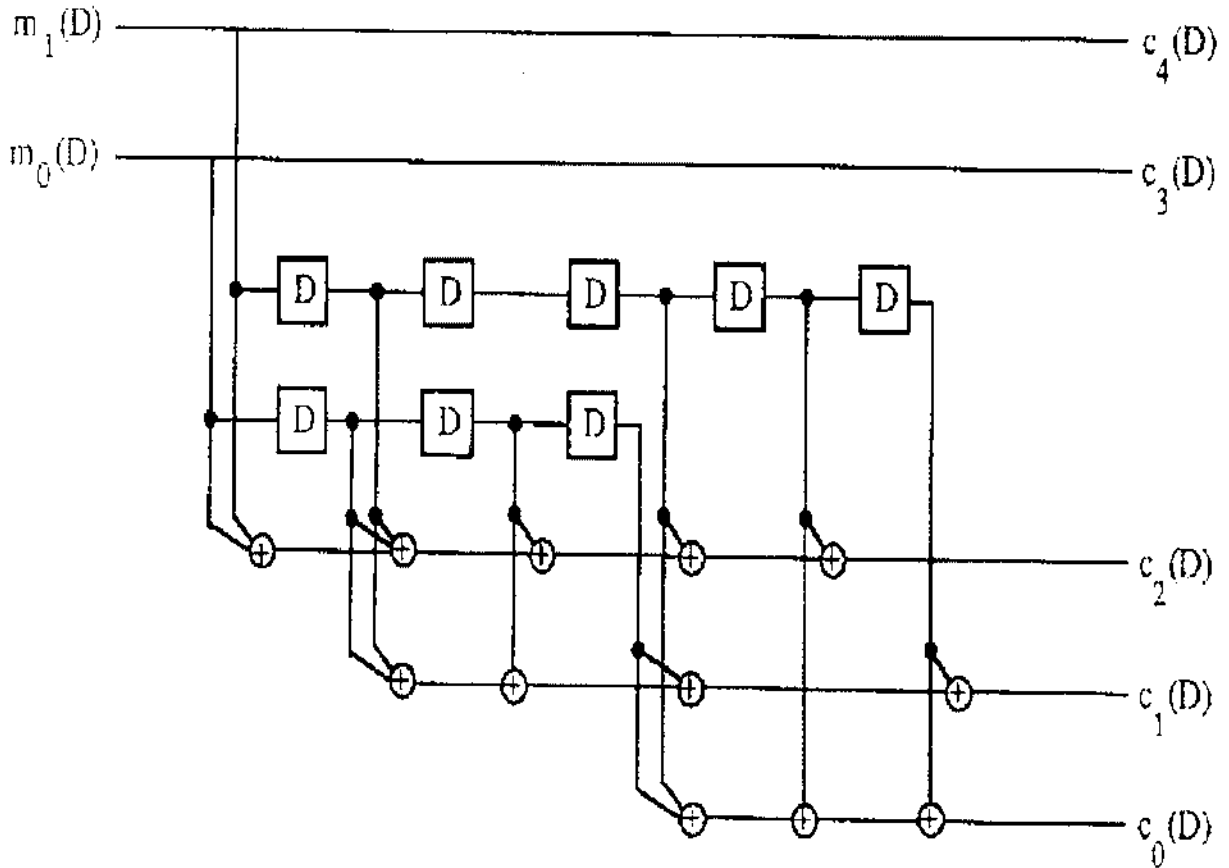


Figure 3.8: Convolutional encoder with  $k=2$ ,  $n=5$ ,  $r=2/5$ ,  $m=8$  and  $K=9$ .

The structure of this 2/5 convolutional encoder is implemented using VHDL code and the simulation results are shown.

## CHAPTER 4

### VITERBI DECODER

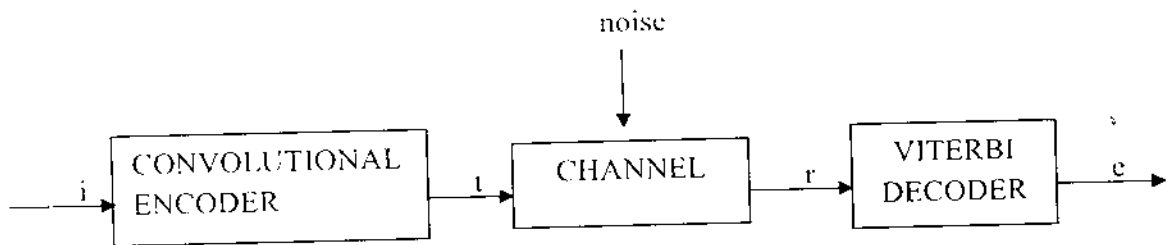
#### 4.1 OVERVIEW AND INTRODUCTION

Viterbi decoding is the best known implementation of the maximum likelihood of convolutional codes. The principle used to reduce the choices is that they occur infrequently. So that the probability of error is small and the probability of two errors in a row is much smaller than a single error, that is because the errors are distributed randomly. In this project, convolutional codes and the Viterbi decoder are used on the transmitter side and receiver side to effectively reduce the bit error. A Viterbi decoder uses the Viterbi algorithm for decoding a bit stream that has been encoded based on convolutional code. It is often used for decoding convolutional codes with constraint length  $k \leq 10$ , but values up to  $k=15$  are used in practice.

The Viterbi decoding algorithm is based on maximum likelihood decoding. The decoding is carried out with the help of a trellis diagram. A typical trellis diagram is shown in which each state can be reached from two paths and the longer of the two is discarded for an optimum path search. This is done with the help of an accumulated path metric. The computation is performed for each of the  $2^{K-1}$  nodes at each time  $t_i$ , where  $K$  is the constraint length of the decoder. As moving deeper into the trellis, only a single path survives. This path gives us the original information that was encoded.

#### 4.2 PROCESSING TECHNIQUE

In the decoder section, the trellis structure is used. It is more efficient than other decoders. For each input bit there will be two bits and those bits are compared with the states from the branch metric and path metrics are determined. In each node there is a branch metric. After certain nodes two paths will be occurred in the trellis structure in forwarding and in backwarding process. The data is secured, where in other decoder the complexity is more.

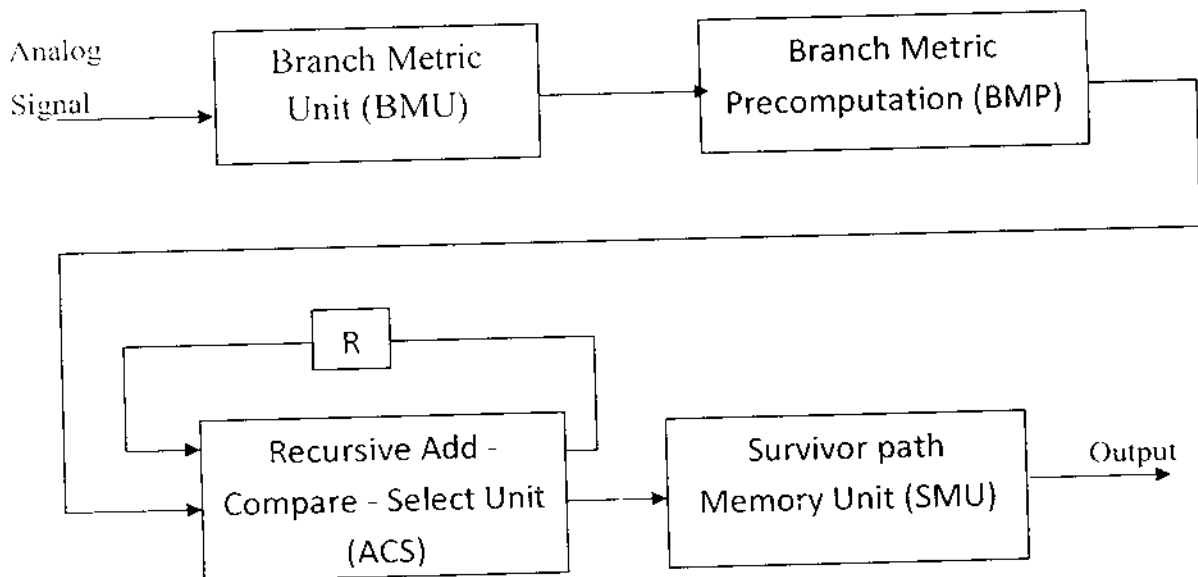


**Figure 4.1: The General Processing Technique**

If the received signal is converted into two levels, either zero or one, it is called hard decision. If the input signal is quantized and processed for more than two levels, it is called soft decision..

### 4.3 BLOCK DIAGRAM

Viterbi decoders comprise three major functional units: branch metric unit (BMU), add-compare select (ACS) unit, and survivor path memory unit (SMU). BMU generates metrics associated with the binary trellis based on the received signal, which are then fed into ACS for updating path metrics. SMU manages the survival paths and outputs the decoded data.



**Figure 4.2: Block diagram**

BMU and SMU are purely forward logic units, whereas the ACS unit is a recursive structure that imposes an iteration bound on the system performance.

### 4.3.1 PREVIOUS WORK:

The ACS unit becomes the speed bottleneck. Look-ahead techniques have been widely applied in ACS in order to achieve the high throughput rate that must be met in high-speed applications. Consider a Viterbi decoder of constraint length  $K$  with  $M$ -step look ahead. Instead of using one-step binary-trellis branch metrics to update path metrics, multiple-step binary-trellis branch metrics are fed into the ACS unit for updating. Thus, the speed breakthrough comes with the expense of more computational complexity and longer delay in Viterbi decoders.

Conventionally, the BMP is carried out one binary-trellis step at a time with pipelined registers inserted between two consecutive steps. Before the trellis is saturated, only add operation is needed. Once the trellis is saturated, each add operation is followed by a compare operation, where parallel paths with less metrics are discarded. The advantage of the conventional approach is that there is no wasted computation because unnecessary parallel paths are discarded at each step.

