FPGA IMPLEMENTATION OF ADAPTIVE VITERBI DECODER USING VHDL

By

BALASUBRAMANIAM.D

Reg. No. 1020106002

of

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution affiliated to Anna University, Coimbatore)

COIMBATORE - 641049

A PROJECT REPORT

Submitted to the

FACULTY OF ELECTRONICS AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF ENGINEERING

IN

APPLIED ELECTRONICS

APRIL 2012

BONAFIDE CERTIFICATE

Certified that this project report entitled "**FPGA IMPLEMENTATION OF ADAPTIVE VITERBI DECODER USING VHDL**" is the bonafide work of Mr.D.Balasubramaniam [Reg. no. 1020106002] who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Project Guide	Head of the Department
Dr. D. Mohanageetha	Dr. Rajeswari Mariappan

The candidate with university Register no. 1020106002 is examined by us in the project viva-voce examination held on …….

Internal Examiner External Examiner

ACKNOWLEDGEMENT

First I would like to express my praise and gratitude to the Lord, who has showered his grace and blessing enabling me to complete this project in an excellent manner. He has made all things beautiful in his time.

I express my sincere thanks to our beloved Director **Dr.J.Shanmugam, Ph.D.,** Kumaraguru College of Technology, I thank for his kind support and for providing necessary facilities to carry out the work.

I express my sincere thanks to our beloved Principal **Dr.S.Ramachandran, Ph.D.,** Kumaraguru College of Technology, who encouraged me in each and every steps of the project work.

I would like to express my sincere thanks and deep sense of gratitude to our HOD, **Dr.Rajeswari Mariappan, Ph.D.,** Department of Electronics and Communication Engineering, for her valuable suggestions and encouragement which paved way for the successful completion of the project work. I also thank her for her kind support and for providing necessary facilities to carry out the work.

In particular, I wish to thank and everlasting gratitude to the project coordinator **Ms.R.Hemalatha, M.E.,** Assistant Professor(SRG), Department of Electronics and Communication Engineering for her expert counseling and guidance to make this project to a great deal of success.

I am greatly privileged to express my deep sense of gratitude to my guide **Dr.D.Mohanageetha, Ph.D.,** Senior Associate Professor, Department of Electronics and Communication Engineering, Kumaraguru College of Technology throughout the course of this project work and I wish to convey my deep sense of gratitude to all the teaching and non-teaching of ECE Department for their help and cooperation.

Finally, I thank my parents and my family members for giving me the moral support and abundant blessings in all of my activities and my dear friends who helped me to endure my difficult times with their unfailing support and warm wishes.

ABSTRACT

In wireless data communications, errors occur frequently due to the non-overlook noises in the air. The convolutional coding technique is used to overcome this problem. By inserting additional bits into the transmitted bits, the convolutional coding technique can achieve good error resilience. When the transmission is done, the error bits can be corrected with the help of some particular decoding algorithms.

The Viterbi algorithm (VA) is often used to perform decoding procedure for convolutional code since it can obtain the maximum-likelihood decoding results. In the past, some high-speed VLSI designs for Viterbi decoder (VD) implementation were proposed to meet the need of high throughput applications. The main drawback of them is that they do not consider the issue of power consumption. VD may consume more than one-third of power for baseband processing in communication applications. Hence, it is desirable to have a low-power VD since the battery capacity of the most portable electronic communication devices is limited.

By computing and keeping all possible 2^{K-1} survivor paths, VA achieves an optimal performance of bit error rate (BER) with higher computational complexity and larger path storage requirement. In the hardware, higher complexity means more power is consumed. To reduce the power consumption required for VD, the adaptive Viterbi algorithm (AVA) has been proposed. It employs preset threshold values, T and Nmax, to reduce the average number of survivor paths and thus can decrease the power consumption efficiently. Hence, T and Nmax should be decided carefully.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

