NOTE TO USERS

This reproduction is the best copy available.

UMI[®]

Viterbi Decoders for Mobile and Satellite Communications

by

Abdul Rafeeq Abdul Shakoor, B.Eng., India

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute of Electrical Engineering

Department of Electronics

Carleton University

Ottawa, Ontario, Canada

May, 2004

© Copyright 2004

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-97397-2

Our file Notre référence

ISBN: 0-612-97397-2

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

PAGINATION ERROR.

ERREUR DE PAGINATION.

TEXT COMPLETE.

LE TEXTE EST COMPLET.

The information used in this thesis comes in part from the research program of Dr. Tad Kwasniewski and his associates in the VLSI in Communications group. The research results appearing in this thesis represent an integral part of the ongoing research program. All research results in this thesis including tables, graphs and figures but excluding the narrative portions of the thesis are effectively incorporated into the research program and can be used by Dr. Kwasniewski and his associates for educational and research purpose, including publication in open literature with appropriate credits. The matters intellectual property may be pursued cooperatively with Carleton University and Dr. Kwasniewski and where and when appropriate.

Abstract

The design and FPGA implementation of configurable, high performance Viterbi decoders targeted for wireless and satellite communication transceivers, are presented in this thesis. Two Viterbi decoder circuits one for constraint length equal to 7 for satellite applications and the other for constraint length 9 for wireless transceivers have been realized. The performance and circuit characteristics were confirmed through simulation and functional verification. The designs for constraint lengths of 7 and 9 operate for code rates of $1/2$ and $1/3$. Furthermore, the trace-back length (TBL) can be reconfigured and set at values ranging from 35 to 64 in steps of five. The modular architecture adopted permits the integration of two ACS (add-compare-select) units to satisfy the computational requirements, which are a function of the constraint length and trace back length. The designs have demonstrated the effectiveness of scaling and look-up techniques in reducing circuit requirements associated with branch metric computation and path metric determination. Functional and VHDL code simulation results have confirmed performance improvement by 0.4 dB upon changing TBL from 35 to 50 for constraint length 7. FPGA implementation results based on Altera's APEX technology, have confirmed the Viterbi decoder throughput to be 2 Mbps.

Acknowledgments

I would like to thank my supervisors, Dr. Tad A. Kwasniewski and Dr. Valek Szwarc, for their guidance and support throughout my research at Carleton University and Communications Research Centre (CRC).

Communications Research Centre is gratefully acknowledged for providing the support for completing the research. I thank my colleagues at CRC for providing feedback on our technical papers before its submission.

I could not have fulfilled my vision of completing my research and thesis without the understanding and support of my family. Thank you for your love and support. Specially I would like to thank my wife Shabana, daughter Ayesha and son Mustafa for their love and understanding.

As a special tribute, I would like to dedicate this thesis in the loving memory of my father Abdul-Shakoor Abdul-Kareem.

Table of Contents

List of Figures	ix
List of Tables	xi
List of Abbreviations	xii
Chapter 1: Introduction	1
1.1 Background.....	1
1.2 Thesis Motivation.....	2
1.3 Thesis Objectives	2
1.4 Thesis Outline.....	3
Chapter 2: Convolutional Encoder and Viterbi Decoder Theory	4
2.1 Introduction.....	4
2.2 Basic Structure and General Properties of Convolutional Codes.....	5
2.2.1 Generator Polynomials (GP)	7
2.2.2 Coding Rate (R)	8
2.3 Operation of Convolutional Encoder	9
2.4 Convolutional Encoder State and Trellis Diagram.....	12
2.5 Generator Polynomials for the IMT 2000 Standard	13
2.5.1 GP for Constraint Length 7, Code Rate 1/2	14
2.5.2 GP for Constraint Length 7, Code Rate 1/3	15

2.5.3 GP for Constraint Length 9, Code Rate 1/2	16
2.5.4 GP for Constraint Length 9, Code Rate 1/3	17
2.6 Viterbi Decoder.....	18
2.6.1 Traceback Operation	26
2.7 Summary.....	27

Chapter 3: Viterbi Decoder Design and Implementation Issues.....

3.1 Introduction.....	29
3.2 Literature Review	29
3.3 IMT 2000 Technical Specifications for Viterbi Decoder.....	38
3.4 Viterbi Decoder Design Considerations	39
3.4.1 Add-Compare-Select Processor	39
3.4.2 Throughput	40
3.4.3 Number of ACS Units.....	41
3.4.4 Path Metric Rescaling or Optimization.....	43
3.4.5 Traceback Length or Window Size.....	44
3.5 Viterbi Decoder Parameters	44
3.6 Viterbi Decoder Applications in Communications.....	46
3.6.1 Applications to Satellite Communications	46
3.6.2 Applications to Mobile Communications.....	46
3.6.3 Other Applications of Viterbi Decoders	47
3.7 Summary.....	48

Chapter 4: Verification, Hardware Synthesis, FPGA Implementation, and Functional Modeling Results	49
4.1 Introduction.	49
4.2 Channel Model	49
4.3 Codec Link Model	51
4.4 Overall Design Flow.	52
4.4.1 Altera APEX EP20K400 Device Features	53
4.4.2 FPGA Design Cycle	53
4.4.3 Design Methodology	54
4.5 Viterbi Decoder Implementation Results	55
4.5.1 Complexity	55
4.5.2 Scalability and Programmability	58
4.5.3 Synthesis Results for Constraint Length $K=7$ and 9 and $R = 1/2$ and $1/3$	58
4.6 Test Methodology and Test Results	63
4.7 Simulation Results	67
4.7.1 Constraint Length $K=7$ and $R= 1/2$	68
4.7.2 Constraint Length $K=9$ and $R= 1/2$	70
4.8 Summary.	71
Chapter 5: Summary and Future Work	72
5.1 Summary.	72
5.2 Suggested Future Research.	74
Bibliography	75

List of Figures

Figure 2.1: Constraint length 3, rate 1/2 convolutional encoder.....	6
Figure 2.2: State diagram of a convolutional encoder for $K=3$ and rate $=1/2$	13
Figure 2.3a: Convolutional encoder for $K=7$ and $R=1/2$	15
Figure 2.3b: Convolutional encoder for $K=7$ and $R=1/3$	16
Figure 2.4a: Convolutional encoder for $K=9$ and $R=1/2$	18
Figure 2.4b: Convolutional encoder for $K=9$ and $R=1/3$	19
Figure 2.5: Trellis diagram for $K=3$ and $R=1/2$ Viterbi decoder	20
Figure 2.5a: Trellis position when stage 0 is being processed.....	21
Figure 2.5b: Trellis position when stage 1 is being processed.....	22
Figure 2.5c: Trellis position when stage 2 is being processed.....	23
Figure 2.6: A butterfly being processed by single ACS unit.....	24
Figure 2.7: Viterbi decoder architecture	25
Figure 3.1: ACS processor implementation based on single ACS unit	42
Figure 3.2: Application of Viterbi decoder in concatenated coding system	47
Figure 4.1: Functional model for convolutional encoder and decoder system	52
Figure 4.2: Block diagram of test bench incorporating: signal source, data encoder and decoder, data integrity comparator and error generator model	64
Figure 4.3: Block diagram of functional and hardware encoder and decoder configurations for the determination of bit error rates in the presence of AWGN	65

Figure 4.4: CRC developed adaptor board for the IMS-100 tester with an Altera EP20K1500 FPGA	67
Figure 4.5: Performance of Viterbi decoder for K=7 and R=1/2	69
Figure 4.6: Performance of Viterbi decoder for K=9 and R=1/2	70

List of Tables

Table 2.1: State table showing input bits, state transitions and output bit	11
Table 3.1: Representative Viterbi decoder designs	36
Table 3.2: Recommended technical specifications for WCDMA applications	38
Table 3.3: Technical specifications for Viterbi decoder implementation	45
Table 4.1: Altera APEX EP20K400 device features	53
Table 4.2: Logic Cell (LC) utilization based on a mapping onto Altera APEX device for data rate of 2 Mbps for K=7 and R=1/2 and 1/3	59
Table 4.3: Logic Cell (LC) utilization based on a mapping onto Altera APEX device for data rate of 2 Mbps for K=9 and R=1/2 and 1/3	60

List of Abbreviations

ACS	Add Compare Select
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BM	Branch Metric
BPSK	Binary Phase Shift Keying
CE	Convolutional Encoder
CMOS	Complementary Metal Oxide Semiconductor
CNTL	Control
FF	Flip Flop
FIFO	First In First Out
FPGA	Field Programmable Gate Array
IMT	International Mobile Telecommunications
ITU	International Telecommunications Union
K	Constraint Length
MLSE	Maximum Likelihood Sequence Estimation
PE	Processing Element
R	Code Rate
RAM	Random Access Memory
REM	Register Exchange Method
SNR	Signal to Noise Ratio
SPM	Survivor Path Memory

SRAM	Synchronous Random Access Memory
TBL	Trace Back Length
UMTS	Universal Mobile Telecommunications Standard
VD	Viterbi Decoder
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLSI	Very Large Scale Integration
WCDMA	Wideband Code Division Multiple Access
XOR	Exclusive OR Gate
3GPP	3rd Generation Partnership Projects Group

Introduction

1.1 Background

In real world communication, errors are introduced in messages sent from one point to another. Channel coding refers to the class of signal transformations designed to improve communications performance by enabling the transmitted signals to better withstand the effects of various channel impairments such as noise, fading and jamming. Channel coding involves addition of redundant bits to information bits being transmitted, this reduces the probability of received information bits being in error at the receiver. Channel coding is broadly divided into two forms of coding, namely block codes and convolutional codes. In most wireless communications systems of today, convolutional coding and Viterbi decoding is one of the preferred methods of error-correction coding to overcome transmission distortions. Convolutional encoding at the transmitter and Viterbi decoding at the receiver helps in error-correction for the data transmitted over the particularly challenging mobile communication channel. The data transmitted through the channel is corrupted primarily because of multiple reflections of the signals, the delay spread, and the amplitude distortion of the signals passing through noisy channels affected by Rayleigh fading, and white Gaussian noise.

1.2 Thesis Motivation

The algorithm development and implementation of the Viterbi decoder was carried out in the framework of a research agreement between the Communications Research Centre Canada (CRC) and the Centre for Development of Telematics (C-DOT) in India. The research agreement involved the investigation and development of selected circuits for a 3rd Generation Wideband CDMA (WCDMA) baseband processor. At C-DOT a Viterbi decoder design was developed in C-language and has been implemented on a TMS320 processor of Texas Instruments. This implementation required 60% of the resources of the TMS chip. In order to free up the resources of the TMS320 chip the Viterbi decoder circuit needed to be implemented in hardware, which could also be used in the baseband processing engine, and as a standalone VHDL core in other designs. The saved resources could be used for other baseband processing functions. The convolutional encoding and Viterbi decoding must meet the IMT 2000 standard requirements as proposed by The International Telecommunications Union (ITU), specifically the standard that relates to “Multiplexing and channel coding” [3Gpp02]. This standard makes reference to the convolutional encoding and Viterbi decoding scheme as well as to Turbo encoding and decoding.

1.3 Thesis Objectives

To satisfy the W-CDMA requirements and to assure maximum flexibility a scalable architecture is aimed. The objectives of the research are as follows:

- 1) To investigate and develop a scalable architecture for the Viterbi decoder for constraint lengths 7 and 9 and for code rates $1/2$ and $1/3$.
- 2) Investigate the use of multiple arithmetic computational units to augment throughput of

the design.

3) To develop programmable design that supports variable window sizes without degradation in throughput.

4) To investigate the performance of the Viterbi decoder as a function of its design parameters in presence of AWGN channel.

5) To apply testing methodology which will enable debugging and verification of the Viterbi decoder both at functional and system level.

1.4 Thesis Outline

The thesis is organized in the following manner. Chapter 2 provides the necessary theoretical background information on convolutional coding and Viterbi decoding. In Chapter 3, selected VLSI architectures of Viterbi decoders and their applications, in the area of mobile and data communication are reviewed. The proposed Viterbi decoder along with its advantages, implementation issues, and applications is also presented. Chapter 4 includes a presentation and discussion of the functional simulation, and implementation results. The functional modeling was performed in the Matlab simulation environment at the block level, while the implementation was done using VHDL. The design and architecture of a scalable and parameterizable Viterbi decoder is presented in Chapter 4. The results of the performance verification, hardware synthesis, FPGA implementation and logic optimization are also presented. Chapter 5 summarizes the thesis and presents recommendations for further research in the area.