In a low-latency branch precomputation architecture for high-throughput-rate Viterbi decoders is used. The limitation and drawbacks are also obvious. 1) The cost of hardware is extremely expensive, particularly for large constraint length  $K$ ; 2) the architecture is only applicable when the look-ahead level is a multiple of  $K$ ; and 3) in the meanwhile, it is not efficient in reducing the latency when the look-ahead level is not a power-of-two multiple of  $K$ .

### 4.3.2 PROPOSED WORK:

Look-ahead techniques give rise to branch metric precomputation (BMP) in the front end of ACS, which is responsible for combining a multiple-step binary trellis into a one-step complex trellis and computing the equivalent branch metrics. Combined branch metrics are then fed into ACS for path metric updating.  $M/K$  layer-1 processors ( $P1$ s) first compute  $K$ -step branch metrics with one compare operation at the end of processing. In the following layer, the authors combine the trellis in a logarithmic manner with  $N^2$   $N$ -to-1 add-compare operations in each

ayer-2 processor (P2). The goal of this structure is to minimize the complexity as well as the latency in the BMP where the look-ahead level  $M$  is also relaxed to be of any value.

#### 4.4 BRANCH METRIC UNIT (BMU)

In the Viterbi algorithm for decoding convolutional codes, the squared Euclidean distance is the optimum branch metric for decoding sequence that are transmitted in an AWGN channel. Multi operations on look up tables are required for the viterbi algorithm to compute the squared distance to obtain the branch metrics.

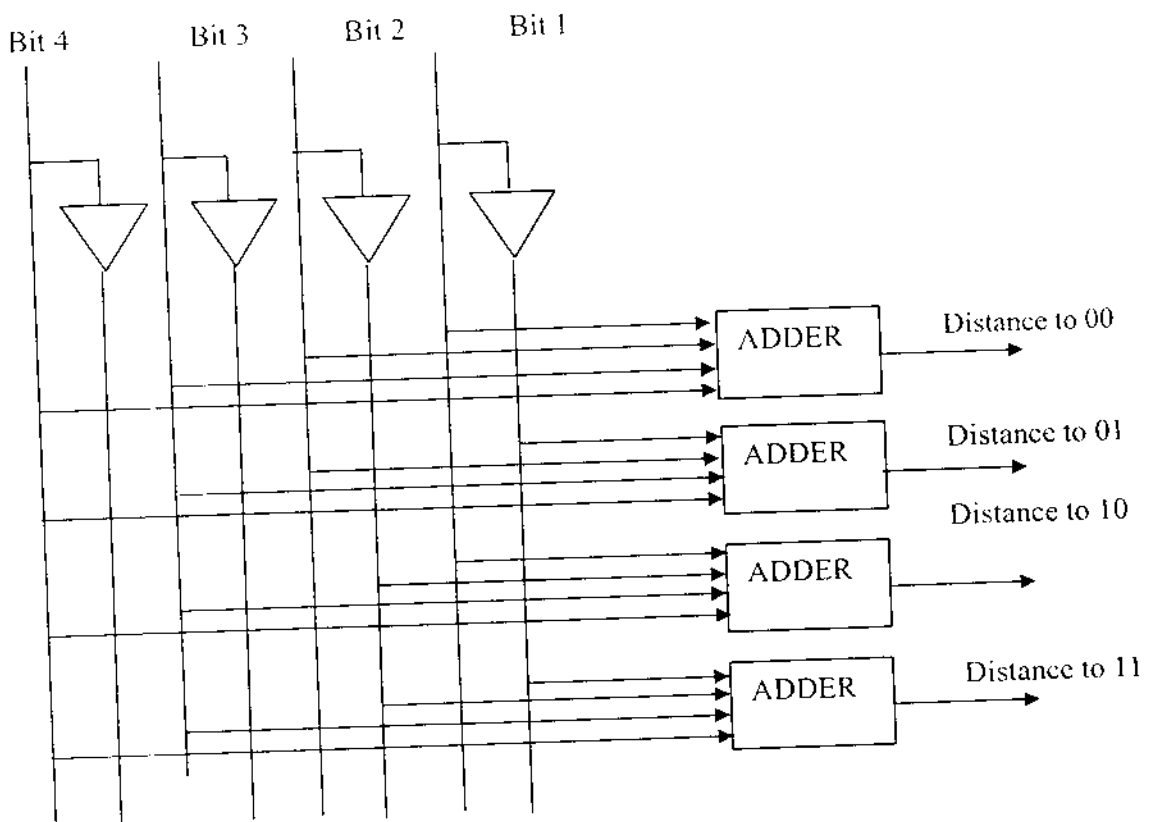


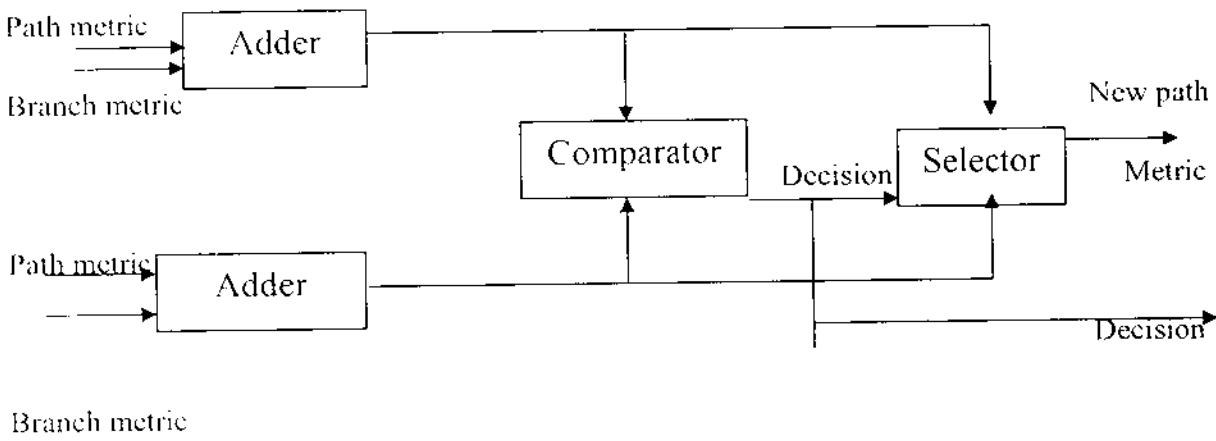
Figure: 4.3 Branch Metric Block



To calculate the hamming distance of code words with all possibilities in different rates, the Branch Metric Unit is implemented. Here, BMU is implemented for the rate  $\frac{1}{2}$  as shown in the figure 4.3. Branch metric is the distance from the received code word to all the possible branch words. For any  $1/n$  rate, the first  $n$  bits are connected to the input buffer and the rest will be set to zero. Also the first  $2^n$  outputs are used as the Branch Metric block outputs. No reconfiguration inside the block is needed to change the code rate.

**4.5 ADD – COMPARE – SELECT UNIT (ACS)**

Add – Compare – Select operation update the path metric (PM) of each trellis state and generate decision bits. Decision bits store and output the decoding sequence. An ACS module is shown in the figure 4.4. The two adders compute the partial path metric of each branch, the comparator compares the two partial metrics and the selector selects an appropriate branch. The new partial path metric updates the state metric and the survivor path – recording block records the survivor path.



**Figure 4.4: Add – Compare – Select Unit**

The number of necessary ACS module is equal to half the number of total states. Time sharing of some ACS modules is possible to save the hardware, but such sharing slows down the

operation and dissipates more power. In terms of speed, the performance of a Viterbi decoder is mainly determined by the number of ACS units and their computation time. The output of each ACS is assigned constantly to the corresponding path metrics and only the first  $2^{k-1}$  path metrics are used.

## 4.6 BRANCH METRIC PRECOMPUTATION

Each trellis transition in a step look-ahead trellis has  $2^{M-K+1}$  parallel paths. Parallel paths are the paths that have the same starting trellis state and same ending trellis state. The existence of parallel paths guarantees it is possible to use look-ahead ACS precomputation. However, if there are no parallel paths in the look-ahead trellis, only the look-ahead additions of the branch metrics are allowed. The conventional and the proposed step look - ahead Viterbi decoder architectures are shown in Fig. 4.5.

### 4.6.1 CONVENTIONAL LOOK - AHEAD ARCHITECTURE

Conventionally, the branch metrics precomputation is carried out step by step with pipelined registers inserted between two consecutive steps. Before the trellis is saturated, only add operation is needed. Whereas once trellis is saturated, each add operation will be followed by a compare operation, where parallel path with less metric is discarded. The advantage of the conventional approach is that there is no waste computation and unnecessary parallel paths are discarded at each step. A low latency branch precomputation architecture for high throughput rate Viterbi decoders is proposed.

Although it shows great improvement in reducing latency, the limitation and drawbacks are: 1) The cost of hardware is extremely expensive, especially for large constraint length  $K$ . 2) The architecture is applicable only when the look-ahead level is a multiple of  $K$ . In addition, it is not efficient in reducing the latency when the look - ahead level is not a power of 2 multiple of  $K$ .