FPGA	**-------**	**Field-Programmable Gate Arrays**
VHDL	**-------**	**Very High Speed Integrated Circuit HDL**
HDL	**-------**	**Hardware Description Language**
VD	**-------**	**Viterbi Decoder**
VA	**-------**	**Viterbi Algorithm**
AVA	**-------**	**Adaptive Viterbi Algorithm**
AVD	**-------**	**Adaptive Viterbi Decoder**
BMU	**-------**	**Branch Metric Unit**
BMG	**-------**	**Branch Metric Generators**
ACSU	**-------**	**Add Compare Select Unit**
SMU	**-------**	**Survivor Memory Unit**

CHAPTER 1
INTRODUCTION

With the rapid development of communication technology and the emergence of various wireless devices, communication standards are becoming diverse. In such diverse environment, a mobile terminal must have the ability to adapt to various communication standards. Conventionally, this problem was solved by an adaptive solution that collects several single function components and then configures them statically or dynamically every time the standard changes. However, there is a multimode, multifunctional solution that has only one component and does not need to configure when the standard changes. To adapt to the current standard, the system can change the fabric instantaneously according to the parameters to switch among various functions. As a result, the mobile termination or wireless station can perform several different functions on a single hardware without extra fabric.

Among various wireless standards, Viterbi algorithm is the most popular decoding algorithm which is used to decode the convolutional code. The reconfigurable Viterbi decoder is the critical part of a multimode multifunctional wireless communication system, which needs to support parameters such as code rate, constraint length, polynomial and truncation length. Each standard can be described with a series of parameters that can be sent to the Viterbi decoder. The wireless communication standards using the Viterbi algorithm as the decoding algorithm are listed in Table 1. The code rates of conventional standards range from 1/2–1/3 and constraint lengths vary from 5–9.

Table 1.1 Channel Coding Parameter for Different Wireless Communication Standard

Standard	Code Rate	Constraint Length	Number of States
GPRS	½	5	16
GSM	½	5	16
WiMAX	½	7	64
802.11a/g	½	7	64
CDMA 2000	½,1/3	9	256
3G	1/3	9	256

1.1 MOTIVATION

Viterbi decoders are mostly implemented on three types of platforms, namely, DSPs, ASICs, and FPGAs. The flexibility and parallelism of DSPs are restricted by its Instruction Set Architectures. ASICs have the best area-power performance, but they have a long development cycle, high design cost, and fixed functionality. In contrast, FPGAs have higher flexibility at the cost of shorter development cycle and lower design cost. Multiple functions can be merged into a single fabric on FPGAs because of their high flexibility. As to the reconfigurable Viterbi decoder, the platform must be a combination of the flexibility of software and the high performance of hardware, which is the characteristic of FPGAs. FPGA is the most suitable platform for the reconfigurable Viterbi decoder.

1.2 PROJECT GOAL

The goal of the project is to implement a Viterbi decoder which uses adaptive Viterbi algorithm for decoding convolutional codes of constraint length K=7, rate r=1/3 and generator polynomials $(117,127,155)_8$.

1.3 OVERVIEW OF THE PROJECT

Convolutional codes become more powerful when its constraint length K increases and this increases the complexity of Viterbi decoders. The Viterbi Algorithm, a decoding algorithm for convolutional codes. Viterbi algorithm, the maximum likelihood decoding algorithm for convolutional codes, works well for less-complex codes, indicated by *constraint length K*, but it requires an exponential increase in hardware complexity to achieve greater decode accuracy. The VA's memory requirement and computation count pose a performance obstacle when decoding more powerful codes with large constraint lengths. The AVA reduces the average number of computations required per bit of decoded information. In the adaptive Viterbi algorithm, the number of candidate data sequences (survivor paths) retained per received symbol (transmitted data bit) varies over time.

1.4 LITERATURE SURVEY

An Adaptive Viterbi Decoder for constraint length K=7, rate=1/3 has been discussed in [1] and implemented on Spartan 3E FPGA using VHDL.

In [2], power reduction is achieved at the architecture level and its Register Transfer Level (RTL) implementation is discussed. A Viterbi Decoder with architectural modification for Add Compare Select Unit and clock gated Survivor Memory Unit have been designed for low power wireless applications. A decoder system with constraint length K=7 and rate=1/2 have been synthesized on FPGA using Verilog HDL. Additional circuitries in Branch Metric Unit for pre-calculated differences and pre-calculated comparisons were done.