Convolutional Encoder and Viterbi Decoder Theory

2.1 Introduction

In this chapter the architecture and operation of convolutional encoders and Viterbi decoders is reviewed. This chapter presents a detailed description of various design parameters of encoder and decoder, the state and trellis diagram, butterfly structure, and memory requirements during the traceback operation of the decoder.

Voice and data traffic transmitted by wireless devices is subject to noise and fading channel conditions, which results in the introduction of errors in the data stream. In order to maintain data integrity the data is normally encoded at the transmitter and decoded at the receiver. Convolutional encoding with Viterbi decoding and Turbo encoding with Turbo decoding is mentioned in the International Mobile Telecommunications (IMT)-2000 standard. This standard related to “Multiplexing and channel coding”, stipulates the use of a convolutional data encoder at the transmitter and a Viterbi decoder at the receiver. The Viterbi algorithm and decoder implementations have found wide acceptance in the communications industry and have been effectively implemented for both wireless and satellite communication applications [Clark88, Proakis89, Sklar88, Sweeney02, Lee97].

The technological and standards evolution from the low speed analog 2nd generation networks to the 3rd generation (3G) networks has seen an ever increasing need for effective error correction capability to mitigate channel noise interference. The constraints inherent in portable wireless devices impose additional performance and power requirements on error correction designs such as the Viterbi decoder.

The basic structure and properties of the convolutional encoder are examined in the next few sections while the Viterbi decoder is examined in section 2.6.

2.2 Basic Structure and General Properties of Convolutional Codes

A convolutional encoder can process information serially, continuously, or in short block lengths. A convolutional encoder has memory, in the sense that the output symbols depend not only on the input symbols, but also on previous inputs and/or outputs. In other words, the encoder is a sequential circuit or a finite state machine. The state of the encoder is defined as the contents of its memory [Zaragoza02]. An example of the schematic diagram of a convolutional encoder is shown in Fig. 2.1. This encoder generates two bits of encoded information for each bit of input information, so it is called a code rate 1/2 encoder. A convolutional encoder is generally characterized in (n, k, m) format, where

n - is number of encoder outputs

k - is number of encoder inputs

m - memory constraint length is the total number of shift register stages in the encoder, excluding any buffering of input and output bits.

Figure 2.1: Constraint length 3, rate 1/2 convolutional encoder.

Convolutional codes are specified by two parameters, r and K . The constraint length of the code is specified by parameter K , where K is the total number of bits involved in the encoding operation which is equal to $m + k$. The constraint length indicates the number of possible delay states (equal to $2^{(K-1)}$ or 2^m) which are needed at the decoder in order to decode the received sequence [Adamek91, Clark88, Lou95]. The code rate of the (n, k, m) encoder is specified by parameter r where $r = k/n$, n is the total number of coded bits generated by k input bits. The typical range of values for r and K are

$$\frac{1}{4} \leq r \leq \frac{7}{8} \quad 2.1$$

and

$$2 \leq K \leq 9 \quad 2.2$$

The encoder shown in the Fig. 2.1 is a (2,1,2) encoder with rate 1/2. In this thesis, the discussion is limited to the decoding of convolutional codes generated by (n, l, m) encoders with rate 1/2 and 1/3.

2.2.1 Generator Polynomials (GP)

In general, when referring to a convolutional code, the generators are expressed as $G_0(x) = 1, x, x^2, x^3, \dots, x^{n-1}$ and written in octal or binary form. For the encoder in Fig. 2.1, the output-bit combination is described by two generator polynomials, which represent the generation of each output bit from the input bit [Adamek91, Sweeney02]. The K=3 and rate 1/2 convolutional encoder shown in Fig. 2.1, is constructed using the following two generator polynomials:

$$G_0(x) = 1 + x + x^2 \quad 2.3$$

$$G_1(x) = 1 + x^2 \quad 2.4$$

The interpretation of these polynomials is that the first output bit is given by the XOR of the bit that has been delayed by two clock cycles (shown by the ' x^2 ' term in the equation 2.3), the bit delayed for one clock cycle (shown by ' x ' term in equation 2.3) and the current input data (shown by '1' in the equation 2.3). The second output is given by the XOR of the bit that has been delayed by two clock cycles (shown by the ' x^2 ' term in equation 2.4) and the current input data. These polynomials are often presented in the literature and in the IMT 2000 standard, using the octal format.

By combining the polynomial coefficients in groups of three, the octal and binary format expresses the polynomials as

$$G_0 = (7)_8 = (111)_2 \quad 2.5$$

$$G_1 = (5)_8 = (101)_2 \quad 2.6$$

The octal numbers 7 and 5 represent the code generator polynomials, which when read in binary (111_2 and 101_2) format correspond to the shift register interconnections to the gates G_0 and G_1 respectively. The selection of polynomials is important because each polynomial has different error-correcting properties [Adamek91, Lutz00]. For the same performance the higher the constraint length the lower will be the required SNR at the receiver.

2.2.2 Coding Rate (R)

Another factor influencing error correction is the coding rate $r = k/n$, the ratio of input data bits to the total number of coded bits generated. In Fig. 2.1, two bits are transmitted for each input bit for a coding rate of 1/2. Three generator polynomials are required for rate 1/3 systems. As a result an extra XOR gate is used at the encoding stage and the output of the encoder is the 3-bit symbol instead of a 2 bit symbol. Although any coding rate is possible, rate 1/2 systems are the most widely used [Wilson96, Zaragoza02]. In a system using FEC coding like convolutional coding, for every k information data bits, n coded bits are transmitted such that $n > k$. Since the transmission time is the same for a block of both coded and uncoded bits,

$$kT_b = nT_{bc} \quad 2.7$$

where T_{bc} is the duration of a coded bit and T_b is the duration of a data bit. The coded bit rate is

$$R_{bc} = \frac{1}{T_{bc}} \quad 2.8$$

upon substituting Equation 2.8 into Equation 2.7, we get

$$R_{bc} = \frac{n}{k} \cdot R_b = \frac{R_b}{r} \quad 2.9$$

where $r = k/n$ is the code rate. Since convolutional coding adds redundancy to the original data, the trade-off of using convolutional coding is the need for increased bandwidth. This bandwidth expansion is a function of the code rate as evident from Equation 2.9.

2.3 Operation of Convolutional Encoder

The convolutional encoder shown in Fig. 2.1 has memory (or two bit delay elements) which is represented by m . A convolutional encoder consists of one or more shift registers and multiple Exclusive OR gates (for a two input XOR gate; the output is high if one and only one input is high). The XOR gates are connected to some stages of the shift registers as well as to the current input to generate the output. There is no theoretical basis for the optimal location of the shift register stages to be connected to XOR gates. It is based on an empirical approach [Adamek91]. The location of stages is determined by the generator polynomial, which is the interconnection function. The location of stages as well as the number of memory elements determines the minimum Hamming distance. For every information bit encoded at the transmitter the four possible symbols which could be transmitted are 00, 10, 01, 11 and these symbols are treated as expected symbols at the receiver.

Suppose that “00” is transmitted then there is a possibility that it could be received as “00” (no error) or “10” or a “01” (single error) or it could be received as “11” resulting in a double error. Similarly if “10”, “01” or “11” is transmitted then four possible symbols could be received. When a symbol is received from the channel, say “00”, then the possible Hamming distances (HD) or free distance (d_{free}) after comparison to expected symbol “00” are

$$\text{HD} <00, 00> = 0; \quad 2.10$$

$$\text{HD} <00, 01> = 1; \quad 2.11$$

$$\text{HD} <00, 10> = 1; \quad 2.12$$

$$\text{HD} <00, 11> = 2; \quad 2.13$$

For a convolutional code, an important parameter is the maximum free distance which specifies the number of errors correctable by the code [Adamek91]. For example, $d_{\text{free}} = 5$ for the (2,1) convolutional code shown in Fig. 2.1 with generator polynomials shown in Equations 2.3 and 2.4 [Zaragoza02]. Hence, in this case, (2,1) code with constraint length equal to 3 corrects up to double errors since $t=2$. As per the proposition reported in [Adamek91] “A convolutional code corrects t errors if and only if its free distance is greater than $2t$ ”.

Generator polynomials or interconnection functions for different rates and different number of memory elements and their minimum Hamming distances are available in the literature [Adamek91, Lee97, Sweeney02].

The stream of information bits flows into the shift register from one end and is shifted out at the other end. Both Flip-Flops (F_2 and F_1) are reset at circuit initialization. Thus, when a single bit of data is clocked into F_2 , F_1 is still in reset mode. The two output bits C_0 and C_1 , which constitute a symbol, are generated and transmitted at every clock cycle. The second input bit is then moved

into the shift register and all the other bits are shifted to the right (the right most stored bit being lost). In this way a stream of input data bits is encoded and transmitted continuously.

A $2^{(K-1)}$ state Viterbi decoder is required to decode $(K-1)$ state convolutionally encoded data originating from a constraint length K encoder [Zaragoza02]. If the state of the encoder is taken as the content of the registers, with the oldest bit to the right and most recent to the left, the possible cases are shown in the state transition Table 2.1 and the final output of the encoder resulting from the input sequence is shown in the last column. The starting state for any operation can easily be found from the previous two inputs. Note that at the end of the sequence the encoder can be cleared by applying two more zeros to the input. This is because the decoding will be more robust if the encoder finishes in, as well as starts from, a known state, which is an all-zero state.

Table 2.1: State table showing input bits, state transitions, and output bits.

Start state $F_2(i), F_1(i)$	Input Data(i)	Final State $F_2(i+1), F_1(i+1)$	Outputs $C_0(i)$ and $C_1(i)$
00	0	00	00
00	1	10	11
10	0	01	10
01	1	10	00
10	0	01	10
01	1	10	00
10	0	01	10
01	1	10	00
10	0	01	10
01	0	00	11

Encoding Example.

Let the input sequence be 0 1 0 1 0 1 0 0 (ten bit input)

The left most bit in the above sequence is the first bit going into the encoder.

Assume that the shift register is initially in reset mode. At first clock cycle the first input bit which is zero in above example, is available to the encoder. The flip-flop outputs are both zero. Also the inputs to the G_0 and G_1 gates are both zero, so the output of the encoder is 00 (C_0C_1). The second clock cycle makes the second input bit available to the encoder which is binary 1 (in the example above). The F_2 clocks in the next bit, which is a zero, and F_1 clocks in the zero output from the F_2 . The inputs to the top most XOR gate G_0 are $(100)_2$, so the output is a one. The inputs to the bottom gate G_1 are $(10)_2$, so the output is also a one. Hence the encoder outputs $(11)_2$, as a channel symbol. In the next clock cycle another bit is shifted into F_2 and the process of shifting and computing continues. A 16 bit encoded sequence is generated as output for the example input sequence. Therefore the encoded output sequence would consist of 16 bits or 8 symbols.

2.4 Convolutional Encoder State and Trellis Diagram

Figure 2.2 represents the state diagram of the encoder shown in Fig. 2.1. Figure 2.2 depicts state transitions and the corresponding encoded outputs. As there are two memory elements in the circuit, there are four possible states that the circuit can assume. These four states are represented as 00, 01, 10, and 11. Each state's information (i.e. the contents of flip-flops for the state) along with an input generate an encoded output code. For each state, there can be two outgoing transitions; one corresponding to a '0' input bit and the other corresponding to a '1' input bit.

Figure 2.2: State diagram of a convolutional encoder for K=3 and R=1/2.

2.5 Generator Polynomials for the IMT 2000 Standard

As the constraint length K of an encoder is varied the performance of the decoder varies. Typically, the larger the K value, the better is the performance of the decoder. It is reported in the literature [Sweeney02, Sklar88] that with an increase in the constraint length the performance of the decoder also increases along with increased complexity of the Viterbi decoder implementation [Pandita99]. In the current research work three different constraint length convolutional encoders and decoders are considered. The first one being constraint length 3 encoder, which is primarily

used for explanation purposes. The second one is the convolutional encoder for constraint length 7 and the third is with constraint length 9. As the constraint length increases the complexity of decoding increases exponentially and the advantage of increased complexity is low BER [Sklar88, Lou95].