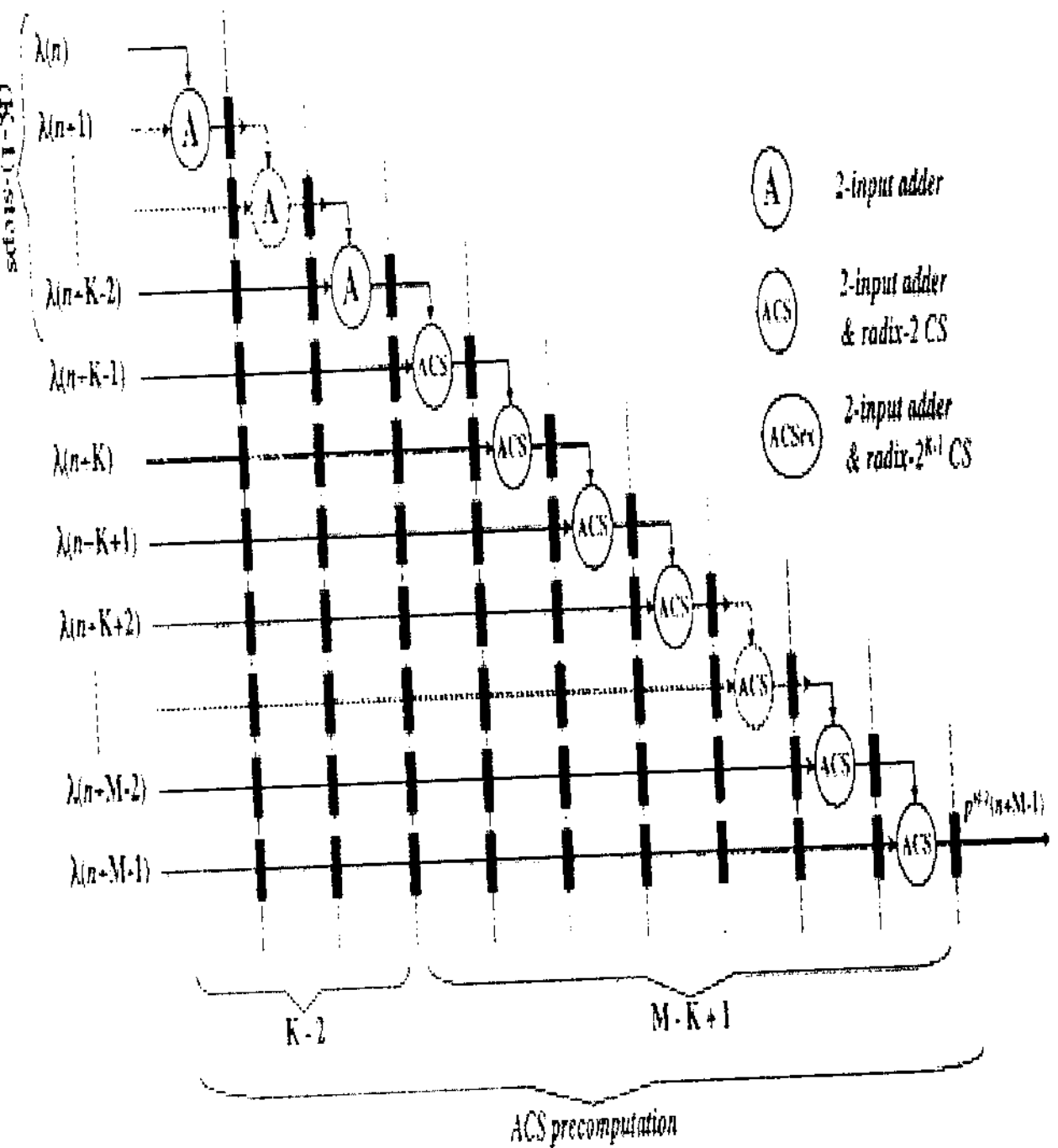


Figure 4.5: A conventional Viterbi decoder architecture

Look-ahead techniques have been widely used in ACS in order to achieve high throughput rate that must be met in high speed applications. Assume a Viterbi decoder with constraint length  $K$  and  $M$ -step look ahead. Instead of using one-step binary trellis branch metrics to update path metrics, multiple-step binary trellis branch metrics are fed into the ACS unit for updating. The iteration bound is broken and higher throughput rate will be achieved by pipelining ACS when the look-ahead level  $M$  is greater than  $K$ . For instance, 3 times throughput speed up can be achieved with  $M = 3K$ -level look ahead. However, the speed break-through is at the expense of higher computational complexity and longer delay in Viterbi decoders, which are caused by the BMP unit that combines multiple-step binary trellis into one-step complex trellis. For deep look-ahead architectures, the complexity and latency caused by BMP dominate the overall complexity and latency. For the sake of achieving low latency, various BMP architectures have been proposed in the past few years.

#### 4.6.2 LAYERED LOOK – AHEAD ARCHITECTURE

The goal is to avoid the linear step by step computation procedure by applying a  $K$ -nested layered look-ahead method.  $M/K$  layer-1 processors (P1s) first compute  $K$ -step branch metrics with one compare operation at the end of processing. In the following layer, it combine the trellis in a logarithmic manner with  $N$  and  $N-1$  add-compare operation in each layer-2 processor (P2). Since there are no parallel paths for each  $(K-1)$  look-ahead branch metric computation in a nest that combines  $K$ -trellis steps, only the additions are allowed for them as shown in figure 4.6. Each adder in A, ACS, and ACSrx is a two input adder.

Each ACS unit in the conventional ACS precomputation architecture and in the layer-1 processor (P1) of the proposed ACS precomputation architecture consists of radix-2 compare-select (CS) units. Each ACS unit in layer-2 and higher layer processors in the proposed architecture consists of radix-2 <sup>$K-1$</sup>  CS unit.

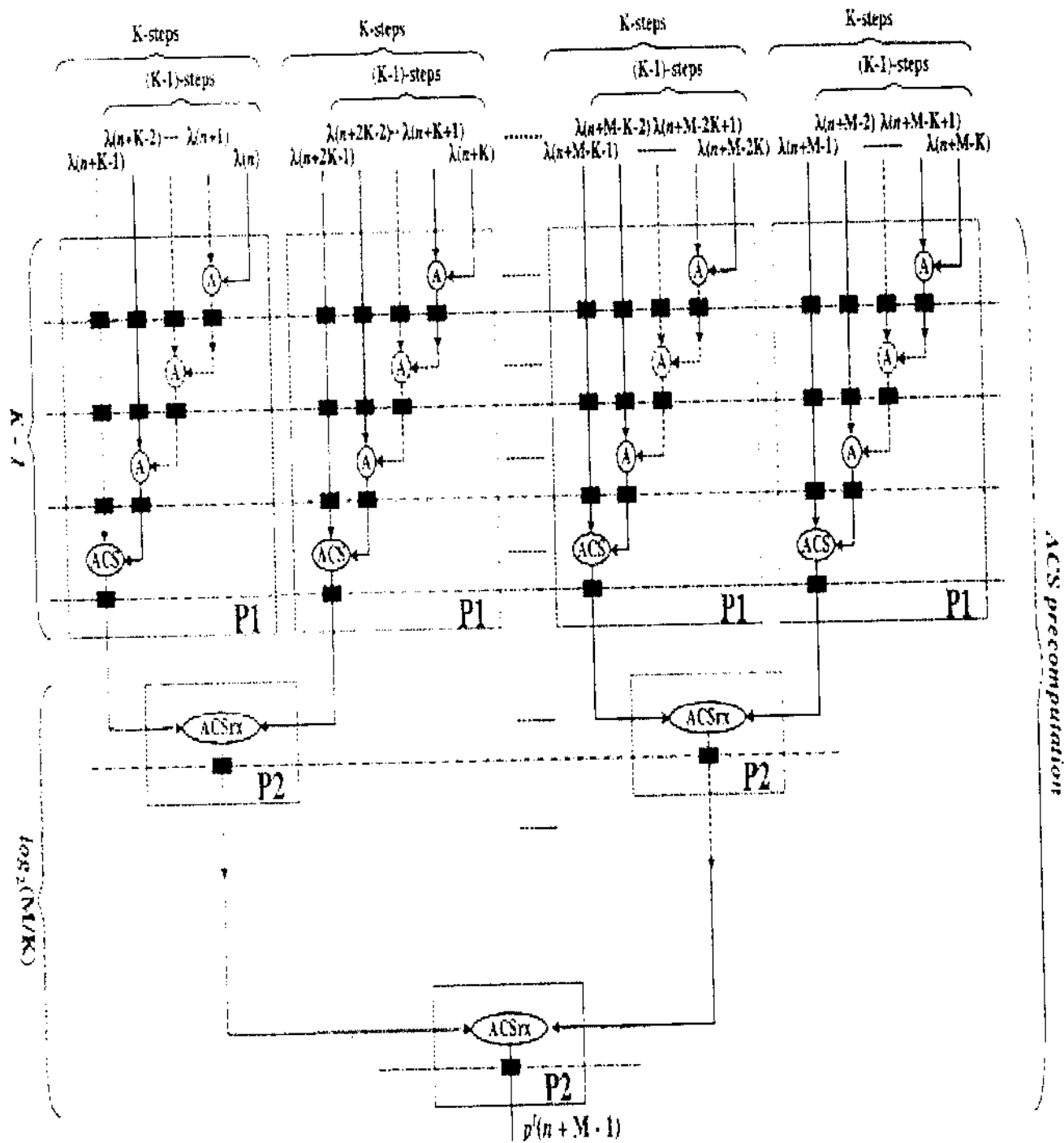


Figure 4.6: A Layered Viterbi decoder architecture

For both conventional and proposed architectures, the ACS recursion processor needs  $2^{K-1}$  ACS units. The main idea of the proposed method involves combining trellis steps as a pipeline structure and then combining the resulting look-ahead branch metrics as a tree structure in a layered manner to decrease the ACS precomputation latency. This leads to a regular and simple high throughput rate Viterbi decoder architecture with logarithmic increase in latency, as opposed to linear increase in conventional look-ahead, with respect to the look-ahead factor.

In the proposed nested LLA method, each layer-1 processor combines trellis steps into one trellis step ( $K$ -nesting). The higher layer processors combine two sub-layer trellises, and each layer processor executes the ACS computations in a distributed manner referred as layered look-ahead. The ACS recursion is executed at the top layer. For each trellis state, the layer-1 processor selects one path from two parallel paths and leads to a fully connected trellis with no parallel paths. When two fully connected trellises with no parallel paths are combined into one,  $2^{K-1}$  parallel paths are generated between any two-trellis states. Therefore, the layer-2 and higher layer processors select one path from  $2^{K-1}$  parallel paths instead of parallel paths shown in figure 4.6. The proposed  $mK$  step  $K$ -nested LLA method, where  $m$  is a positive power-of-two integer equal to, consists of  $(\log_2 m + 1)$  ACS precomputation layers and one ACS recursion layer. This method requires  $(2m-1)$  ACS precomputation units and one ACS recursion unit as shown in figure 4.6. A restriction of this method is that the look-ahead step should be a multiple of the encoder constraint.