A low-power, Viterbi Decoder for Software Defined WiMAX receiver used for decoding convolutional codes of constraint length K=7 and code rate r=1/2 with generating polynomials g(171,133)8 has been implemented [3] on Xilinx Vertex 2 Pro using VHDL. This Viterbi Decoder reduces the dynamic power by using the concept of traceback along with clock gating technique and disables the decoding circuits when traceback process is not activated. The number of input codes given to this decoder is 40 per frame and optimal traceback depth of 5*K was used.

The Viterbi Algorithm (VA) has two methods to extract the decoded bits: the Register Exchange (RE) and the Trace Back (TB). The RE technique is acceptable for trellises with only a small number of states, whereas the TB approach is acceptable for trellises with a larger number of states. A Viterbi decoder with constraint length K=3, rate=1/2 which can correct up to two errors in 16 bits of transmitted data had been discussed in [4]. The implementation of Viterbi Algorithm is given here in detail. The generator polynomial of convolutional encoder used are $(7, 3)_8$. Truncation length of 16 and Trace Back approach are employed in this decoder.

The Modified State Exchange (MSE) Algorithm that simplifies the Trace back approach with block decoding capability to achieve low power, low latency and low memory requirements were given in [5]. The Viterbi Decoder using MSE architecture for decoding (2,1,6) convolutional codes can record “survival state numbers” which is resulted decoded data and so no decision bits are required during Trace Back and storing.

A new Survivor memory unit architecture which combines the benefits of Trace-forward and Trace-backward is given in [6]. Its features were low power and low latency. The power consumption of this architecture is slightly higher than 3-pointer even Trace Back architecture.

When the constraint length used in the convolutional encoding process is increased, the more powerful the code is produced. A Viterbi Algorithm (VA) based on strongly connected trellis decoding of binary convolutional codes has been presented. The FPGA implementation of Viterbi Decoders with K=11 and code rate=1/3 has been discussed in [7]. Here the decoded data bits are stored in registers using shift update and selective update methods in Traceback approach.

[8] proposes a new class of hybrid VLSI architectures for survivor path processing to be used in Viterbi Decoders which combines the benefits of Register Exchange and Trace Forward algorithms. It can be efficiently applied to codes with a larger number of states where Traceback architectures are dominant. Latency and storage requirements are traded for implementation complexity.

A high performance soft input hard output prototyped version of Viterbi Decoder (VD) is presented in [10]. The generic parameters of Viterbi Decoders are constraint length, generator polynomials of encoder, metric size and number of surviving paths. Other parameters of VD such as the trellis window length, number of the best traceback paths, and the bit width of the metrics are kept generic as well. This prototyped VD accepts two soft inputs and makes a hard decision producing one bit to output.

The thesis in [11] analyses the different Viterbi decoders and implement a reconfigurable Viterbi Decoder with shared hardware structure for GSM, GPRS, EDGE and WiMAX technology. The shared hardware structure would help in limiting the number of logic cells used compared to if implementation were to be done separately. The designed decoder could be adapted to any wireless communications that requires robust error correction and also supports high data rate. The hardware implementation combines the Viterbi decoder of constraint length K=7 and K=5 and it could be used on EDGE (Enhanced Data rates for Global Evolution) and IEEE 802.11 (WLAN) / 802.16 (WiMAX) systems.

The Viterbi decoder in [13] uses pointer concept to obtain the decoded data and bit serial architecture for the Add Compare Select Unit implementation. A low power VD has been proposed in [14] using Trace Back and Register Exchange method for producing the decoded data bits.

The concept of convolutional codes, encoding, constraint length, code rate, Viterbi Algorithm, Viterbi decoding, Trellis diagram and Survivor paths were well explained and discussed in [15], [16], [17], [18]. The concept of VHDL and its programming were explained in [20], [21], [22], [23], [24], [25], [26], [27].