2.5.1 GP for Constraint Length 7, Code Rate 1/2

Among the most widely used convolutional codes is one with a memory $m = 6$ ($K=7$), rate 1/2 with generators (171, 133) and another with a memory $m=6$ ($K=7$), rate 1/3 and with generators (171,133, 145). These codes have come to be known as the NASA Planetary Standard Codes [Lin83]. In the 1977 launch of the Voyager space mission to Mars, Jupiter, and Saturn, NASA used a codec designed by Jet Propulsion Laboratory which can operate with either of these two codes [Peterson95, Zaragoza02]. It is also used in many digital communications standards [Wilson96], as well as in the newly adopted *IEEE 802.16* standard for broadband wireless communications. The code rate is defined in the same way as it was described for constraint length 3 encoder. Following are the generator polynomials for $K=7$, rate 1/2 encoder.

Generator Polynomials

The output-bit combination is described by following generator polynomials

$$G_0(x) = 1 + x + x^2 + x^3 + x^6 \quad 2.14$$

$$G_1(x) = 1 + x^2 + x^3 + x^5 + x^6 \quad 2.15$$

and the corresponding convolutional code properties are:

free distance(d_{free})=10 [Lin83] and correctable bits=4 (since $t=4$).

By combining the polynomial coefficients of Equations 2.14 and 2.15 in groups of three, the octal and binary format representations of those polynomials are as follows:

$$G_0 = (171)_8 = (001\ 111\ 001)_2 \quad \mathbf{2.16}$$

$$G_1 = (133)_8 = (001\ 011\ 011)_2 \quad 2.17$$

Figure 2.3a: Covolutional encoder for K=7 and R=1/2.

2.5.2 GP for Constraint Length 7, Code Rate 1/3

Generator Polynomials

The output-bit combination is described by following generator polynomials

$$G_0(x) = 1 + x + x^2 + x^3 + x^6 \quad 2.18$$

$$G_1(x) = 1 + x^2 + x^3 + x^5 + x^6 \quad \mathbf{2.19}$$

$$G_2(x) = 1 + x + x^4 + x^6 \quad \mathbf{2.20}$$

with convolutional code properties:

$d_{\text{free}}=15$ [Lin83] and correctable bits =7 (since $t = 7$).

By combining the polynomial coefficients in groups of three, the octal and binary format expresses the polynomials as:

$$G_0 = (171)_8 = (001\ 111\ 001)_2 \quad 2.21$$

$$G_1 = (133)_8 = (001\ 011\ 011)_2 \quad 2.22$$

$$G_2 = (145)_8 = (001\ 100\ 101)_2 \quad 2.23$$

Figure 2.3b: Convolutional encoder for K=7 and R=1/3.

2.5.3 GP for Constraint Length 9, Code rate 1/2

The constraint length is increased from 7 to 9 by increasing the encoder memory (m) from 6 to 8. Increased encoder memory results in greater protection. At the same time the complexity of the maximum likelihood decoding algorithm increases exponentially and the algorithm is more difficult

to implement because of the increased number of states and the very high number of computations required [Blahut03, Barbulescu96, Lou95]. The IMT 2000 standard for WCDMA recommends an 8 state ($K=9$) convolutional encoder with rate $R=1/2$ in the forward link (base to mobile station) and the same encoder but with rate $R=1/3$ in the reverse link (mobile to base station) [3Gpp01].

Generator Polynomials

The output-bit combination is described by following generator polynomials

$$G_0(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8 \quad 2.24$$

$$G_1(x) = 1 + x^2 + x^3 + x^4 + x^8 \quad 2.25$$

with convolutional code properties:

$d_{\text{free}} = 12$ and correctable bits = 5 (since $t = 5$).

The polynomials in Equations 2.24 and 2.25 can be expressed in octal and binary format as:

$$G_0 = (561)_8 = (101\ 110\ 001)_2 \quad 2.26$$

$$G_1 = (753)_8 = (111\ 101\ 011)_2 \quad 2.27$$

2.5.4 GP for Constraint Length 9, Code Rate 1/3

Generator Polynomials

The output-bit combination is described by following generator polynomials:

$$G_0(x) = 1 + x^2 + x^3 + x^5 + x^6 + x^7 + x^8 \quad 2.28$$

$$G_1(x) = 1 + x + x^3 + x^4 + x^7 + x^8 \quad 2.29$$

$$G_2(x) = 1 + x + x^2 + x^5 + x^8 \quad 2.30$$

The polynomials can be expressed in octal and binary format as:

Figure 2.4a : Convolutional encoder for K= 9 and R=1/2.

$$G_0 = (557)_8 = (101\ 101\ 111)_2 \quad 2.31$$

$$G_1 = (663)_8 = (110\ 110\ 011)_2 \quad 2.32$$

$$G_2 = (711)_8 = (111\ 001\ 001)_2 \quad 2.33$$

2.6 Viterbi Decoder

In modern digital communication systems, the computationally efficient Viterbi algorithm is widely used for the optimum decoding of convolutional codes. In this section, the architecture and the operation of the proposed Viterbi decoder are presented. For purposes of illustration the operation of the Viterbi decoder with a constraint length $K=3$ encoder will be considered by means of a corresponding four state trellis diagram shown in Fig. 2.5. The nodes in a trellis diagram correspond to the states of the convolutional encoder. The constraint length determines the number of states in the trellis diagram. If the constraint length is equal to 3 then the number of possible states is $2^{K-1} = 4$. Knowledge of the states is very useful in data decoding. Memory elements are labeled as

Figure 2.4b: Convolutional encoder for $K=9$ and $R=1/3$.

F_2, F_1 in Fig. 2.1. States in the trellis diagram are labeled in binary format as “00,01,10,11”. For example, if $F_2, F_1 = 11$ for the encoder, the corresponding state is 11, which is state three in the trellis diagram. The state number ranges from 0 to 3 for $K=3$, 0 to 63 for $K=7$ and 0 to 255 for $K=9$ Viterbi decoder.

The encoded data is decoded through the knowledge of the possible state transitions, created from the dependence of the current symbol on past data while encoding the information. The Viterbi algorithm is a maximum-likelihood means for decoding convolutional codes and has played an important role in applications ranging from satellite communications to cellular telephony.

The function of the Viterbi algorithm is to estimate the maximum-likelihood (ML) path through the trellis (shown in Fig. 2.5) based on received symbols. The algorithm reconstructs the actions of the encoder by selecting the most likely path in a structured graph, corresponding to a

Figure 2.5: Trellis structure for constraint length = 3, $R = 1/2$ Viterbi decoder.

tree or trellis that describes all possible states as shown in Fig. 2.5. Stated differently the Viterbi algorithm is a search algorithm, which for a given received symbol, searches all states of the trellis and determines the most likely information sequence by computing Hamming distances for each path through the trellis.

Convention: In writing a sequence, the leftmost symbol is meant to be the first symbol that goes into the channel.

A trellis diagram is an extension of a state diagram that explicitly shows the passage of time.

When a new symbol is received from the channel by the decoder, the branch-metric (BM) unit calculates the Hamming distances for all possible state transitions in the trellis. The calculation of Hamming distances starts from the “00” state as shown in Fig. 2.5. The encoder is flushed with zeros at the end of each frame. Thus the decoder, which performs frame by frame decoding, starts from state “00”. The computation of branch metrics is based on a comparison of current input symbol with the expected value. The Hamming distance corresponds in value to the number of positions in which the current and expected symbols differ. Thus in Fig. 2.5a the Hamming distances of the two branches determined at stage 0 have a value of ‘0’ and ‘2’ respectively. The Hamming distances or path metrics are identified in bold face at each node of the trellis in Figures 2.5a, 2.5b, and 2.5c.

Figure 2.5a: Trellis position when stage 0 is being processed.

At stage 1 as shown in Fig. 2.5b, the second received symbol “00” is processed. First the branch metrics are computed for all 4 incoming paths, then to these branch metrics the 2 path

metrics computed at stage 0 are added. Therefore, 4 new path metrics are computed at the four nodes A, C, F, and G.

At stage 2, the third received symbol "11" is processed. As shown in Fig. 2.5c, each one of the nodes B, E, D and H has two incoming paths. The branch metrics are computed for all 8 paths. Let us consider the two branches converging at node B. An ACS (Add-Compare-Select) unit adds the branch metric between nodes A and B to the partial path metric at node A and also it adds the branch metric between nodes C and B to the partial path metric at node C. Subsequently a comparison operation is done by the ACS unit between the two path metrics arriving at node B and the one

Figure 2.5b: Trellis position when stage 1 is being processed.

The ACS unit performs a comparison and selection of the path metrics stored in the SMS unit with the resulting survivor path stored in the survivor path storage (SPS) unit. Proceeding this way the whole 4 state trellis is traversed and survivor paths are stored for each state for up to 15 stages, before decoding starts. One survivor path is stored for each state in the trellis per each stage. For a four state, 15 stage decoding window, 60 bits are stored in the SPS unit. Once the whole trellis has been scanned and processed and a survivor path determined, the traceback unit in the output block takes over.

Specifically for the constraint length $K=7$ Viterbi decoder implementation the SMS and SPS units have greater dimensionality and the ACS unit operates at higher clock rates. The SMS unit stores 64 states per stage and processes the data for all stages in 32 master clock cycles or one clock cycle at the symbol data rate. The memory required to implement this block is $10 \times 128 \times 40$ bits where 10 is the path metric word size, 128 is the number of path metrics per stage, and 40 corresponds to the trace back length or the decoding window. The SPS unit stores 64 survivor paths per stage. The memory required to implement this block consists of $1 \times 64 \times 40$ bits. The SPS unit stores 256 survivor paths per stage for $K=9$ and hence in this case memory required to implement $TBL = 45$ would be $1 \times 256 \times 45$ bits.

Figure 2.6: A butterfly being processed by single ACS unit.

The major building blocks of a Viterbi decoder are shown in Fig.2.7. There are six blocks in a Viterbi decoder, and the function of each block is described below.

Figure 2.7: Viterbi decoder architecture.

Branch Metric: This unit calculates the branch metric of each stage in the trellis. It also calculates the Hamming distances between the received symbol and expected symbol.

ACS: The Add-Compare-Select unit receives two branch metrics and the state metrics. An ACS module adds each incoming branch metric of the state to the corresponding state metric and compares the two results to select the smaller one. The state metric of the state is updated with the selected value, and the survivor path information is recorded in the survivor path storage module.

State Metric Storage(SMS): The unit stores the partial path metric of each state at the current stage.

Survivor Path Storage(SPS): The unit records the survivor path of each state selected by the ACS module. It requires one bit of memory per state per stage to indicate whether the survivor path is the upper one or the lower one.

Output Block: This block, which incorporates combinational logic generates the decoded output sequence. During the traceback along the survivor path the decoded output sequence is latched to a register.

2.6.1 Trace Back Operation

The output block includes a trace back unit and a shift register unit. One iteration corresponds to one symbol time or decoding time. The traceback unit executes a trace back algorithm and reconstructs the data from the overall survivor path, which is the “lowest metric path”, among all the survivor paths remaining at the end of the decoding window. The survivors of each state at each iteration are the hypothesized information sequences, which are to be used in the “window” decoding process. At each node during the computations of state metric and path metrics not all the paths corresponding to the 4 states are kept in the memory but only those paths are kept which form

the most likely decodable path. One does not need to store the information sequence related to intermediate survivor paths but instead only store the results of each comparison. After $5K$ branches have been processed, these trellis connections are recalled in reverse order since at any given state, we can only continue backward on a path that survived upon entry into that node. Since each node has only one entering survivor, our traceback operation always yields a ML path. The important parameter that affects the decoding process is called traceback length (or path) or simply TBL. This path, which is traced back through the trellis diagram, is used to determine the decoded information bit. Decoding does not start until the iterations have reached window length L . When the iterations reach L , the first information bit is outputted as a decoded bit and sent out of the decoding window and a new bit takes the position at $L+1$ iteration, hence the saving of memory space. At the decoding time $(L+1)$, the second information bit is outputted and so on.

The lowest metric path is determined by the trace back unit through a comparison of the data path metrics stored in the SPS unit. The shift register unit executes a parallel transfer of data from the trace back unit after the information is decoded and outputs the data serially at symbol rate.