The ACS computation without the addition of the previous state metrics (or path metrics or accumulated branch metrics) is referred as the ACS precomputation to distinguish it from the ACS recursion, which adds the previous state metrics to the look-ahead branch metrics. ACS computation units contain P1 and P2 for ACS precomputation and P3 for ACS recursion. The required numbers of processors for ACS computation are as follows: for P1, for P2, and 1 for P3. If  $m$  is not a power-of-two, slight modifications of the ACS precomputation and survivor path management are needed.

## 4.7 SURVIVOR PATH MEMORY UNIT

A register exchange consists of a two - dimensional array of one - bit registers and multiplexers as shown in figure 4.7. The registers in successive stages are interconnected to resemble the trellis structure of the convolutional code. A global clock signal controls the registers. The frequency of the clock determines the throughput of the Viterbi decoder. The path decision from each of the four ACSs is input to the register-exchange pipeline and also selects the outputs of a corresponding row of multiplexers.

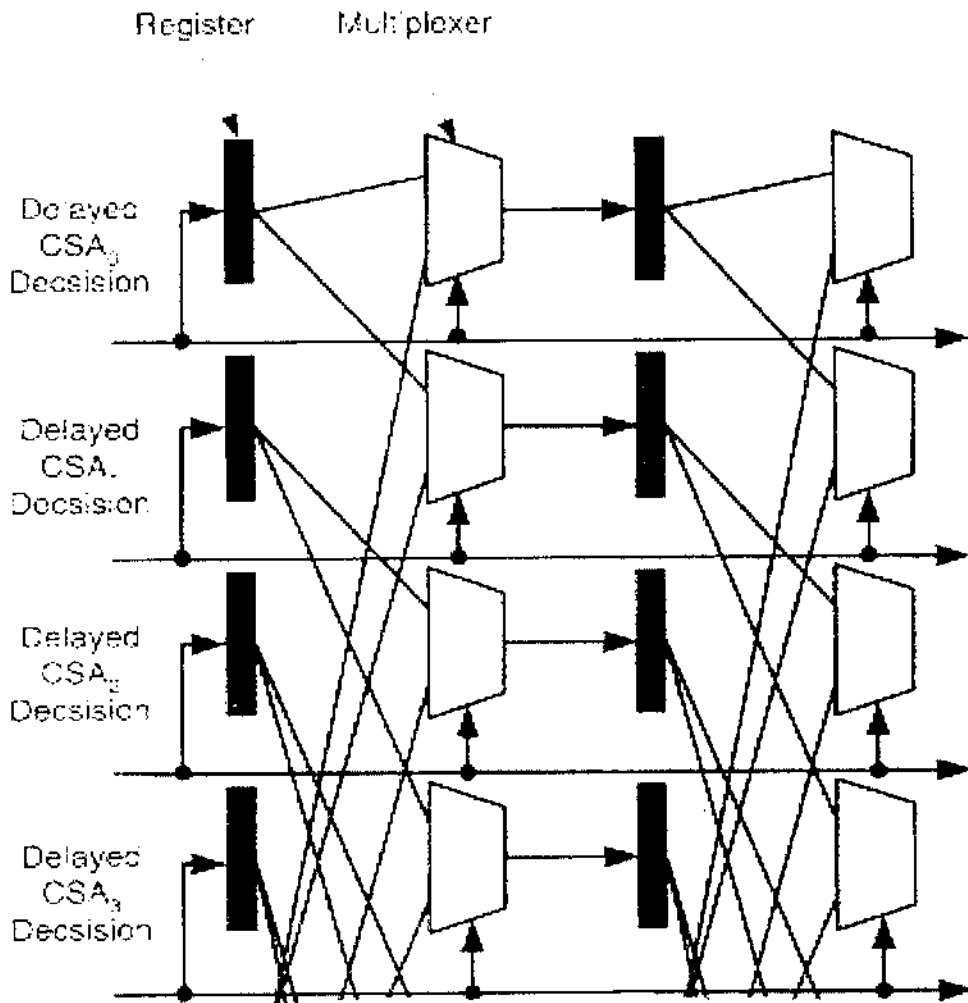


Figure 4.7: Survivor Path Memory Unit

As the number of states rises, the register exchange is required to shift an increasing number of bits through its pipeline. However, for decoders with a small number of states, the use of standard SRAM modules offers little power or area advantage over register exchange because of the overhead of peripheral circuitry and standard word addressing. Register exchange achieves high throughputs easily because its critical path only consists of a multiplexer and a register. Standard SRAM macros in 0.18- $\mu\text{m}$  technology have much longer cycle times than the synthesized CSA recursion. Hence, the register exchange is the appropriate structure for high-throughput implementations of a decoder with a small number of states.

The survivor path storage block is necessary only for the traceback approach. The block records the survivor path of each state selected by the ACS module. It requires one bit of memory per stage to indicate whether the survivor path is the upper one or lower one. It generates the decoded output sequence. In the traceback approach, the block incorporates combinational logic, which traces back along the survivor path and latches the path to a register. The BM unit is used to calculate branch metric for all 8 – trellis branches from the input data. The absolute difference is choose as a measure of branch metric.



## CHAPTER 5

### RESULTS AND DISCUSSION

The simulation of this project has been done using MODELSIM XE III 6.2g and implemented using XILINX ISE 9.2i.

Modelsim is a simulation tool for programming VLSI. Modelsim provides a comprehensive simulation and debug environment for complex ASIC and FPGA designs. Support is provided for multiple languages including Verilog, SystemVerilog, VHDL and SystemC. Here, the convolutional encoder and Viterbi decoder structure is implemented using VHDL code.

Xilinx was founded in 1984 by two semiconductor engineers, Ross Freeman and Bernard Vonderschmitt, who were both working for integrated circuit and solid-state device manufacturer Zilog Corp. The Virtex-II Pro, Virtex-4, Virtex-5, and Virtex-6 FPGA families are particularly focused on system-on-chip (SOC) designers because they include up to two embedded IBM PowerPC cores. The ISE Design Suite is the central electronic design automation (EDA) product family sold by Xilinx. The ISE Design Suite features include design entry and synthesis supporting Verilog or VHDL, place-and-route (PAR), completed verification and debug using Chip Scope Pro tools, and creation of the bit files that are used to configure the chip. Thus, the hardware implementation of the convolutional encoder and Viterbi decoder structure is done by using Xilinx.

## 5.1 SIMULATION RESULT

### A RATE 1/2 CONVOLUTIONAL ENCODER

Here,  $x$  is the input, produces two bits of encoded information for each bit of information and  $x = \{1011\}$ . When reset is made zero it produces the encoded output  $c$  as  $c = \{11, 10, 00, 01\}$ . The next state and the corresponding encoded outputs for the message input are shown.

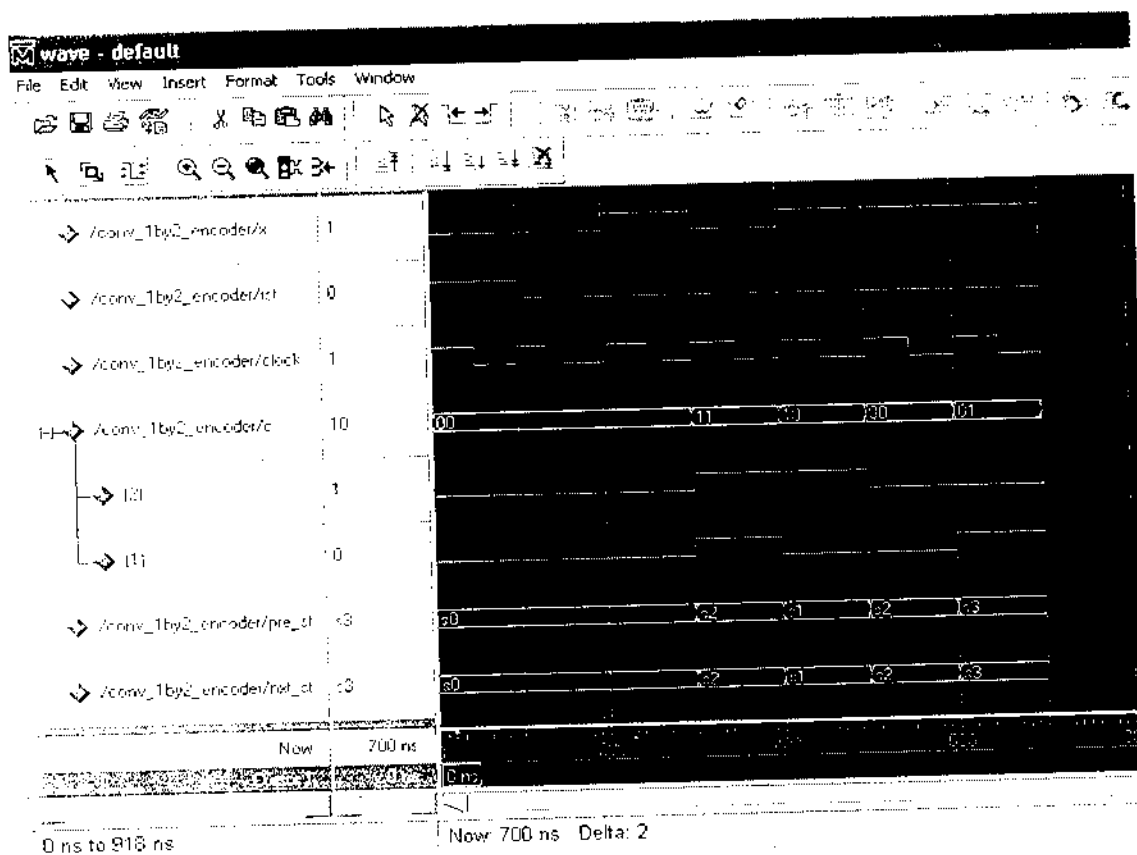
**Input and present state**                      **Next state and encoded outputs**

$x=1$   $s_0$                        $\rightarrow$                        $s_2$   $c=11$

$x=0$   $s_2$                        $\rightarrow$                        $s_1$   $c=10$

$x=1$   $s_1$                        $\rightarrow$                        $s_2$   $c=00$

$x=1$   $s_2$                        $\rightarrow$                        $s_3$   $c=01$



**Figure 5.1 Simulated result of rate 1/2 convolutional encoder**

## A RATE 2/3 CONVOLUTIONAL ENCODER

Here,  $u$  is the input, produces three bits of encoded information for two bits of information. When reset is made zero it produces the encoded output as  $v$ . The encoded output for the different message input are shown in figure 5.2.