CHAPTER 2
CONVOLUTIONAL CODES

Error correction coding is used to detect and correct data transmission errors in communication channels. Encoding is accomplished through the addition of redundant bits to transmitted information symbols. These redundant bits provide decoders with the capability to correct transmission errors. The process of adding this redundant information is known as channel coding.

Convolutional coding and block coding are the two major forms of channel coding. Convolutional codes operate on serial data, one or a few bits at a time. Block codes operate on relatively large (typically, up to a couple of hundred bytes) message blocks. There are a variety of useful convolutional and block codes, and a variety of algorithms for decoding the received coded information sequences to recover the original data.

Convolutional Codes were first introduced by Elias in 1955, which offer an alternative to block codes for transmission over a noisy channel. Convolutional coding can be applied to a continuous input stream (which cannot be done with block codes), as well as blocks of data.

A convolutional encoder can be viewed as a finite state machine. A convolutional encoder consists of one or more shift registers and multiple XOR gates. The stream of information bits flows in to the shift register from one end and is shifted out at the other end. A convolutional encoder is a Mealy machine, where the output is a function of the current state and the current input. In convolutional coding, the encoded output of a transmitter (encoder) depends not only on the set of encoder inputs received during a particular time step, but also on the set of inputs received within a previous span of K-1 time units, where K is greater than 1. The parameter K is the constraint length of the code. The encoding of convolutional codes can be accomplished with shift registers and generator polynomials (XOR functions).

A convolutional encoder is represented by the number of output bits per input bit (v), the number of input bits accepted at a time (b), and the constraint length (K), leading to representation (v, b, K). Figure 1 depicts a (2, 1, 3) convolutional encoder since the

encoder accepts one input bit per time step and generates two output bits. The two output bits are dependent on the present input and the previous two input bits. The operation of the encoder can be represented by a state diagram. Nodes represent the present state of the shift register while edges represent the output sequence and point to the next state of transition.

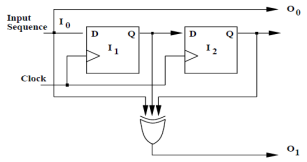


Figure 2.1 A (3, 1, 3) convolutional encoder

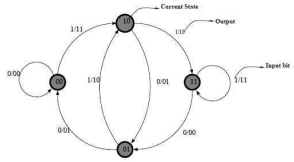


Figure 2.2 State diagrams for the convolutional encoder in Figure 2.1

Successive evaluation of state over time leads to the trellis diagram. The diagram is a time-ordered mapping of encoder state with each possible state represented by a point on the vertical axis. Nodes represent the present state of the shift register at specific points in time while edges represent the output sequence and point to the next state of transition. The horizontal axis represents time steps. Branch lines indicate the transition of the present state of the shift register to the next state upon receiving a particular input bit, b. The upper branch leaving a node implies an input of 0 while the lower branch implies an input of 1.

The code rate, k/n, is expressed as a ratio of the number of bits into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. The constraint length parameter, K, denotes the "length" of the convolutional encoder, i.e. how many k-bit stages are available to feed the combinatorial logic that produces the output symbols.

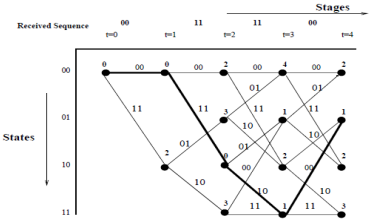


Figure 2.3 Trellis diagram for the convolutional encoder in Figure 2.1

The trellis diagram is used to display the computed branch and path measures, at each branch of initial state and each time sample. 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. The following table introduces the common generator sequence of the convolutional coding with constrain length from 7 to 9.

Table 2.1 Generator Polynomial for different Rate and Constraint Length of Viterbi Decoders

Rate	Constraint Length	Generator Polynomials
1/2	7	1001111, 1101101
1/2	8	10011111, 11100101
1/2	9	110101111, 100011101
1/3	7	1001111, 1010111,1101101
1/3	8	11101111,10011001,10101101

The location of stages is determined by the generator sequence, which also determines the minimum Hamming distance. Minimum Hamming distance determines the maximal number of correctable bits also the decoding performance. Usually with a lower coding rate and a longer constrain length; there must be a larger minimum Hamming distance.