2.7 Summary

In this chapter the architecture for the convolutional encoder and Viterbi decoder is presented. The operation of both encoder and decoder is described in detail for constraint length 3. The trellis diagram for $K=3$, has 4 states, similarly $K = 7$ has 64 states, and $K = 9$ has 256 states. Hence as the constraint length increases the complexity increases exponentially and as a result the number of arithmetic operations, time for computing the path metrics, and storage area is also increased. The design at higher constraint length is more complex than at lower constraint lengths.

The implementation of higher constraint lengths of 7 and 9, are of interest for this thesis work. The constraint length $K=7$ Viterbi decoder is extensively used by NASA for satellite and deep space communication and this encoder is commonly referred to as the NASA encoder. The IMT 2000 standard for mobile communications recommends a constraint length 9 because of its good performance under low signal-to-noise ratio conditions. The $K=9$ implementation is also addressed in this thesis, in view of our interest in developing an architecture suitable for the IMT-2000 standard for WCDMA applications.

Viterbi Decoder Design and Implementation Issues

3.1 Introduction

Convolutional codes as described in chapter 2 are one of the major classes of error correction codes. The existence of a maximum likelihood decoding procedure called the Viterbi algorithm that can be implemented with reasonable complexity is the reason for their widespread use. In this chapter the various existing VLSI architectures and implementations of the Viterbi decoder are reviewed. The IMT 2000 specifications that impact the implementation are also discussed along with applications of the Viterbi decoder in mobile and satellite communications. Some of the recent recommendations for Viterbi decoder usage can be found in the IEEE 802.11a and 802.16c Wireless LAN Standard [Kosa02].

3.2 Literature Review

The decoding of each bit in a Viterbi decoder requires the execution of a large number of arithmetic operations. Typically for a constraint length equal to 9, the Viterbi decoder performs 512 add and 256 compare-select operations for every decoded bit. The unit, which performs this function, is referred to as the Add-Compare-Select (ACS) unit. The overall performance of the Viterbi decoder depends on the throughput of the ACS unit [Kivioja99, Jia98]. A number of different Vit-

erbi decoder implementations whose architecture, throughputs and areas are critically dependent on the design and configuration of the ACS unit have been reported in the literature [Shim99, Burger99, Cabrera97, and Fettweis91]. The architectures range from implementations based on a single ACS unit to fully parallel systolic designs that exploit the parallelism inherent in the Viterbi algorithm. Specifically, the reviewed papers belong to one of the following four categories or architectures:

- A) Single ACS unit based design
- B) Two/four ACS unit based design
- C) Parallel/systolic array based design
- D) Neural network based design

A) Single ACS unit based design

Shim and Suh [Shim99], have reported on a Viterbi decoding scheme which requires a reduced number of ACS operations and incorporates metric re-scaling and overflow detection. The overflow detection unit checks if an overflow occurs in any one of the path metrics and if so the path metric is rescaled by subtracting a fixed value from the contents of the path metric storage registers.

A synthesizable parameterized Viterbi decoder design reported by Burger et al. [Burger99], is based on a shared ACS unit. The single ACS unit is used to compute all 16 decoder states. The sharing of the ACS unit, however, dictates that the clock rate must be 16 times the data rate.

Other designs based on single ACS unit are presented in [Temerinac95, Xilinx, Qualcomm00, and Intel00].

B) Two/four ACS unit based design

The architecture reported by Cabrera et al. [Cabrera97], is composed of two processing elements (PEs) which compute all the states of the trellis. Each PE is composed of two ACS units and a FIFO. Each PE calculates and updates 4 path metrics at a time. The PE's FIFO units buffer the incoming symbols. The IC fabricated in a 0.7- μm CMOS technology occupies an area of 9 mm² and operates at a maximum clock frequency of 75 MHz.

In [Pandita99], Pandita and Roy have presented an implementation methodology which enables rapid prototyping based on high level synthesis targeted for Field Programmable Gate Arrays (FPGAs). Broadly speaking, the VLSI architecture implementations of the Viterbi algorithm can be divided into two categories:

- Node Parallel
- Node Serial

In the node parallel approach, as the computations for all the nodes in the trellis are done concurrently, the decoding speed is high. In the node serial approach, a single node does these computations sequentially and the decoding operation is much slower.

In [Pandita99], the Viterbi decoder implementation uses a node serial approach and two ACS pairs are utilized to process the data serially. The design has been divided into various blocks such as Branch metric, Add-Compare-Select (ACS), path metric memory and trace back blocks. Two pairs of ACS units are used for processing the 256 nodes in a constraint length 9 Viterbi decoder. The throughput of the design is limited to 19.2 Kbps.

Kong et al. in [Kong96] presented a fast serial Viterbi decoder Application Specific Integrated Circuit (ASIC) for a base station channel card for an IS-95 DS-CDMA system. The design

processes and stores all the path metrics by making use of four ACS pairs for processing the 256 nodes in a constraint length 9 Viterbi decoder with several internal SRAMs and one external 6K byte SRAM. The decoding depth or the traceback length (TBL) of the fast serial decoder is variable and depends on the coding gain. Depending on the application of the decoder the traceback length, which is usually fixed at 5 times the constraint length, could be programmed to achieve better coding gain. Another advantage of this design is reported to be the reduction of decoding time, hence faster decoding speed. The design operates at a maximum clock frequency of 30 MHz for a variable data rate ranging from 1.2 Kbps to 9.6 Kbps.

Fettweis et al. [Fettweis91], have achieved a very high throughput by exploiting the parallelism inherent in the Viterbi algorithm as well as the parallelism present at bit and word levels. For a 4 state decoder implementation, the IC fabricated in a 2.0 μm CMOS technology occupies an area of 74 mm² and operates at a maximum clock frequency of 29 MHz for a data rate of 115 Mbps.

C) Parallel/systolic array based design

Kivioja, Isoaho and Vanska have implemented complex DSP algorithms and telecommunication circuits in FPGAs [Kivioja99]. The authors have used multiple Xilinx FPGAs to implement a constraint length 7 Viterbi decoder. The maximally parallel implementation had 64 ACS units for processing 64 nodes and the resulting ACS block alone occupied two 4028-EX FPGAs, as a result the main problem was the transfer of data from one FPGA to another, especially as each ACS unit needed two FPGAs containing many feedback paths. The throughput of the design was reported to be 12 Kbps and operated at 1 MHz clock. Design criteria adopted for comparison are speed performance, routability and inter-chip communication. A more optimal solution would be to use

only one chip which would result in lower cost and power as well as a more compact implementation.

The systolic array based Viterbi decoder architecture reported by Truong et al. [Truong92], supports full parallel decoding of convolutional codes. The parallel decoding resulted in an implementation with small distributed memories. This in turn resulted in a small chip size and high throughput.

An other design reported in [Chang00] uses 256 ACS units in its implementation and uses a 80 MHz clock which results in significant power dissipation.

D) Neural network based design

Wang and Wicker [Wang96] presented a Viterbi decoder design based on the concept of neural networks. A neuron is used as a memory storage element at each node in the trellis structure of the Viterbi decoder. The fully parallel architecture is obtained by implementing most of the Viterbi algorithm using analog neurons as opposed to digital circuits. The authors claim, that the decoder is faster than comparable digital-only designs due to its parallel architecture. The decoder requires one-sixth the number of transistors required by a digital decoder. The trace back method called the Register Exchange Method (REM) is used for decoding information. In REM a shift register is associated with every state in the decoding trellis. Such a register contains information bits at a given time for a given state that are associated with the surviving partial path. During decoding the contents of the registers are updated and exchanged as surviving branches keep changing from state to state and from stage to stage. At the end of trellis the contents of the final register, corresponding to the surviving path, represent the decoded information bits. The decoder presented is designed for binary signaling with signal set (+1, -1) over Additive White Gaussian Noise (AWGN) channels.

The implementation of the adaptive neural network (ANN) decoder corresponds to an analog representation of a single trellis element. The authors have concluded that the Viterbi decoder can be implemented most effectively through the use of an analog neural network. The parallel architecture of neural networks is well suited for the Viterbi decoder. In the Viterbi algorithm, at each node the survivor path must be stored and this path is dynamic in nature meaning that it keeps “changing” at successive stages as the Hamming distance is calculated from stage to stage. Hence at each node the Hamming distance and the survivor path metric need to be stored and the authors use neurons at each node as storage elements. The authors use the REM for trace back purposes. In REM, the power consumption is greater as compared to the trace back algorithm. At every node in the trellis the register contents need to be updated as the bits decoded are shifted at every node and at every stage of the trellis. As a result the percentage of flip flops active per clock cycle in the REM is greater as compared to the percentage of flip flops active in the trace back algorithm. Hence the REM results in a greater power consumption.

The traceback algorithm presented in Chapter 2, consumes less power of the two schemes, and is better suited for power critical applications such as mobile and satellite communications.

Alston and Chau [Alston90] presented another neural network based Viterbi decoder architecture. The complexity of this architecture is a function of the constraint length i.e. as K increases the design becomes more complex in implementation and the complexity increases exponentially with respect to number of memory elements. The disadvantage with this neural network implementation is that the decoding algorithm is no longer maximum likelihood [Wang96], which may result in significant performance degradation. The implementation in [Alston90] is for the shorter constraint length of 3. Since the algorithm is not maximum likelihood, it is not clear how the neural network

based architectures would perform for constraint lengths of 7 and 9, where 64 and 256 states respectively need to be processed.

In summary, the internal architecture of the ACS unit affects its throughput, which in turn presents a performance constraint on the overall operation of the Viterbi decoder. The number of ACS units used in a particular design also affects the overall throughput of the Viterbi decoder. The ACS unit executes a large number of add and select operations. Typically, for case $K=9$, the decoder's ACS unit executes 23,000 additions and 11,520 select operations for every decoded bit. As shown in Table 3.1 different authors have carried out implementation for constraint lengths ranging from 3 to 9 and the number of ACS units in each design ranges from 1 to 256 depending upon the application and throughput requirements.

Table 3.1: Representative Viterbi decoder designs.

Referenced work	(K, R)	No. of ACS units	Data rate (Mbps)	Clock (MHz)	Gate Equivalent	Design/ Technology (CMOS)
Fettweis91	(3, NA)	4	115	29	8,800	ASIC/ 2.0- μ m
Truong92	(3, 1/2)	4	NA	NA	NA	NA
Altera01	(3, 1/2)	4	135	144	5000 ¹	FPGA
Cabrera97	(5, NA)	4	NA	75	11,646 [Lsi90]	ASIC/0.7- μ m standard cells library
Burger99	(5, 1/2)	2	NA	71.43	NA	ASIC/0.25- μ m standard cells library
Shim99	(7, 1/2)	1	>10	NA	NA	NA/simulations only
Kivioja99	(7, 1/2)	64	12 (Kbps)	1	18,866	FPGA/0.5- μ m
He00	(7, 1/2)	64	40	20	95,250 ²	ASIC/0.5- μ m
Temerinac95	7	1	2.5	60	4.7 (mm ²)	ASIC/0.8- μ m
Xilinx	(7, 1/2)	1	155	157	1346 Slices	FPGA
AHA Corp.	(7, 1/2)	1	62	NA	NA	ASIC
Qualcomm00	(7, 1/2)	1	25	NA	NA	ASIC
Intel00	(7, 1/2)	1	9	NA	NA	ASIC
Shakoor03	(7, 1/2)	1	2	46	122,796 ¹	Altera FPGA
Shakoor03	(7, 1/3)	1	2	42.8	151,860 ¹	Altera FPGA
Kong96	(9, 1/3)	4	9.6 (Kbps)	30	50,000	ASIC/0.8- μ m
Pandita99	(9, 1/2)	2	19.2 (Kbps)	NA	NA	FPGA/0.5- μ m
Chang00	(9, 1/3)	256	20	80	50,000 ³	ASIC/0.5- μ m
Kang97	(9, 1/2)	4	14.4 (Kbps)	0.9216	16,250 ³	ASIC/0.8- μ m
Tsui00	(9, NA)	4	NA	1.228	623,530	NA

Referenced work	(K, R)	No. of ACS units	Data rate (Mbps)	Clock (MHz)	Gate Equivalent	Design/ Technology (CMOS)
Shakoor03	(9, 1/2)	2	2	37	153,540 ¹	Altera FPGA
Shakoor03	(9, 1/3)	2	2	35.8	177,972 ¹	Altera FPGA

1: the gate count includes the memory implementation along with decoder logic. Conversion 12 gates per LE. [Altera AP Note 110]

2: 1 mm² contains 7.5 K gates for 0.55-μm [LSI Logic's ASIC Technology Roadmap]

3: converted to gates: 4 transistors per gate

3.3 IMT 2000 Technical Specifications for Viterbi Decoder

The parameters of the Viterbi decoder implementation for $K=9$ are summarized in Table 3.2. These parameters are in accordance with IMT 2000 technical standard for 3GPP and the WCDMA system [3Gpp01].