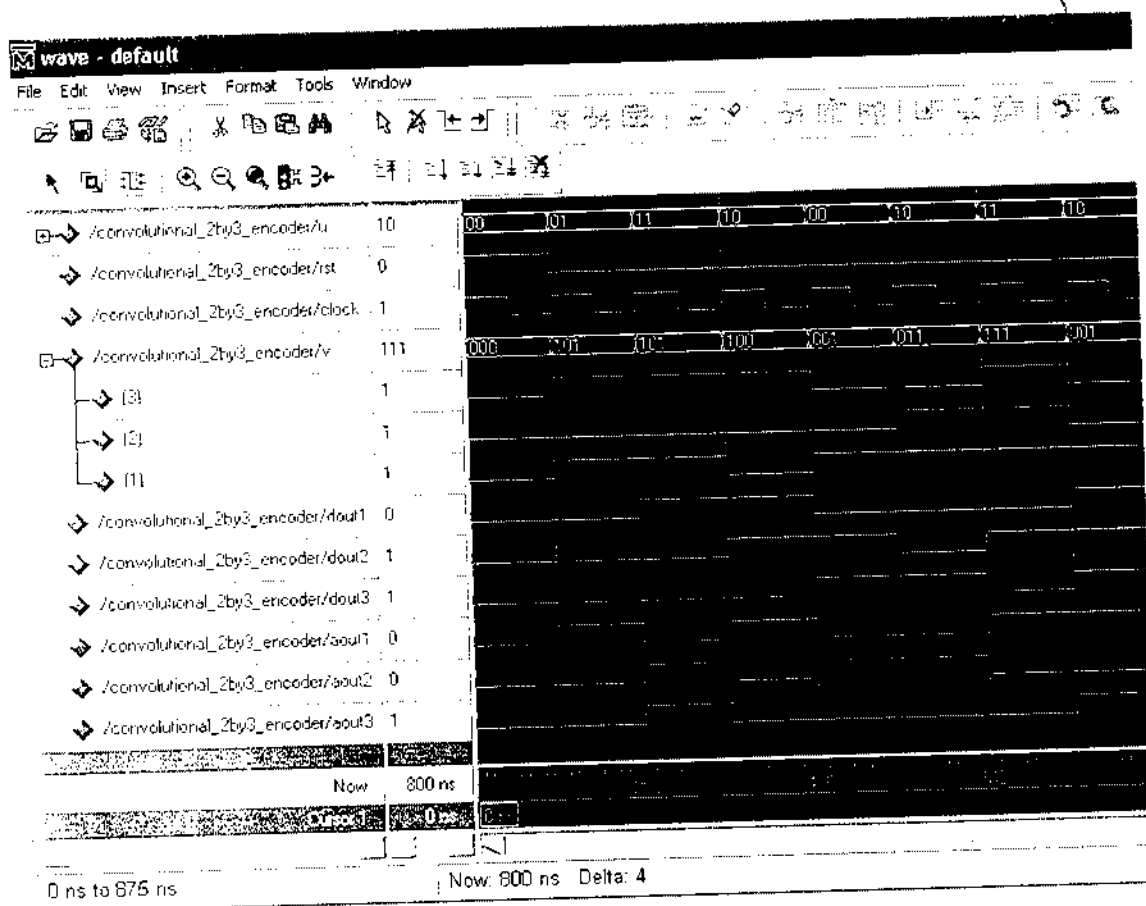


Figure 5.2 Simulated result of rate 2/3 convolutional encoder

# A RATE 2/5 CONVOLUTIONAL ENCODER

Here,  $m$  is the input, produces five bits of encoded information for two bits of information. When reset is made zero it produces the encoded output as  $c$ . The encoded output for the different message input are shown in figure 5.3.

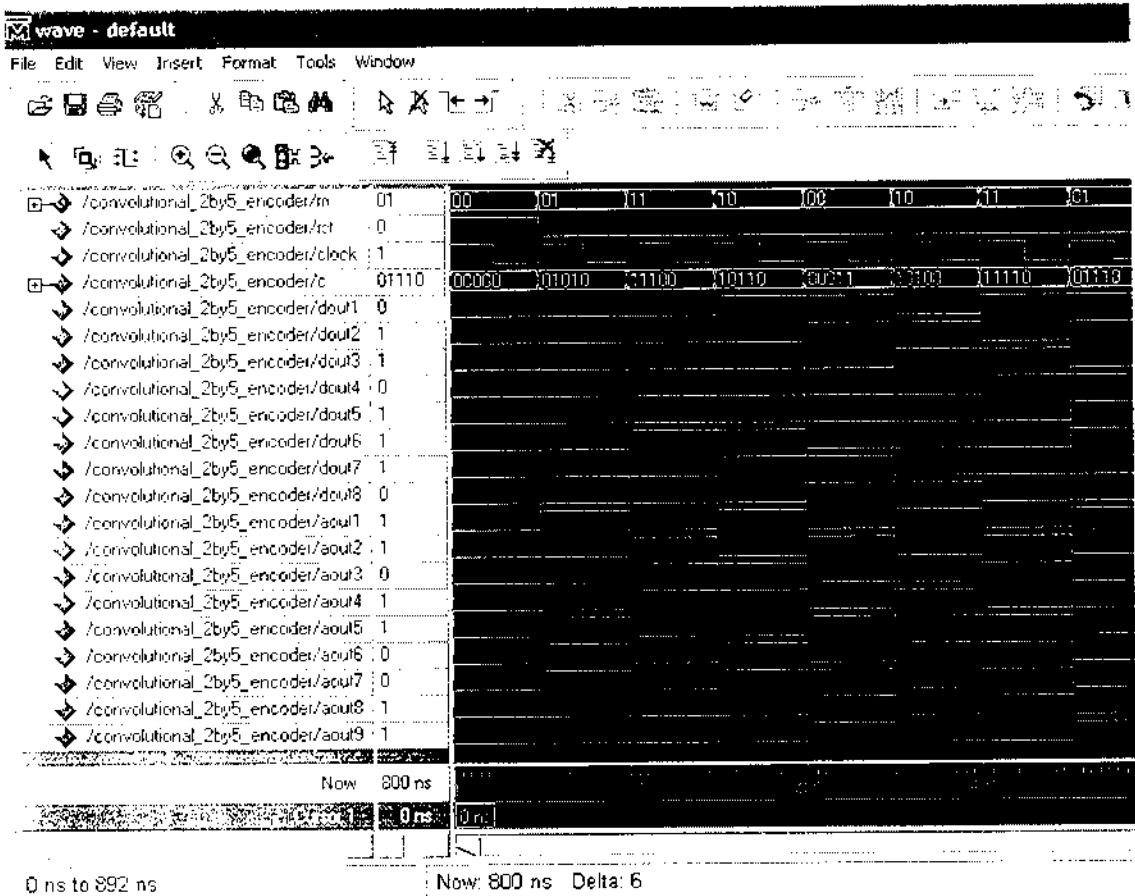


Figure 5.3 Simulated result of rate 2/5 convolutional encoder

# BRANCH METRIC UNIT

Branch metric is generated for the rate 1/2 convolutional encoder. Here, bm1 to bm4 are the inputs and bmu\_out1 to bmu\_out4 are the outputs.

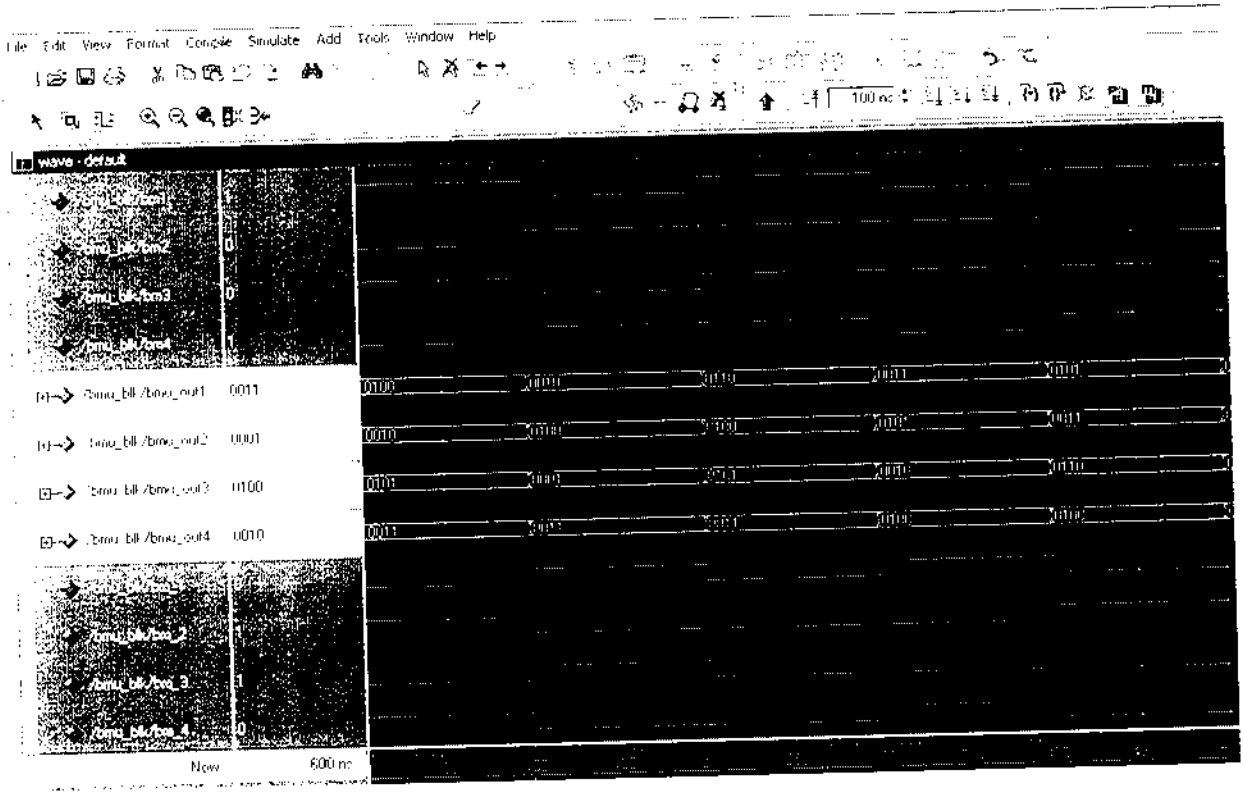


Figure 5.4: Simulated result of Branch Metric Block

# ADD – COMPARE – SELECT UNIT

Path metric is updated and decision bit is generated as shown in figure 5.5.