The convolutional encoder used in this project has a constraint length K=7 and rate r=1/3. The encoder produces 3 output code bits (G0, G1, G2) for each input bit and has totally 64 states. The generator polynomials (are usually denoted in the octal notation) used in this encoder are G0=117_8, G1=127_8, G2=155_8. The convolutional encoder k=7 and r=1/3 is shown below.

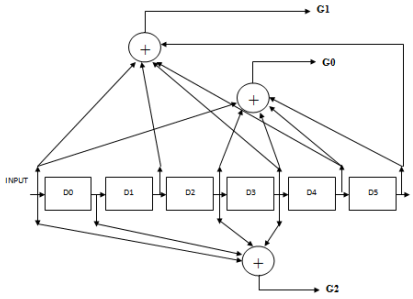


Figure 2.4 K=7, r=1/3 Convolutional Encoder

2.1 Applications

Convolutional codes are used extensively in numerous applications in order to achieve reliable data transfer, including digital video, radio, mobile communication, and satellite communication. These codes are often implemented in concatenation with a

hard-decision code, particularly Reed Solomon. Prior to turbo codes, such constructions were the most efficient, coming closest to the Shannon limit.

2.2 Popular convolutional codes

An especially popular Viterbi-decoded convolutional code, used at least since the Voyager program has a constraint length k of 7 and a rate r of 1/2.

- Longer constraint lengths produce more powerful codes, but the complexity of the Viterbi algorithm increases exponentially with constraint lengths, limiting these more powerful codes to deep space missions where the extra performance is easily worth the increased decoder complexity.

- Mars Pathfinder, Mars Exploration Rover and the Cassini probe to Saturn use a k of 15 and a rate of 1/6; this code performs about 2 dB better than the simpler k=7 code at a cost of 256× in decoding complexity (compared to Voyager mission codes).

2.3 Decoding convolutional codes

Several algorithms exist for decoding convolutional codes. For relatively small values of k, the Viterbi algorithm (VA) is universally used as it provides maximum likelihood performance and is highly parallelizable. Viterbi decoders (VD) are thus easy to implement in VLSI hardware and in software on CPUs with SIMD instruction sets.

Longer constraint length codes are more practically decoded with any of several sequential decoding algorithms, of which the Fano algorithm is the best known. Unlike Viterbi decoding, sequential decoding is not maximum likelihood but its complexity increases only slightly with constraint length, allowing the use of strong, long-constraint-length codes. Such codes were used in the Pioneer program of the early 1970s to Jupiter and Saturn, but gave way to shorter, Viterbi-decoded codes, usually concatenated with large Reed-Solomon error correction codes that steepen the overall bit-error-rate curve and produce extremely low residual undetected error rates.

CHAPTER 3

VITERBI ALGORITHM

3.1 BACKGROUND

A Viterbi Decoder (VD) uses the Viterbi Algorithm (VA) for decoding a bit stream that has been encoded using forward error correction based on a convolutional code. There are other algorithms for decoding a convolutionally encoded stream (for example, the Fano algorithm). The Viterbi Algorithm (VA) is the most resource-consuming, but it does the maximum likelihood decoding. It is most often used for decoding convolutional codes with constraint lengths k<=10, but values up to k=15 are used in practice. Viterbi decoding was developed by Andrew J. Viterbi and published in the paper "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", IEEE Transactions on Information Theory, Volume IT-13, pages 260-269, in April, 1967.

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events, especially in the context of Markov information sources, and more generally, hidden Markov models. The forward algorithm is a closely related algorithm for computing the probability of a sequence of observed events. These algorithms belong to the realm of probability theory.