Table 3.2: Recommended technical specifications for WCDMA applications [3GPP01].

Parameter	Value
Constraint Length (K)	9
Code rates	1/2 (forward channel) and 1/3 (reverse channel)
Generator polynomials for $K=9$ and $R=1/2$	$G_0 = (561)_8$ and $G_1 = (753)_8$ $G_0(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8$ $G_1(x) = 1 + x^2 + x^3 + x^4 + x^8$
Generator polynomials for $K=9$ and $R=1/3$	$G_0 = (557)_8$, $G_1 = (663)_8$ and $G_2 = (711)_8$ $G_0(x) = 1 + x^2 + x^3 + x^5 + x^6 + x^7 + x^8$, $G_1(x) = 1 + x + x^3 + x^4 + x^7 + x^8$, $G_2(x) = 1 + x + x^2 + x^5 + x^8$
Data Rates	144 Kbps, 384 Kbps, upto 2 Mbps
Traceback length	64
Bandwidth(Current)	5 MHz, for WCDMA applications
Bandwidth(future-4GPP)	20 MHz
Fame by Frame encoding	Every frame of data is padded with $(K-1)$ number of tail bits for flushing the state of the encoder.

3.4 Viterbi Decoder Design Considerations

The requirements of digital communications systems for cost effective transmission with a desired bit error rate in a mobile communications environment promotes usage of convolutional coding and Viterbi decoding.

In section 3.2 various VLSI architectures have been introduced and the impact of the architecture on such parameters as gate count, throughput, scalability, power dissipation, and traceback length have been addressed where appropriate.

In this section we'll consider the Viterbi decoder performance, functionality, and architectural options and requirements in the context of the WCDMA standard's specifications as summarized in Table 3.2.

3.4.1 Add-Compare-Select Processor

The add-compare-select unit is the most critical part for the speed of the Viterbi decoder in terms of throughput. Its add-compare-select recursion limits the overall operation of the decoder.

Main issues in the ACS module design are:

- Throughput
- Number of ACS units
- Path metric rescaling or optimization
- Traceback length or window size
- Power and area

The speed-power, area-speed trade offs are considered in a number of the referenced papers, such as [Nikolic99, Kong96], and similarly the impact of traceback length and window size on area and for power are presented in others [Bustamantee89, McEliece89].

3.4.2 Throughput

Another very important aspect of decoding is the throughput of the ACS unit. In the implementation of the Viterbi decoder, the ACS unit executes a large number of add and select operations making it the most computationally intensive block in the design. As a result the throughput of the ACS unit is a limiting factor with respect to the overall decoder throughput. Typically for a $K=9$ Viterbi decoder there are 256 states in the trellis diagram. The Viterbi decoder performs 2 add operations and one select operation per each node of the trellis in one clock cycle. Thus the number of add and select operations corresponds to 512 and 256, respectively, per stage. The actual dimensionality of the trellis and the total number of add and select operations needed for decoding depend upon the size of the “decoding window”. The “decoding window” in a particular implementation corresponds to the number of received symbols that need to be processed to make a decision with respect to a current input. For a Viterbi decoder with constraint length $K=9$, a decoding window or a traceback length (TBL) of 45 symbols is recommended [Truong92, Clark88]. Thus for the $K=9$ case the decoder executes 23,000 additions and 11,520 select operations for every decoded bit. Similarly for the $K=7$ and $TBL=35$ case the decoder executes 4480 additions and 2250 select operations per decoded bit.

It has been shown that the traceback method decodes with high probability if TBL is equal to or greater than $5K$ [Lou95, McEliece89]. For decoders that satisfy this criterion the survivor paths converge within the decoding window. For the $K=9$ case the IMT 2000 standard stipulates a TBL of 64. Thus, for this case with its 256 state trellis, a decoding window greater than $5K$ is utilized to assure that all the paths converge to allow a correct decoding of the data [McEliece89].

3.4.3 Number of ACS Units

The Viterbi decoder implementation presented is based on using single ACS unit for constraint length $K=7$ and for code rates $1/2$ and $1/3$. The design has two ACS units in the case of constraint length 9 and code rates $1/2$ and $1/3$ where massive computational resources are required since there are 256 states to be processed. The number of ACS units in the ACS processor was kept to a minimum for each constraint length so that the overall gate count is minimized while maintaining the necessary throughput of 2 Mbps.

The architecture of the ACS and its operation is shown in Fig. 3.1. The path metrics computed for previous stage and branch metrics computed for the current symbol are added using 2 adders, and the 6 bit results are compared using a subtractor. Most significant bit of the difference controls the multiplexer and the new value of path metric is stored in the registers named new path metric. The decision bit which is either “0” or “1” is also stored as a part of the newly computed path metric.

Figure 3.1: ACS processor implementation based on a single ACS unit.

3.4.4 Path Metric Rescaling or Optimization

The path metric is the accumulated sum of branch metrics. Since the branch metrics are positive numbers, the path metric grows continuously, with each branch metric added. Therefore, their calculation requires some method of normalization. Several methods of metric normalization have been proposed in the past [Shim99, Nikolic99].

Shim et al. [Shim99] address the issue of quality of the metrics or word length of path metric registers and its optimization during the decoding. The issue of path metric optimization is important and substantial savings in terms of design area can be achieved through judicious optimization. The word length of the path metric registers in the ACS unit has a direct impact on its area. Synthesis of the Viterbi decoder design with 8 bit path metric registers resulted in a 16,217 logic cell circuit. To reduce the area of the decoder it is desirable to decrease the dimensionality of the path metric registers. However, a reduction in register size from 8 bit to 6 bits resulted in overflow problems and subsequent decoding errors.

The path metric register size was determined based on the following criteria:

- Error free decoding,
- No overflow,
- minimum area.

To circumvent the problem of overflow a metric rescaling technique was adopted [Shim99]. For this implementation, with 6 bit path metric registers, the decoder required 10,233 logic cells when mapped to Altera's APEX EP20K400E-3 device.

3.4.5 Traceback Length or Window Size

Another Viterbi decoder parameter is traceback length (TBL). Choosing the right traceback length is extremely important in Viterbi decoding. We have observed in our simulations that for example for $K=7$ a shift from a TBL of 35 to 50 results in a marginal improvement in BER. For $K=9$ and rate $1/2$ the graphs show a similar kind of improvement but with a low SNR. The graphs for different values of TBL of 35, 45, 50, 55 and 64 are presented in the Chapter 4.

To achieve the objective of developing a scalable, programmable and high throughput design which is area efficient and which meets the IMT-2000 standard requirement a single ACS unit (for $K=7$) based Viterbi decoder was implemented while the 2 ACS unit based Viterbi decoder was implemented for $K=9$, and also to address the requirements of wireless and satellite communications and potential system on a chip implementation the decoder design was realized as a reconfigurable VHDL macrocell.

3.5 Viterbi Decoder Parameters

The parameters that influence the overall area of the decoder are the constraint length, code rate, window size and the path metric registers inside the ACS unit. In satellite applications generally a $K=7$ Viterbi decoder provides satisfactory coding gain under typical SNR conditions whereas in mobile communications we need to implement a $K=9$ Viterbi decoder to get the acceptable coding gain.

Table 3.3: Technical specifications for the Viterbi decoder implementations.

Design Parameters	Value
Constraint Length (K)	3, 7 and 9
Code rate	1/2 and 1/3
Generator polynomials for K=3	$G_0 = (7)_8$ and $G_1 = (5)_8$ $G_0 = 1 + x + x^2$, $G_1 = 1 + x^2$
Generator polynomials for K=7, R=1/2	$G_0 = (171)_8$ and $G_1 = (133)_8$ $G_0 = 1 + x + x^2 + x^3 + x^6$, $G_1 = 1 + x^2 + x^3 + x^5 + x^6$
Generator polynomials for K=7, R=1/3	$G_0 = (171)_8$, $G_1 = (133)_8$ and $G_2 = (145)_8$ $G_0 = 1 + x + x^2 + x^3 + x^6$, $G_1 = 1 + x^2 + x^3 + x^5 + x^6$, and $G_2 = 1 + x + x^4 + x^6$
Generator polynomials for K=9, R=1/2	$G_0 = (561)_8$ and $G_1 = (753)_8$ $G_0 = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8$, $G_1 = 1 + x^2 + x^3 + x^4 + x^8$
Generator polynomials for K=9, R=1/3	$G_0 = (557)_8$, $G_1 = (663)_8$ and $G_2 = (711)_8$ $G_0 = 1 + x^2 + x^3 + x^5 + x^6 + x^7 + x^8$, $G_1 = 1 + x + x^3 + x^4 + x^7 + x^8$, and $G_2 = 1 + x + x^2 + x^5 + x^8$
Data Rates	64 Kbps, 144 Kbps, 384 Kbps, and 2 Mbps
Clock	37 MHz for K = 9 at TBL = 45
Fame by Frame encoding	Every frame of data is padded with (K-1) number of tail bits for flushing the state of the encoder.
Path-metric Optimization	6 bits

3.6 Viterbi Decoder Applications in Communications

3.6.1 Applications to Satellite Communications

The Viterbi decoding is used in geostationary satellite communications in conjunction with Reed-Solomon coding (RS codes). The combination of VD and RS codes is also referred to as concatenated coding system [Lin83, Peterson95]. Figure 3.2 shows this kind of implementation for a receiver station. Furthermore, the VSAT (Very Small Aperture Terminal) modems and the Direct Broadcast satellite systems also use Viterbi decoders in their implementations.

3.6.2 Applications to Mobile Communications

In mobile communications, the available power and bandwidth are very limited. The transmitted signal is subject to multipath fading, and the size of the mobile receiver is also restricted. Recently introduced mobile communication systems make extensive use of error-control coding techniques (such as Viterbi decoding and Turbo codes) and cellular concepts. The concept of cellular mobile radio makes maximum use of the channel bandwidth by reusing the available frequencies in different cells and is well suited for the growing demand of efficient digital mobile transmission on bandlimited channels. As a result, cellular radio systems suffer from significant co-channel interference as well as adjacent channel interference and fading. Good error correcting codes such as convolutional codes with higher constraint length ($K=9$) and Turbo codes are recommended in the IMT-2000 standard [Lee97, Nikolic99].

Figure 3.2: Application of Viterbi decoder in concatenated coding system.

3.6.3 Other Applications of Viterbi Decoders

Presented below are some more applications of the Viterbi decoders:

- Digital Communication [Sklar88]
- Wired/wireless communication [Wilson96, Zaragoza02]

- Voice-Band Data Communications [Lee97]
- HDTV

In mass storage media

- Hard disk and Magnetic Recording Channel [Nikolic99]
- CD-ROM
- DVD

In general the following advantages and costs are associated with coding.

- Accuracy of received information is improved.
- Less energy is required for transmission in coded system when compared to uncoded system for the same BER.
- Cost associated is increased system complexity.

3.7 Summary

A literature review was presented and applications of the Viterbi decoder were discussed in this chapter. The features of the Viterbi decoder which make it so attractive is its constant decoding time and MLSE (Maximum Likelihood Sequence Estimator) optimality. The parameters that influence the overall area of the decoder were presented. The convolutional encoding and Viterbi decoding is extensively used in satellite, mobile communications, digital communications, HDTV, CD-ROM, and DVD. The Viterbi decoding is recommended in IMT 2000 standards which come under the umbrella of ITU. In the next chapter the results of both hardware and software implementation of the Viterbi decoder are presented.