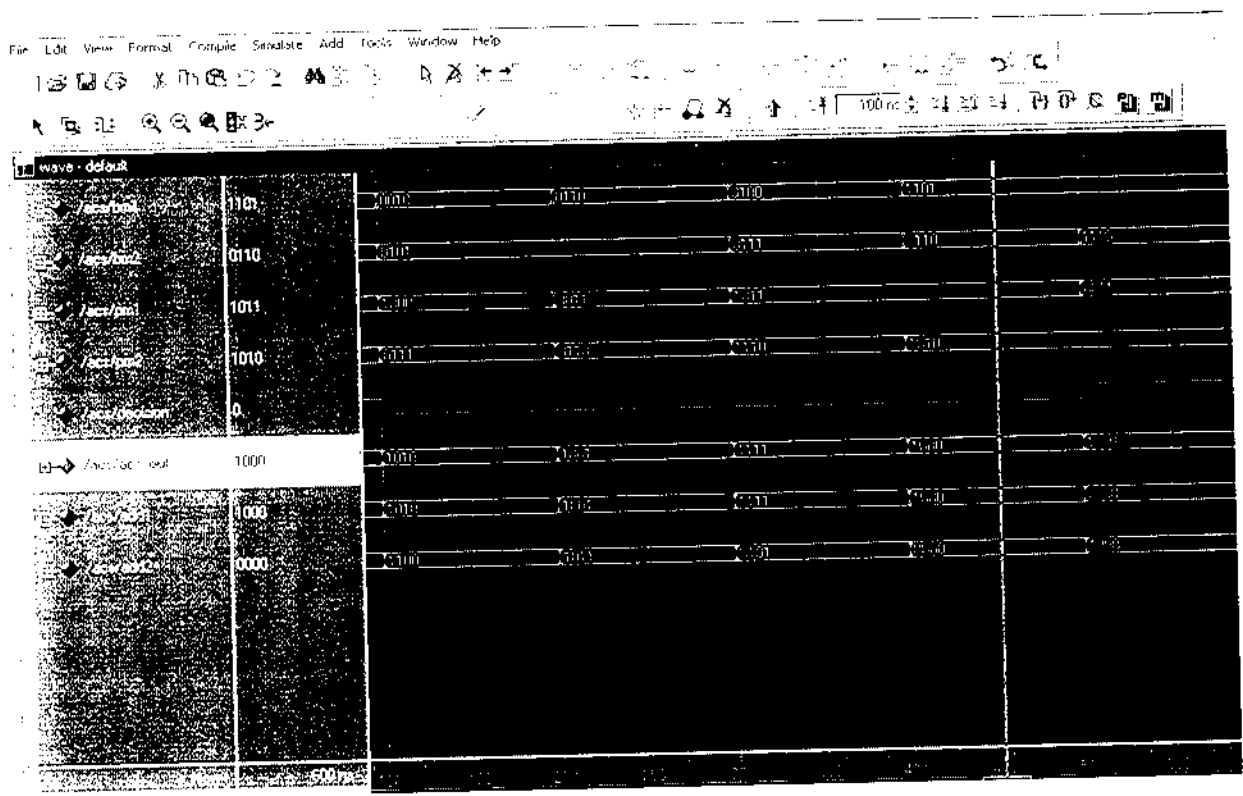


Figure 5.5: Simulated result of Add – Compare – Select Block

## BRANCH METRIC PRECOMPUTATION – CONVENTIONAL METHOD

Here, conventionally the branch metrics are precomputed step by step with pipelined registers inserted between two consecutive steps. Before the trellis is saturated, only add operation is performed and then compare operation is performed as shown in the figure 5.6.

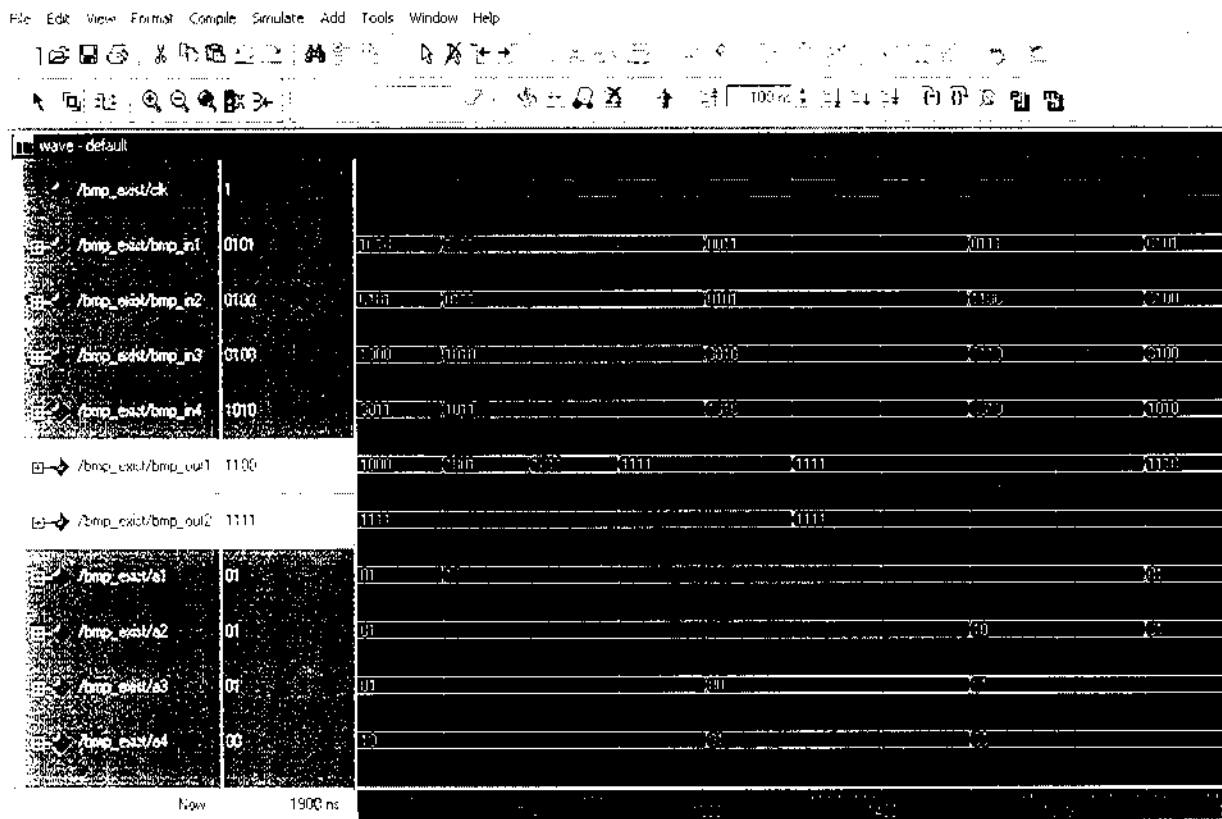


Figure 5.6: Simulated result of conventional precomputation

## BRANCH METRIC PRECOMPUTATION – LAYERED METHOD

Here, layer-1 processors (P1s) first compute  $K$ -step branch metrics with one compare operation at the end of processing. In the following layer, it combine the trellis in a logarithmic manner with  $N$  and  $N-1$  add-compare operation in each layer-2 processor (P2). The simulated result of Layered look-ahead method is shown in the figure 5.7 which reduces the latency.