The Viterbi algorithm was conceived by Andrew Viterbi in 1966 as a decoding algorithm for convolutional codes over noisy digital communication links. The algorithm has found universal application in decoding the convolutional codes used in both CDMA and GSM digital cellular, dial-up modems, satellite, deep-space communications, and 802.11 wireless LANs. It is now also commonly used in speech recognition, keyword spotting, computational linguistics, and bioinformatics. The Viterbi decoding algorithm is also used in decoding trellis-coded modulation, the technique used in telephone-line modems to squeeze high ratios of bits-per-second to Hertz out of 3 kHz-bandwidth analog telephone lines. Viterbi decoding is one of two types of decoding algorithms used with convolutional encoding-the other type is sequential decoding. Sequential decoding

has the advantage that it can perform very well with long-constraint-length convolutional codes, but it has a variable decoding time.

Viterbi decoding has the advantage that it has a fixed decoding time. It is well suited to hardware decoder implementation. But its computational requirements grow exponentially as a function of the constraint length, so it is usually limited in practice to constraint lengths of K = 9 or less. Stanford Telecom produces a K = 9 Viterbi decoder that operates at rates up to 96 kbps, and a K = 7 Viterbi decoder that operates at up to 45 Mbps. Advanced Wireless Technologies offers a K = 9 Viterbi decoder that operates at rates up to 2 Mbps.

3.2 ALGORITHM DESCRIPTION

A. J. Viterbi proposed an algorithm as an 'asymptotically optimum' approach to the decoding of convolutional codes in memory-less noise. The Viterbi algorithm (VA) is knows as maximum likelihood (ML)-decoding algorithm for convolutional codes. Maximum likelihood decoding means finding the code branch in the code trellis that was most likely to be transmitted. The algorithm is based on calculating the Hamming distance for every branch and the path that is most likely through the trellis will maximize that metric. Viterbi algorithm performs ML decoding by reducing its complexity. The algorithm reduces the complexity by eliminating the least likely path at each transmission stage. The path with the best metric is known as the survivor, while the other entering paths are non-survivors. If the best metric is shared by two or more paths, the survivor is selected from among the best paths at random. The selection of survivors lies at the heart of the Viterbi algorithm and ensures that the algorithm terminates with the maximum likelihood path. The algorithm terminates when all of the nodes in the trellis have been labeled and their entering survivors are determined. We then go to the last node in the trellis and trace back through the trellis. At any given node, we can only continue backward on a path that survived upon entry into that node. Since each node has only one entering survivor, our trace-back operation always yields a unique path. This path is the maximum likelihood estimate that predicts the most likely transmitted sequence. The Viterbi algorithm is an optimum algorithm for estimating the state sequence of a finite state process, given a set of noisy observations.

A data sequence x is encoded to generate a convolutional code word y. after y is transmitted through a noisy channel. The convolutional decoder takes the received vector r and generates an estimate z of the transmitted code word.

The maximum likelihood (ML) decoder selects the estimate that maximizes the probability p (r|z), while the maximum a posteriori probability (MAP) decoder selects the estimate that maximizes p(z|r). If the distribution of the source bits x is uniform, the two decoders are identical.

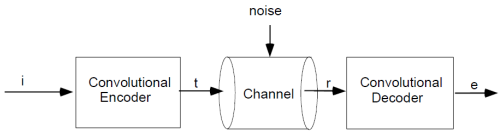


Figure 3.1 Convolutional Decoding

The Viterbi algorithm based on the ML algorithm and the hard decision is illustrated in figure3.2. The trellis in the figure corresponds to the convolutional encoder. The received code symbols are shown at the bottom of the trellis. The encoder encodes an input sequence (11010100) and generates the code word (111,000,001,001,111,001,111,110). This code word is transmitted over a noisy channel, and (101,100,001,011,111,101,111,110) is received at the other end. As mentioned earlier, the length of the trellis is equal to the length of the input sequence, which consists of the information bits followed by the reset sequence. The reset sequence, "00", forces the trellis into the initial state, so that the traceback can be started at the initial state.