Verification, Hardware Synthesis, FPGA Implementation and Functional Modeling Results

4.1 Introduction

The Viterbi decoder described in Chapter 3 was implemented both in hardware and software. The hardware implementation was done using VHDL and synthesis and simulations were carried out using Mentor Graphic's Leonardo Spectrum and Modelsim CAD tools respectively. The software implementation was carried out using Matlab's simulation tools. The test results of the hardware and software implementations are presented in this chapter. The channel model utilized is described. The remainder of this chapter includes graphical representations of the results along with the test bench setup, which was used for hardware testing.

4.2 Channel Model

The channel used in the implementation corrupts the signal by the addition of additive white Gaussian noise (AWGN). At the receiving end, the signal is demodulated and decoded using the convolutional decoder. To model in matlab the AWGN channel the built-in-function *randn* was

used, which generates a random binary sequence that has a normal distribution with mean 0 and variance $N_0/2=1$, where N_0 is the two-sided noise spectral density.

The relationship between the noise power per symbol ($npow$) at the receiver and the energy per bit E_b is given by the following expression:

$$npow = \frac{spow}{b_r} \cdot \frac{S_r}{\frac{E_b}{N_o}} \quad 4.1$$

where,

b_r : is the bit rate,

S_r : is the symbol rate,

$spow$: is the signal power per symbol, and

$npow$: is the noise power per symbol.

Energy per bit “ E_b ” and noise power density N_o are defined:

$$E_b = \frac{spow}{b_r} \quad 4.2$$

$$N_o = \frac{npow}{S_r} \quad 4.3$$

After combining Equations 4.1 and 4.2, we get

$$\frac{E_b}{N_o} = \frac{spow}{b_r} \cdot \frac{S_r}{npow} \quad 4.4$$

The noise power, $npow$ is calculated from Equation 4.1. From Equation 4.1, it can be seen that at low values of E_b/N_o the effect of noise on data in the channel can result in high BER at the receiver. In contrast at high E_b/N_o the effect of noise on data is less dominant and hence the BER is low at the receiver.

4.3 Codec Link Model

Referring to Fig. 4.1, the information bits received from the data source are convolutionally encoded and then modulated using BPSK before transmission. At the receiver, the received signal is contaminated by AWGN before its arrival. The received signal is then demodulated and decoded. In the power level detector, the signal power level is detected and depending on this signal value, the noise level is adjusted and accordingly the noise is inserted into the channel. In this way simulations are controlled by varying just the ratio of E_b/N_o and performance curves are plotted as shown in section 4.7.

Figure 4.1: Functional model for proposed convolutional encoder and Viterbi decoder link.

4.4 Overall Design Flow

The design of the Viterbi decoder was initiated with the development of a functional model in Matlab. The design was simulated for various TBL's of 35, 40, and 50 in case of $K=7$, and for TBLs of 45, 55, and 64 in case of $K=9$ by judicious application of test vectors to its input. The functional simulation results obtained are presented in section 4.7.

4.4.1 Altera APEX EP20K400 Device Features

The implementation of the Viterbi decoder was done on the APEX 20K400 device. The basic features of this device are listed in Table 4.1.

TABLE 4.1: Altera APEX EP20K400 device features.

Features	EP20K
Typical gates	400,000
Logic elements	16,640
Maximum I/O pins	502
Package used	EBC652
Speed grade	3
CMOS process	N/A

4.4.2 FPGA Design Cycle

The design process for using any kind of FPGA's requires the following steps:

1. Entering the design in the form of Hardware description languages called VHDL,
2. Simulating the design for functional verification,
3. Synthesizing the design into the FPGA architecture,
4. Placing and routing the FPGA design,
5. Timing analysis,
6. Generating the FPGA device configuration/programming file (bit file).
7. Testing the circuit's functional behavior.

4.4.3 Design Methodology

Below is a brief description of the design methodology adopted in this work.

Development of VHDL code was initiated in conjunction with functional modelling based on Systemview (CAD tool by Elanix Inc.,) and its library for communication system design. Simulations were carried out iteratively to study the effects of change in three Viterbi decoder parameters, namely constraint length, code rate, and traceback length.

In the next step, a functional model of the Viterbi decoder was developed in Matlab. The Matlab realization corresponded to the architecture and VHDL implementation of the circuit. Results were compared with Systemview's performance, and were used to guide the completion of the VHDL code for a Viterbi algorithm realization that supports different constraint lengths and code rates. This completed the step 1 in the design process.

At step 2 the Modelsim simulation tool, the Industry standard CAD tool supplied by Modelsim Technologies was used to verify the functionality of the VHDL code.

At step 3 the Mentor Graphics CAD tool called the Leonardo Spectrum synthesis tool was used to synthesize the design at the behavioral level. Leonardo spectrum is one of the powerful synthesis tools used Industry wide for FPGA design purposes. It takes in VHDL design files and synthesizes the code and generates a file in a special format called Electronic Data Format (EDF). This generated EDF file contains the full design information and is a gate level representation of the design. At this stage design constraints such as frequency of operation, speed or area based optimization criteria were investigated and selected. At the end of the synthesis process a gate level circuit was obtained.

In steps 4 to 6, the EDF file generated in step 3 is imported into Altera's Quartus-II design environment in which translation, mapping of the design into Altera's FPGA technology and the design was placed and routed after which timing report was generated and finally a bitfile is generated.

Step seven involved actual circuit testing with digital IC test equipment. The Viterbi decoder was tested by means of an IMS-100 tester that operates at clock rates of up to 100 MHz. The hardware testing methodology is presented in section 4.6.

4.5 Viterbi Decoder Implementation Results

Effective implementation of the Viterbi decoder dictated that we investigate ways to reduce the complexity of the algorithm and its circuit implementation. A new design technique for reducing complexity is presented in this section. Furthermore, the Viterbi decoder architecture's scalability is presented in this section. The implementation results of the Viterbi decoder for different constraint lengths and different code rates are also presented in this section.

4.5.1 Complexity

For Viterbi algorithm implementation at each time step $2^{(m+1)}$ distance calculations and 2^m comparison operations are required.

For example for a $K = 9$ Viterbi decoder,

$2^{(m+1)} = 512$ distance calculations and

$2^m = 256$ comparisons are required per stage.

And for a trace back length T the number of distance calculations and comparisons can be expressed as $2^{(m+1)} T$ and $2^m T$ respectively.

Also each distance calculation requires “ n ” binary comparisons for a rate $1/n$ system. Hence the number of addition and comparison operations are given by Y_a and Y_c

$$Y_a = n T 2^{(m+1)}, \quad 4.5$$

$$Y_c = n T 2^m, \quad 4.6$$

For every information bit encoded at the receiver the four possible symbols which could be transmitted are 00, 10, 01, 11 and these symbols are treated as expected symbols at the receiver. Suppose “00” is transmitted then there is a possibility that it could be received as “00”(no error) or “10” or “01” for single error or it could be received as “11” resulting in double error. Similarly if “10”, “01”, or “11” is transmitted then four possible symbols could be received.

A considerable number of computations is needed if distance computations take place at each node of each stage, this is especially true for a 256 states decoder. In order to reduce this complexity the design was implemented in the following manner.

All four possible Hamming distance (HD) are pre-coded in hardware so that only a 2 bit comparison for four symbols is required per stage. When a symbol is received from the channel say “00”, then the possible Hamming distances after comparison to expected symbol “00” are:

$$HD<00, 00> = 0,$$

$$HD<00, 01> = 1,$$

$$HD<00, 10> = 1, \text{ and}$$

$$HD<00, 11> = 2$$

These Hamming distances are used to process each stage of the trellis and when the next symbol is received just a 2 bit comparison will yield the four Hamming distances and so on. This reduces the need for comparing and generating at every node Hamming distances to be used for computing path metrics at all 256 states of the particular trellis stage.

Let us now consider the resulting savings in terms of the reduction in operations or complexity.

For example at TBL = 64 and for K=9, after substitution the complexity equation 4.1 becomes:

$$Y_c = n T 2^m = 2 \times 64 \times 256 \quad 4.7$$

$$Y_c = 512 \times 64$$

after applying the above mentioned technique in which only 4 possible symbols are compared at the beginning of the processing of each state. Hence the need to make 512 bit comparisons per stage is eliminated and reduced by the following factor:

$$512/4 = 128 \text{ and } Y_c = 128 \times 64$$

As another example let us consider the case where TBL = 50 and K=7 for which the complexity equation becomes:

$$Y_c = n T 2^m = 2 \times 50 \times 64$$

$$Y_c = 128 \times 50 \quad 4.8$$

After applying the above mentioned technique only 4 possible symbols are compared at the beginning of the processing of each state. Hence the need to make 128 bit comparisons per stage is eliminated and the number of comparisons is reduced by the following factor:

$$128/4 = 32 \text{ and } Y_c = 32 \times 50$$

4.5.2 Scalability and Programmability

Developing a scalable architecture was one of the motivations behind this research. Scalability of a Viterbi decoder architecture can be considered in terms of three different parameters. First, a Viterbi decoder architecture can be scalable with respect to its code rate ($1/n$), meaning that the user can easily scale the code-rate as per the application requirements and availability of the resources in the targeted FPGA. Second, a Viterbi decoder circuit can be scalable with respect to its constraint length. Different constraint lengths of 3, 7, and 9 may be required. Third, a user can scale the decoder with respect to traceback length. The published literature [McEliece89] stipulates that with increased traceback length the performance of the decoder increases. The higher the TBL the higher the performance. By theory a $5K = 45$ [Lou95, McEliece89] for $K=9$ Viterbi decoder should be adequate, however, the IMT-2000 standards for $K=9$ rate $1/2$ and $1/3$ decoders suggest a TBL of 64 to improve performance. Programmability is introduced in the architecture to enable the user to select the required TBL for a particular application. For $K=7$ the possible TBLs which could be programmed are 35, 40, and 50 for code-rate $1/2$ and $1/3$. While for $K=9$ the possible TBLs are 45, 55, and 64 for both code rates.

4.5.3 Synthesis Results for Constraint length $K=7$ and $K= 9$ and for Code rates of $1/2$ and $1/3$

The parameterized implementation of the Viterbi decoder permits the selection of the constraint length, code rate and decoding window dimension. The constraint lengths could be 7 or 9 depending on the type of application with $K=7$ commonly used in satellite communications [Bustamante89] and $K= 9$ in wireless applications [3Gpp01]. In addition to that, the code rate can also be varied and set to a value of $1/2$ or $1/3$. The affects of increased window size for fixed con-

straint lengths of $K=7$ and 9 decoders are presented in Tables 4.2 and 4.3, which also depict the device utilization and the operating frequency for the implementation of the Viterbi decoder for $K=7$ and $K=9$ on Altera's APEX EP20K400E-3 devices. The throughput and area of the decoder are dependent on the window size. The minimum frequency of operation to assure a throughput requirement of 2 Mbps, for $R=1/2$ is 46 MHz with $TBL=35$ while for the same TBL but for $R=1/3$ and the same throughput the minimum frequency of operation is 42.8 MHz. The decoder was successfully simulated with a master clock operating at 100 MHz.

TABLE 4.2: Logic Cell (LC) utilization based on a mapping onto Altera APEX EP20K400E-3 device for data rate of 2 Mbps for constraint length 7, code rates (R) $1/2$ and $1/3$, and Traceback lengths (TBL) of 35, 40 and 50.

TBL	$R=1/2$ LCs	Clock (MHz)	$R=1/3$ LCs	Clock (MHz)
35	10,233	46.0	12,655	42.8
40	10,606	46.8	13,209	43.7
50	11,260	47.7	14,090	44.5

Hardware implications of rate $1/2$ and rate $1/3$ decoders:

For rate $1/2$ systems every information bit is coded into 2 outgoing bits as a result we have 100% redundancy being added to original information bits. Here more bandwidth will be required to transmit a coded system than an uncoded system. For rate $1/3$ systems every information bit is coded into 3 outgoing (coded) information bits as a result the transmitted signal occupies more transmission bandwidth than the one in rate $1/2$ systems. Also in this case the redundancy is 33% more than in rate $1/2$ systems.