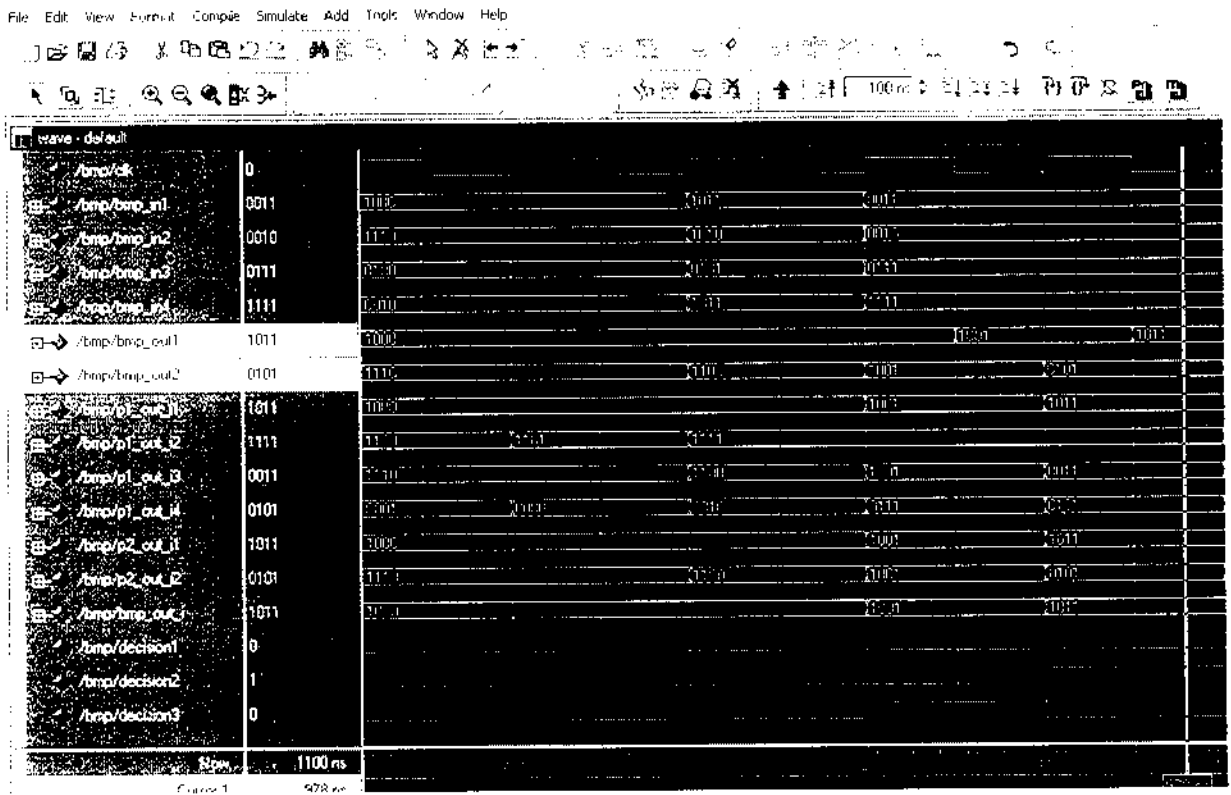


Figure 5.7: Simulated result of layered precomputation



## SURVIVOR PATH MEMORY UNIT

The path decision from each of the four ACSs is input to the register-exchange pipeline and it also selects the outputs of a corresponding row of multiplexers. The decoded data can be retrieved as shown in the figure 5.8.

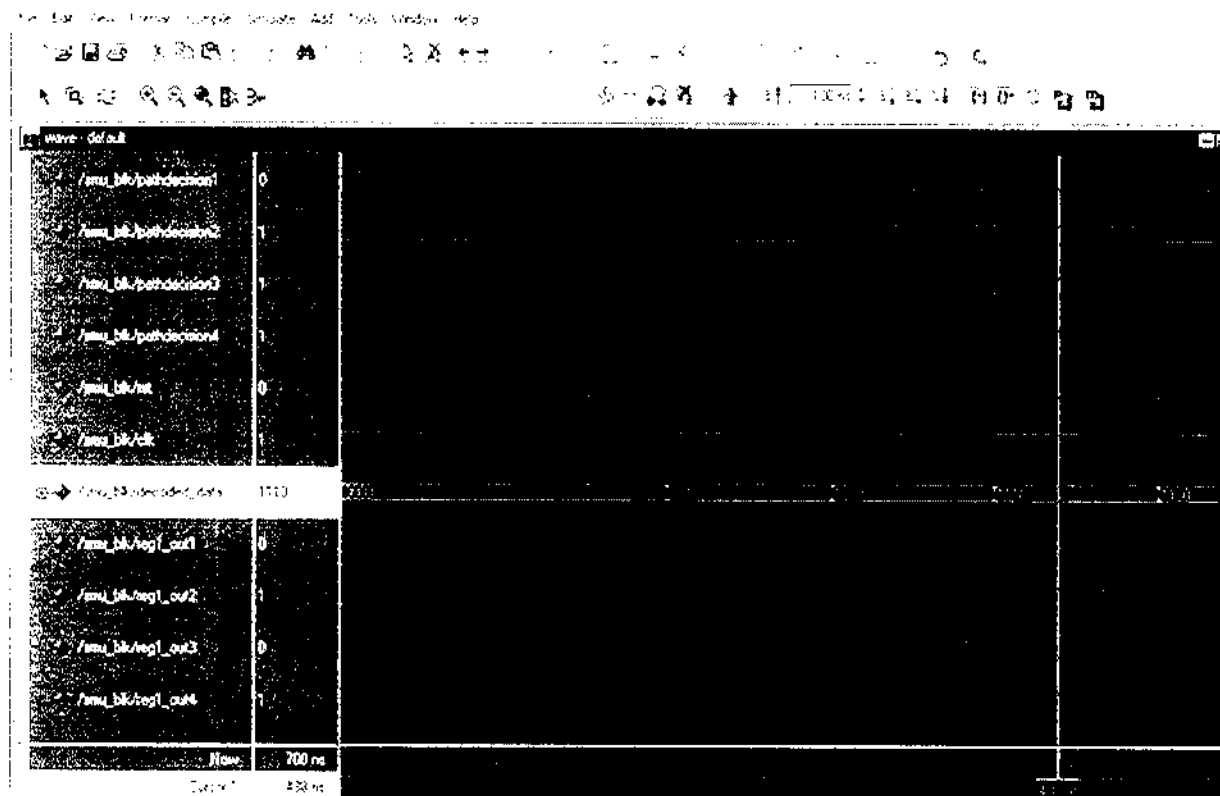


Figure 5.8: Simulated result of survivor path memory unit

## VITERBI DECODER – CONVENTIONAL METHOD

When implementing the blocks of the Viterbi decoder namely BMU, SMU in forward manner and ACS in recursive manner with the conventional method of BMP, the complexity and latency is increased. The simulated result is shown in figure 5.9.

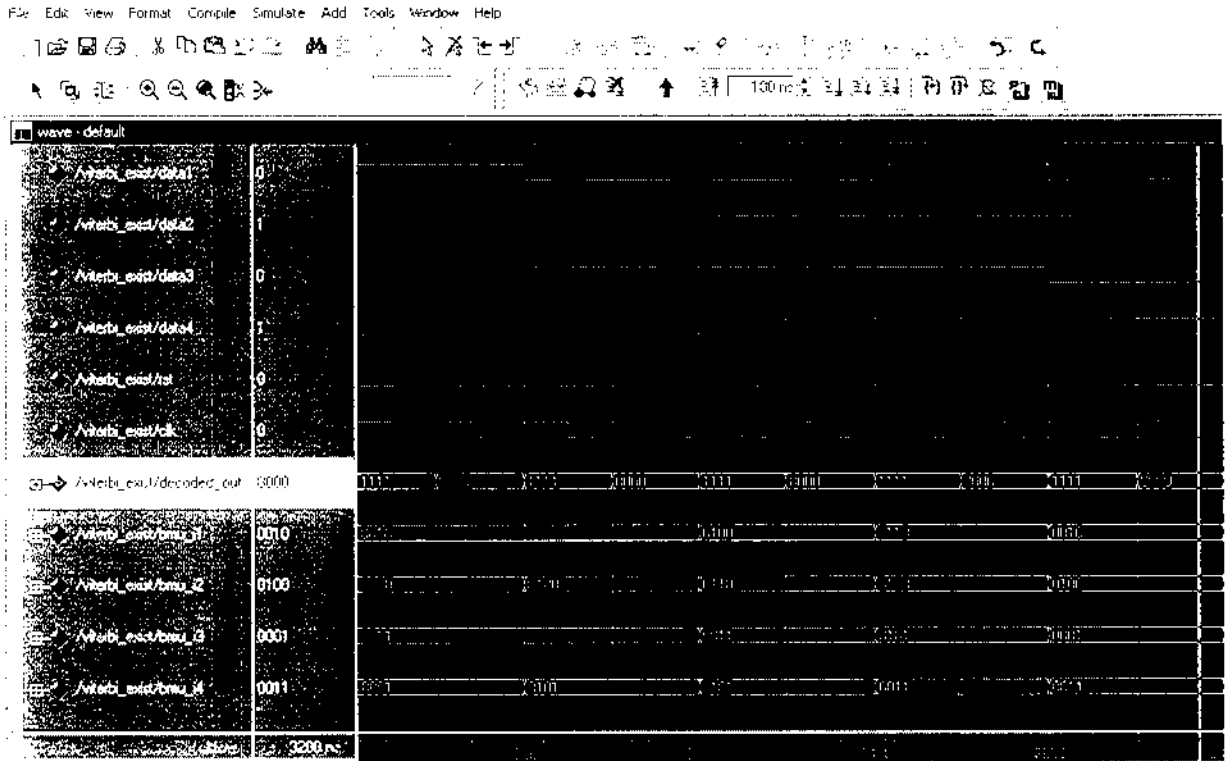


Figure 5.9: Simulated result of viterbi decoder in conventional method

## VITERBI DECODER – LAYERED METHOD

When implementing the blocks of the Viterbi decoder namely BMU, SMU in forward manner and ACS in recursive manner with the layered method of BMP which uses two processors, the complexity and latency is reduced comparatively. The simulated result is shown in figure 5.10.