The rate 1/2 decoder receives 2 bit symbols and rate 1/3 decoder receives 3 bit symbols. The rate 1/2 decoder requires fewer hardware resources than the rate 1/3 decoder. However the rate 1/3 decoder provides higher protection against noise as compared to the rate 1/2 decoder, since extra redundancy is added at the transmitter for the rate 1/3 case and this is done at the cost of increased complexity and more resources. It is evident from Table 4.2 that for $K=7$ that the number of LCs at $TBL = 50$ for the rate 1/2 decoder is 11,260 and for the rate 1/3 decoder the LCs count is 14,090.

TABLE 4.3: Logic Cell (LC) utilization based on a mapping onto Altera APEX EP20K400E-3 device for data rate of 2 Mbps for constraint length 9, code rates (R) 1/2 and 1/3; and Traceback lengths (TBL) of 45, 55 and 64.

TBL	R=1/2 LCs	Clock (MHz)	R=1/3 LCs	Clock (MHz)
45	12,795	37.0	14,831	35.8
55	12,844	38.3	15,428	36.9
64	13,582	38.4	16,024	37.4

Constraint length 7 decoder:

For constraint length 7 Viterbi decoders, there are possible 64 states in the decoding trellis. Thus a total of 128 path metrics are computed at any given time, out of which only 64 path metrics are retained at every stage of the trellis. In conjunction with the path metric storage at every node of the trellis there is also either a “0” or “1” stored as a decision bit per state.

For $K=7$ decoder the memory required for a TBL of 35, for a path metric size (W) of 6 bits wide, with 64 states is:

$$\begin{aligned} \text{memory size} &= 2^{(K-1)} \text{ W TBL} \\ \text{memory size} &= 64 \times 6 \times 35 = 13,440 \text{ bits (13K)}. \end{aligned} \quad 4.9$$

Referring to Equation 4.10, it can be said that memory size in a circuit is a function of the number of states in the design, the path metric register width, and the traceback length.

A distributed memory mapping is used in the FPGA. The distributed memory is one in which logic cells are configured as memory blocks. The higher the memory requirement the greater the number of logic cells utilized by the synthesizer during the mapping of the circuit onto a FPGA. Hence in this case 13 Kbits of memory is mapped on an FPGA and as a result the total number of LC's occupied by the full circuit is 10,233 with a frequency of operation of 46 MHz.

Constraint length 9 decoder:

In a larger system namely one with $K=9$, there 256 states in the decoding trellis. Thus there are 192 states more as compared to $K=7$ which has 64 states. For this case a total of 512 path metrics are computed at any given time, out of which only 256 path metrics are retained at every stage of the trellis. With the path metric storage at every node of the trellis there is also either a "0" or "1" stored as a decision bit per state. In order to handle the increased number of states 2 ACS units are used to process the 256 states per stage.

A traceback length for the decoder design of 5K has been suggested [Lou95]. Hence the traceback length is 45 for $K=9$. This implies increased memory requirements for TBL of 45 as compared to TBL of 35 for $K=7$. This consequently results in a bigger design for $K=9$ as compared to one with $K=7$.

Therefore, the increased number of states, the multiple ACS units and increased TBL of 45 consequently results in a bigger circuit design.

For example for a $K=9$ decoder with $TBL=45$, and a path metric size of 6 bits wide, the memory required corresponds to:

$$\begin{aligned} \text{memory size} &= 2^{(K-1)} \times W \times TBL \\ \text{memory size} &= 256 \times 6 \times 45 = 69,120 \text{ bits (69K)}. \end{aligned} \quad 4.10$$

From the above equation it is clear that the number of memory bits in a circuit is a function of the number of states in the design, the path metric register width, and traceback length. If any one of these parameters is increased it results in an increase in FPGA resource requirements in terms of logic cells and memory.

Hence in this case 69K bits of memory must be mapped on an FPGA. The total number of LCs required to map a $K=9$ circuit onto a FPGA corresponds to 12,795 for a frequency of operation of 37 MHz.

A comparison of the resource allocation requirements as a function of the different design parameters is presented in Tables 4.2 and 4.3.

Hardware implications of decoders with different constraint lengths and traceback length:

The decoder's master clock frequency is another parameter which is affected by the increased data rate. For a $K=7$ decoder the frequency of operation is 46 MHz with 10,233 LCs for a memory requirement of 13 Kbits. For a $K=9$ decoder the frequency of operation is 37 MHz with 12,795 LCs with a memory requirement of 69 K bits.

The complexity of the decoder with $K=9$ is much higher than that of one for which $K=7$. The design with $K=9$ consumes more clock cycles because here more memory is needed, greater number of ACS operations is required, due to the greater TBL, number of states, and greater num-

ber of times that the memory is accessed as compared to the $K=7$ case. Even decoders with different TBL's have been found to have different resource requirements during the synthesis process. As depicted in Table 4.2 and 4.3, the TBL's affect the overall LC count in the design. Decoders with TBL=64 are recommended in the IMT-2000 standard [3Gpp01]. And simulation results confirm that this configuration provides the best performance as compared to the other decoder realizations.

4.6 Testing Methodology and Test Results

As depicted in Fig. 4.2, the testing of the Viterbi decoder was done by means of a VHDL test bench with encoder, decoder and error generator hardware modules instantiated in the test bench. The objective of the functional test is to verify the data integrity of the signal after the encoding/decoding process in the presence of additive white Gaussian noise. Referring to Fig. 4.2 the input stimulus generated in the signal source block is directed to the encoder block where the data is convolutionally encoded for $K=7$ and 9, and afterwards the channel errors were inserted randomly in the encoded data inside the error generator module. The corrupted input data for various ratios of the average bit energy (E_b) to N_o is fed to the Viterbi decoder and the decoded output bit stream

is compared to the original input data stream to determine the bit error rate.

Figure 4.2: Block diagram of test bench incorporating signal source and data integrity comparator, encoder, decoder and error generator module.

As shown in Fig. 4.3 the test methodology for computation of the BER, involves the addition of redundancy to data by means of convolutional coding at the transmitting end and simultaneously performing detection and correction of errors on the received data stream at the receiver

side. The information bits are routed to the convolutional encoder realized in matlab functional model and the same set of information bits are input to the encoder designed in VHDL. Errors are generated in the error generation module and a decoder in matlab as well as the hardware (VHDL) decoder process the data independently and feed it to the BER computational module to generate the BER v/s SNR curves as shown in Figures 4.5 and 4.6.

Figure 4.3: Block diagram of functional and hardware encoder and decoder configurations for the determination of bit error rates in the presence of AWGN.

Hardware Testing

To test the Viterbi decoder use was made of the IMS-100 tester in conjunction with CRC designed and fabricated adapter board depicted in Fig. 4.4, which contains the Altera EP20K1500

device. After the completion of the synthesis process the Viterbi decoder with constraint length $K=7$ required 27% of the available LE's on the EP20K1500 FPGA. Prior to hardware testing the bitfile generated by the Place and Route tool Quartus is loaded onto a PC connected to the IMS tester. Next, the FPGA board is connected to this PC and the bitfile is loaded onto the FPGA. The Viterbi decoder mapped onto the FPGA is exercised by means of a set of test vectors. The test vectors consisted of convolutionally encoded data with additive white Gaussian noise. The output bits are compared with the known vector outputs and the results are displayed on the PC monitor. The decoder circuit was tested for $R=1/2$ and $1/3$ and TBL's of 35, 40 and 50. The decoder was fully operational at the 2 Mbps for all configurations.

Figure 4.4: CRC developed adaptor board for the IMS-100 tester with an Altera EP20K1500 FPGA.

4.7 Simulation Results

The functional simulation results provided in this section are based on the following set of Viterbi decoder parameters:

- Data Rate: 2 Mbps (recommended in IMT-2000 standards for 3G/WCDMA applications for K=9)
- Code rate: 1/2
- Channel: Additive white Gaussian noise (AWGN)
- Traceback lengths (for K=7): 35, 40, 50
- Traceback length (for K=9) : 45, 55, 64
- Functional simulator: Matlab simulator

The IMT-2000 standards require 10^{-3} BER for voice applications. The performance measurements determined with the setup in Fig. 4.2 has shown that for a $E_b/N_0 = 6.8$ dB the BER is better than 10^{-4} .

4.7.1 Constraint Length K=7 and Code Rate 1/2

The functional modeling of the parameterized Viterbi decoder implementation was accomplished in the Matlab simulation environment. The simulation results in Fig. 4.5 show the error performance of the encoder and decoder link in terms of E_b/N_0 , the ratio of energy per bit of information to noise power spectral density (N_0). The E_b/N_0 is in decibels (dB) and the BER is plotted on a logarithmic scale covering four decades from 10^{-1} to 10^{-4} . Figure 4.5 displays the three performance curves for TBL (or window size) corresponding to values of 35, 40 and 50. The BER performance is dependent, if only modestly, on the window size. For TBL of 35 which is 5K as rec-

ommended in the literature we get the dashed line curve in Fig. 4.5. The performance for a

Figure 4.5: Performance of Viterbi decoder for K=7, at code rate (R)=1/2.

TBL = 35 was found to be lower than the performance of with TBL of 50. The performance gain for this TBL value as compared to that of TBL = 35 is approximately 0.4 dB. The longer TBL gives extra time for the decoding algorithm to converge before making the decision during the traceback operation. The gain of 0.4 dB is significant when each tenth of a dB represents a significant reduction in the cost of a communication system [Barbulescu96]. The functional models and simulations of the decoder were implemented only for the rate 1/2 case. However, both code rates 1/2 and 1/3 for K=7 and 9 were implemented in VHDL.

4.7.2 Constraint Length $K=9$ and Code Rate $=1/2$

Figure 4.6: Performance of Viterbi decoder for $K=9$ at code rate $(R)=1/2$.

The matlab design and simulations were also done for constraint length 9, code rate $1/2$ in the same way as for $K=7$. Figure 4.6 displays the three performance curves for constraint length 9 and for TBL corresponding to respectively 45, 55 and 64. The best performance is provided by the TBL=64 case, which represents a performance gain of 0.3 dB over the TBL=45 decoder realization. The total number of bits transmitted for $K=7$ and 9 to determine each point on the graph was 1 million at a bit rate of 2 Mbps.

4.8 Summary

This chapter discusses in detail the implementation and the functional, behavioral and hardware test results of the Viterbi decoder for constraint length 7 and 9 for code rates of 1/2 and 1/3. The design is parameterizable and can be configured for various traceback lengths. By way of increasing TBL from 35 to 50 ($K=7$) and 45 to 64 ($K=9$) it was possible to lower the E_b/N_o needed to achieve the desired BER. Synthesis results for each implementation were presented. The circuit is optimized for 6 bit representation of the internal registers of the ACS unit yielding an area efficient design.

This chapter provided an overview of the FPGA design steps with device features for the Altera Apex EP20K400 FPGA utilized to synthesize the circuits.

The results of the Leonardo synthesis tool from Modelsim Technology indicate that the Viterbi decoder implementations support a throughput of 2 Mbps in each case but the frequency of operation depends on the TBL and the code rate. The hardware functional verification shows equivalence to simulation thus verifies the device correct operation.

Higher throughputs can be achieved by utilizing a greater number of ACS units. However, such increases in throughput can only be achieved at the cost of higher gate counts and possibly would require a larger FPGA device. From the results it can be deduced that it is possible to implement different constraint length designs with different traceback lengths and still have the same throughput performance.

Summary and Future Work

5.1 Summary

In this thesis we have presented the circuit and VHDL implementation of a parameterized and reconfigurable Viterbi decoder macrocell. The parameters which can be configured are the constraint length (7 and 9), code rate (1/2 and 1/3) and the decoding window dimension (35, 40, 45, 50, 55 and 64). In this context it should be mentioned that to our knowledge this is the only decoder realization that permits a dynamic reconfiguration of the decoding window depth. Such a feature could be effectively exploited in a cognitive radio whose characteristics are adapted to optimize performance for given channel conditions. The Viterbi decoder implemented in this work is area and power efficient. A key contribution has been the development of a modular decoder architecture that permits customization of design to address throughput requirements while minimizing the gate count and power. This is achieved by matching the parallel processing capabilities of the ACS processor with the decoder's throughput requirements. The circuit design has been targeted for an FPGA implementation. The design methodology employed is not specific to just the Viterbi decoder but can be used for the hardware implementation of any other Digital Signal Processing algorithm. The circuit simulation results were cross verified with simulations based on a functional

model of the Viterbi decoder reflecting the hardware architecture in the Matlab environment. The functional simulations in the Matlab environment have closely matched the hardware implementation. The choice of traceback length or window size affects the BER performance. Performance graphs based on the functional models have shown a coding gain of 0.4 dB in moving from a TBL of 35 to one of 50. The designs were mapped onto the Altera APEX EP20K400EBC-652-3 devices. Simulation results confirm that the FPGA implementations of the decoder circuits support a throughput of 2 Mbps. Path metric optimization has resulted in area savings of approximately 37% [Shakoor01, Shakoor03, Shakoor04]. The integration of the convolution encoder, random pattern generator, Viterbi decoder and comparator in a single macrocell provides a unique capability for self-test. This of course would facilitate the development of a test strategy for any macrocell incorporating this Viterbi decoder.