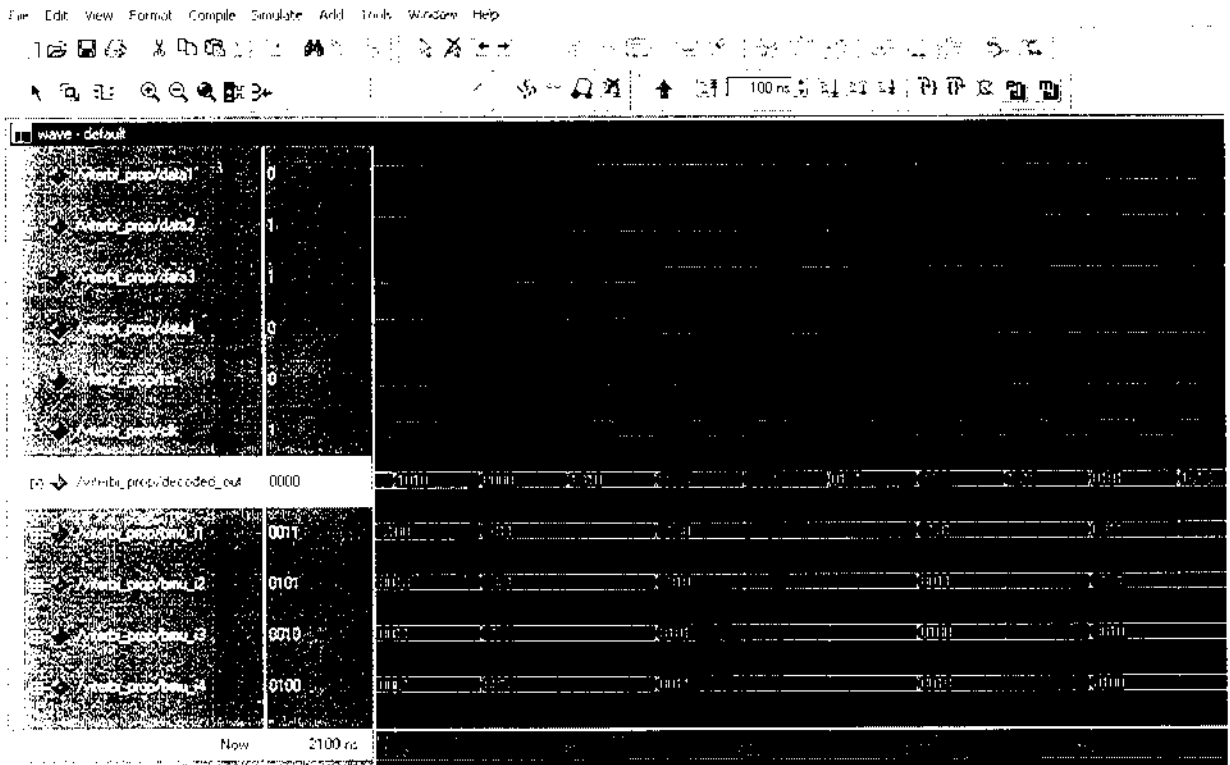


Figure 5.10: Simulated result of viterbi decoder in layered method

## 5.2 SYNTHESIS REPORT

### A RATE 1/2 CONVOLUTIONAL ENCODER

#### HDL SYNTHESIS REPORT

##### Macro Statistics

# Registers	: 2
Flip-Flops	: 2
# Xors	: 3
1-bit xor2	: 3

#### DEVICE UTILIZATION SUMMARY

Selected Device	: 3s100etq144-5
Total equivalent gate count for design	: 17
Number of Slices	: 1
Number of Slice Flip Flops	: 1
Number of 4 input LUTs	: 1
Number of IOs	: 5
Number of bonded IOBs	: 5
Number of GCLKs	: 1
Speed Grade	: -5

# A RATE 2/3 CONVOLUTIONAL ENCODER

## HDL SYNTHESIS REPORT

### Macro Statistics

# Registers	: 3
Flip-Flops	: 3
# Xors	: 6
1-bit xor2	: 6

## DEVICE UTILIZATION SUMMARY

Selected Device	: 3s100ctq144-5
Total equivalent gate count for design	: 45
Number of Slices	: 2
Number of Slice Flip Flops	: 3
Number of 4 input LUTs	: 3
Number of IOs	: 7
Number of bonded IOBs	: 7
Number of GCLKs	: 1
Speed Grade	: -5

## A RATE 2/5 CONVOLUTIONAL ENCODER

### HDL SYNTHESIS REPORT

#### Macro Statistics

# Registers	: 8
Flip-Flops	: 8
# Xors	: 12
1-bit xor2	: 12

### DEVICE UTILIZATION SUMMARY

Selected Device	: 3s100etq144-5
Total equivalent gate count for design	: 100
Number of Slices	: 4
Number of Slice Flip Flops	: 8
Number of 4 input LUTs	: 5
Number of IOs	: 9
Number of bonded IOBs	: 9
Number of GCLKs	: 1
Speed Grade	: -5

# VITERBI DECODER – CONVENTIONAL METHOD

## HDL SYNTHESIS REPORT

### Macro Statistics

# Adders/Subtractors	: 21
2-bit adder	: 9
4-bit adder	: 12
# Registers	: 156
1-bit register	: 148
4-bit register	: 8
# Comparators	: 4
4-bit comparator less	: 4

### DEVICE UTILIZATION SUMMARY:

Selected Device :	3s500eft256-5	
Number of Slices:	135 out of 4656	2%
Number of Slice Flip Flops:	77 out of 9312	0%
Number of 4 input LUTs:	249 out of 9312	2%
Number used as logic:	235	
Number used as Shift registers:	14	
Number of bonded IOBs:	10 out of 190	5%
Total equivalent gate count for design	: 2979	

## TIMING REPORT

Minimum period: 7.762ns (Maximum Frequency: 128.831MHz)

Minimum input arrival time before clock : 4.228ns

Maximum output required time after clock : 12.503ns

## VITERBI DECODER – LAYERED METHOD

### HDL SYNTHESIS REPORT

#### Macro Statistics

# Adders/Subtractors : 20

2-bit adder : 8

4-bit adder : 12

# Registers : 75

1-bit register : 65

4-bit register : 10

# Comparators : 7

4-bit comparator less : 7

1-bit xor2 : 4



## DEVICE UTILIZATION SUMMARY:

Selected Device : 3s500eft256-5

Number of Slices: 107 out of 4656 2%

Number of Slice Flip Flops: 92 out of 9312 0%

Number of 4 input LUTs: 201 out of 9312 2%

Number used as logic: 196

Number used as Shift registers: 5

Number of bonded IOBs: 10 out of 190 5%

Number of GCLKs: 1 out of 24 4%

Total equivalent gate count for design : 1572

## TIMING REPORT

Minimum period: 6.718ns (Maximum Frequency: 148.858MHz)

Minimum input arrival time before clock: 4.573ns

Maximum output required time after clock: 11.375ns

**Table 5.1: Comparison of Conventional and Layered method**

PARAMETERS	CONVENTIONAL METHOD	LAYERED METHOD
Number Of LUT's	2979	1572
Time Delay (ns)	7.762	6.718

## CHAPTER 6

### CONCLUSION AND FUTURE SCOPE

#### 6.1 CONCLUSION

Convolutional codes are useful for real - time applications because they can be continuously encoded and decoded. It is useful for low-latency communications. Using convolutional coding, the information can be extracted without any error from noisy channel if the SNR is big enough. When the SNR is lower than some value, there will be some error and the error rate increases as the SNR decreases. When the SNR is lower than some certain value, the convolutional encoder cannot extract the information.

Convolutional encoding with viterbi decoding is a FEC technique that is particularly suited to a channel in which the transmitted signal is corrupted mainly by Additive White Gaussian Noise. It can be used to improve the performance of wireless systems.

Viterbi decoder is designed which decodes the trellis code modulation to cope with the varying channel condition. Sacrifices in the speed of the trellis unit result in large hardware savings for the other processing blocks by applying computation rate matching. Synthesized designs that provide different transmission rate combinations show that the task flexibility inherent in the BM unit impacts the design size far more than the rate flexibility of trellis and SP units.

#### 6.2 FUTURE SCOPE

Viterbi decoders employed in digital mobile communications are complex in its implementation and dissipate large power. Here, the architecture for reduced complexity is implemented. Further, the asynchronous design techniques can be used to reduce power consumption. The asynchronous design based on Quasi Delay Insensitive timing model can be used for robust and low power applications.

## REFERENCES

- [1] J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. theory*, vol. IT - 13, no. 2, pp. 260 - 269, Apr. 1967.
- [2] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS Bottleneck," *IEEE Trans. Comm.*, vol. 37, no. 8, pp. 785 - 790, Aug. 1989.
- [3] G. Fettweis and H. Meyr, "High - rate Viterbi processor: A systolic array solution," *IEEE Sel. Areas Comm.*, vol. 8, no. 8, pp. 1520 - 1534, Oct. 1990.
- [4] P. J. Black and T. H. Y. Meng, "A 1 - Gb/s, four - state, sliding block viterbi decoder," *IEEE J. Solid - state Circuits*, vol. 27, pp. 1877 - 1885, Dec. 1992.
- [5] V. S. Gierenz, O. Weis, T. G. Noll, I. Carew, J. Ashley, and Karabed, "A 550 Mb/s radix- 4 bit-level pipelined 16-state CMOS Viterbi decoder," in *Proc. IEEE Int. Conf. Appl.- Specific Syst., Archit. Process.*, 2000, pp. 195-201.
- [6] Lee and J. L. Sonntag, "A new architecture for the fast Viterbi algorithm," *IEEE Trans. Commun.*, vol. 51, no. 10, pp. 1624-1628, Oct. 2003.
- [7] J. J. Kong and K. K. Parhi, "Low - latency architectures for high - throughput rate viterbi decoders," *IEEE Trans. VLSI Syst.*, vol. 12, no. 6, pp. 642 - 651, Jun. 2004.
- [8] M. Kling, "Channel Coding Application for CDMA2000 implemented in a FPGA with a Soft processor Core," Linköping University, Department of Electrical Engineering, 2005.
- [9] W. Chen, "RTL Implementation of Viterbi Decoder," Linköping University, Department of Electrical Engineering, June 2006.
- [10] S. Hema, V. S. Babu and P. Ramesh, "FPGA Implementation of Viterbi Decoder," *Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications*, Corfu Island, Greece, February 16-19, 2007.

- [11] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, 1999, ch. 2.
- [12] Cheng and K. K. Parhi, "Hardware efficient low-latency architecture for high throughput rate Viterbi decoders," *IEEE Trans. Circuit Syst. II. Expr. Briefs*, vol. 55, no. 12, pp. 1254 -1258, Dec. 2008.
- [13] R. Liu and K. K. Parhi, "Minimal complexity low-latency architectures for Viterbi decoders," in *Proc. IEEE Workshop SiPS*, Oct. 2008, pp. 140–145.