The specific contributions can be summarized as follows:

- A scalable architecture for the Viterbi decoder is demonstrated for constraint lengths 3, 7 and 9 and for code rates $1/2$ and $1/3$.
- Modular ACS architecture with one or more arithmetic computational units that matches throughput requirements.
- A new technique for determining Hamming distance calculations requiring a reduced number of comparison operations.
- An architectural realization of the Viterbi decoder that supports on-line trace-back length reconfigurability without degradation in throughput.
- Demonstrated word length normalization techniques for Viterbi decoders with constraint length $K=9$.

5.2 Suggested Future Research

Broadband communications in the future is expected to support data throughputs in the hundreds of mega bits per second. Furthermore, cognitive software defined radio will have to be adaptive and capable of determining both the channel conditions and possible optimal encoding and decoding requirements.

In view of the above it seems that there is an ample opportunity to carry out research on:

- Decoder architectures capable of operating in conjunction with a broad range of convolutional encoders, code rates and constraint lengths.
- High performance, low power decoder realizations capable of processing data with throughputs in the hundreds of mega bits per second.

Bibliography

- [Adamek91] J. Adamek, *Foundations of Coding: Theory and Applications of Error-Correcting Codes, With An Introduction to Cryptography and Information Theory*, John Wiley, Toronto, 1991.
- [AHA] *Viterbi with Reed-Solomon Decoder*, Product Specification, Product listing # AHA4210, AHA Corporation.
- [Alston90] M.D. Alston and P.M. Chau, "A Neural Network Architecture for the Decoding of Long Constraint Length Convolutional Codes," Proc. of the International Joint Conference on Neural Networks, pp. 121-126, June 1990.
- [Altera01] *Implementing an IEEE Std. 802.16-Compliant FEC Decoder*, White Paper, Altera, 2001.
- [Bae99] S. J. Bae, G. Kang, H. S. Lee, and E. K. Joo, "A Low Power Implementation Scheme of Viterbi Decoder for CDMA Mobile Communication Systems," IEEE International Symposium on Intelligent Signal Processing and Communication Systems, pp. 315-318, December 1999.
- [Barbulescu96] S.A. Barbulescu, "Iterative Decoding of Turbo Codes and Other Concatenated Codes," Ph.D. Dissertation, School of Electronic Engineering, University of South Australia, February 1996.
- [Blahut03] R.E. Blahut, *Algebraic Codes for Data Transmission*, Cambridge University Press, Cambridge, 2003.
- [Brown00] S. D. Brown and Z. G. Vranesic, *Fundamentals of Digital Logic with VHDL Design*, McGraw Hill, Boston, 2000.
- [Burger99] R. Burger, G. Cesana, M. Paolini, M. Turolla, and S. Vercelli, "A Fully Synthesizable Parameterized Viterbi Decoder," Proc. of IEEE Custom Integrated Circuits Conference, pp. 27-30, May 1999.

- [Bustamante89] H.A. Bustamante, I. Kang, C. Nguyen, and R.E. Peile, "Stanford Telecom VLSI Design of a Convolutional Decoder," Proc. of IEEE Military Communications Conference (MILCOM), vol.1, pp. 171-178, October 1989.
- [Cabrera97] C. Cabrera, M. Boo, and J. D. Bruguera, "VLSI Implementation Of An Area-Efficient Architecture For The Viterbi Algorithm," Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 1, pp. 623-626, April 1997.
- [Chang00] Y.-N. Chang, H. Suzuki, and K. K. Parhi, "A 2-Mbps 256 State 10-mW Rate 1/3 Viterbi Decoder," IEEE Journal of Solid State Circuits, vol. 35, no. 6, pp. 826-834, June 2000.
- [Clark88] G. C. Clark Jr., and J. B. Cain, *Error Correction Coding for Digital Communications*, Plenum Press, New York, 1988.
- [Conway99] T. Conway, "Implementation of High Speed Viterbi Detectors," Electronics Letters, vol. 35, no. 24, pp. 2089-2090, November 1999.
- [Fettweis91] G. Fettweis and H. Meyr, "High-Speed Parallel Viterbi Decoding: Algorithm and VLSI Architecture," IEEE Communications Magazine, vol. 29, no. 5, pp. 46-55, May 1991.
- [3Gpp01] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and Channel Coding FDD, 3GPP TS 45.003 Version 5.0.0 2001, URL: <http://www.3gpp.org>.
- [3Gpp02] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Multiplexing and Channel Coding FDD, 3GPP TS 25.212 Version 5.2.0 Release 5. URL: <http://www.3gpp.org>.
- [Gray91] C. A. Gray, M. J. S. Smith, J. Rowson, and M. O'Brien, "A Comparison of an ASIC Synthesized Design to a Schematic Entry Design for a Viterbi Decoder," Proc. of IEEE Custom Integrated Circuits Conference, May 1991.
- [Harada02] H. Harada and R. Prasad, *Simulation and Software Radio for Mobile Communications*, Artech House, Boston, 2002.
- [He00] K. He and G. Cauwenberghs, "Integrated 64-State Parallel Analog Viterbi Decoder," IEEE International Symposium on Circuits and Systems, vol. IV, pp. 761-764, May 2000.
- [Intel00] *Convolutional Encoder and Viterbi Decoder at 256 Kbps*, Product Overview, Intel Corporation, 2000.
- [Jia98] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "An Area-Efficient ACS Architecture for Viterbi Decoders," Proc. of the 16th NORCHIP Seminar, pp. 109-116, November 1998.
- [Kivioja99] M. Kivioja, J. Isoaho, and L. Vanska, "Design and Implementation of Viterbi Decoder with FPGA's," The Journal of VLSI Signal Processing, vol. 21, pp. 5-14, May 1999.
-

- [Kong96] J. J. Kong, Y. W. Park, M. G. Kim, Y. Kim, and Y. C. Jeung, "A Fast Serial Viterbi Decoder ASIC for CDMA Cellular Base Station," Proc. of IEEE TENCON, Digital Signal Processing Applications, pp. 333-337, November 1996.
- [Kosa02] I. Kosa, "Performance of IEEE802.11a Wireless LAN Standard Over Frequency-Selective, Slowly Fading Nakagami Channels in a Pulsed Jamming Environment," M.Sc. Thesis, Naval Post Graduate School, Monterey, California, December 2002.
- [Kwon97] H. M. Kwon, S. Tatanamahatana, and J. H. Shim, "One Viterbi Decoder with Data Rate Estimation for IS-95 CDMA Wireless Communications," Proc. of IEEE Global Telecommunications Conference, GLOBECOM, vol. 2, pp. 594-598, November 1997.
- [Lee97] L. H. C. Lee, *Convolutional Coding Fundamentals and Applications*, Artech House Inc., Boston, 1997.
- [Lin83] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, New Jersey, 1983.
- [Lou95] H. L. Lou, "Implementing the Viterbi Algorithm," IEEE Signal Processing Magazine, vol.12, pp. 42-52, September 1995.
- [Lsi90] 0.7- μ m Array-Based Products Databook, LSI Logic, April 1990.
- [Lutz00] E. Lutz, M. Werner, and A. Jahn, *Satellite Systems for Personal and Broadband Communications*, Springer-Verlag, Berlin, 2000.
- [Nikolic99] B. Nikolic, "Algorithmic and Circuit Techniques in Digital Viterbi Detector For Magnetic Recording Channel," Ph.D. Dissertation, Dept. of Electrical and Computer Engineering, University of California, Davis, 1999.
- [McEliece89] R. J. McEliece and I. M. Onyszchuk, "Truncation Effects in Viterbi Decoding," Proc. of IEEE Military Communications Conference (MILCOM), vol.1, pp. 541-545, October 1989.
- [Pandita99] B. Pandita and S.K. Roy, "Design and Implementation of a Viterbi Decoder using FPGAs," Proc. of the Twelfth International Conference on VLSI Design, pp. 611-614, January 1999.
- [Peterson95] R. L. Peterson, R.E. Ziemer, and D.E. Borth, *Introduction to Spread Spectrum Communications*, Prentice-Hall, New Jersey, 1995.
- [Proakis89] J. G. Proakis, *Digital Communications*, McGraw Hill, Singapore, 1989.
- [Qualcomm00] *Viterbi/Trellis Decoders*, Product listing, Qualcomm Incorporated, 2000.
-

- [Ryu99] J. H. Ryu, S. C. Kim, J. D. Cho, H. W. Park, and Y. H. Chang, "Lower Power Viterbi Decoder Architecture With a New Clock-Gating Trace-Back Unit," Proc. of 6th International Conference on VLSI and CAD (ICVC), pp. 297-300, October 1999.
- [Shakoor01] A. R. A. Shakoor, V. Szwarc, and T. A. Kwasniewski, "Low Power Viterbi Decoder Implementation for CDMA Applications," Proc. of World Multiconference on Systemics, Cybernetics and Informatics, vol. XIV, pp. 42-45, July 2001.
- [Shakoor03] A. R. A. Shakoor, V. Szwarc, and T. A. Kwasniewski, "A High Throughput, Area Efficient Viterbi Decoder for W-CDMA," International Signal Processing Conference (ISPC), Dallas, April 2003.
- [Shakoor04] A. R. A. Shakoor, V. Szwarc, and T. A. Kwasniewski, "High Speed Viterbi Decoder for WLAN and Broadband Applications," accepted for publication into The 2nd Annual North East Workshop on Circuits and Systems (IEEE-NEWCAS) Conference, to be held in Montreal in June 2004.
- [Shim99] B. Shim and J. C. Suh, "Pipelined VLSI Architecture of The Viterbi Decoder for IMT-2000," Proc. of Global Telecommunications Conference GLOBECOM, vol. 1a, pp. 158-162, December 1999.
- [Singh99] P. K. Singh and S. Jayasimha, "A Low-Complexity, Reduced-Power Viterbi Algorithm," Proc. of 12th International Conference on VLSI Design, pp. 61-66, January 1999.
- [Sklar88] B. Sklar, *Digital Communications, Fundamentals and Applications*, Prentice-Hall, New Jersey, 1988.
- [Sweeney02] P. Sweeney, *Error Control Coding From Theory To Practice*, John Wiley & Sons, Ltd., New York, 2002.
- [Temerinac95] M. Temerinac, O. Witte, R. Backes, W. Sinnhofer, and M. Klump, "A Programmable Architecture for Viterbi Decoder," Proc. of The International Conference on Signal Processing Applications and Technology, pp. 133-137, October 1995.
- [Truong92] T.K. Truong, M.-T. Shih, I. S. Reed, and E. H. Satorius, "A VLSI Design for a Trace Back Viterbi Decoder," IEEE Trans. on Communications, vol. 40, no. 3, pp. 616-624, March 1992.
- [Tsui00] C.Y. Tsui, R.S.K. Cheng, and C. Ling, "Low Power Rake Receiver and Viterbi Decoder Design for CDMA Applications," An International Journal of Wireless Personal Communications, no. 14, pp. 49-64, 2000.
- [Wang96] Z. Wang and S. B. Wicker, "An Artificial Neural Net Viterbi Decoder," IEEE Trans. on Communications, vol. 44, no. 2, pp. 165-171, February 1996.
-

[Wilson96] S. G. Wilson, *Digital Modulation and Coding*, Prentice-Hall, New Jersey, 1996.

[Xilinx] *IEEE 802.16a Complaint Viterbi Decoder Feature Set*, Product note, Xilinx Inc.

[Zaragoza02] R. H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, John Wiley & Sons, Ltd., New York, 2002.