A ML path is found with the aid of a branch metric and a path metric. A branch metric is the Hamming distance between the estimate and the received code symbol. The branch metrics accumulated along a path form a path metric. A partial path metric at a state, often referred as state metric, is the path metric for the path from the initial state to the given state. After the trellis grows to its maximal size, there are two incoming branches for each node. Between two branches, the branch with a smaller (in terms of Hamming distance) partial metric survives, and the other one is discarded. After

surviving branches at all nodes in the trellis have been identified, there exists a unique path starting and ending at the same initial state in the trellis. The decoder generates an output sequence corresponding to the input sequence for this unique path.

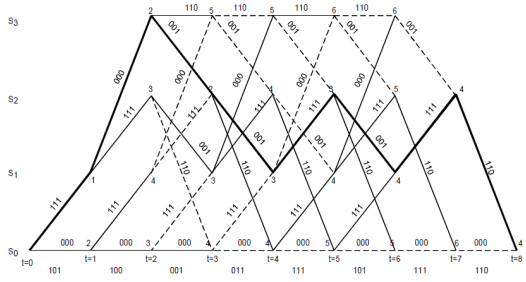


Figure 3.2 Viterbi Decoding for (3,1,3) Convolutional codes

The procedure is explained below using the trellis diagram in figure 3.2. The path metric for state S0 at time t=0 is initialized to zero. At time t=1 there is only one branch entering state S0. This branch metric is the Hamming distance between the expected input "000" and the received input "101", which is two. The path metric of S0 at time t=1 is the sum of the old path metric of S0 and the branch metric. Similarly, the path metric of S1 at t=1 is one. At t=1 there is only one branch entering these nodes. The sole branch is the survivor branch. The same process repeats for t=2. At t=3 there are two branches entering each node. For example, at state S0, a branch with the partial path metric six (which is the sum of the path metric 3 of S2 and the branch metric 3) enters to the state from S2. The other branch with the partial path metric four also enters the state from S0. Between the two branches, the branch from S0 survives and the other one is discarded. Surviving branches are depicted in solid lines and discarded ones are in dotted lines in Figure. Once the trellis is tagged with partial path metrics at each node, we perform a traceback to extract the decoded output sequence from the trellis. We start with state S0 at time t=8 and go backward in time. The sole survivor path leads to state S2 at time t=7.

From state S2 at time t=7, we traceback to S1 at time t=6. In this manner, a unique path shown in the bold line is identified. Note that each branch is associated with specific source input bit. For example, the branch from state S2 at time t=7 to node S0 at time t=8 corresponds to a bit '0' whose bit position is the seventh in the source input sequence. So while tracing back through the trellis, the decoded output sequence corresponding to these branches is generated.

3.3 Applications

The Viterbi decoding algorithm is widely used in the following areas:

- Decoding trellis-coded modulation (TCM), the technique used in telephone-line modems to squeeze high spectral efficiency out of 3 kHz-bandwidth analog telephone lines. The TCM is also used in the PSK31 digital mode for amateur radio and sometimes in the radio relay and satellite communications.
- Automatic speech recognition
- Decoding convolutional codes in satellite communications.
- Computer storage devices such as hard disk drives.

CHAPTER 4
VITERBI DECODER

A Viterbi Decoder consists of four functional units namely

- Branch Metric Unit,
- Add Compare Select Unit,
- Survivor Memory Unit, and
- Path Metric Memory Unit

The block diagram of general Viterbi Decoder is shown in Fig.4.1:

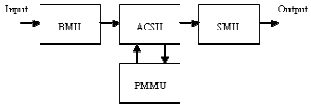


Figure 4.1 Functional blocks of VD

1. Branch Metric Unit (BMU): It calculates the hamming distance between the received code and the expected code for each path in trellis diagram.
2. Add-Compare-Select Unit (ACSU): It calculates the sum of current branch metrics and previous path metrics, then compares and selects the survivor path metric by discarding suboptimal trellis branches in each trellis stage and stores the decision bit.
3. Path Metric Memory Unit (PMMU): It stores the survivor path metric for ACSU to be used in the next cycle.
4. Survivor Memory Unit (SMU): Based on the decision bit from the ACSUs, the SMU produce the decoded bits along the reconstructed state sequence through the trellis and the survivor path metric.

Two main general approaches for the SMU are register exchange (RE) and trace back (TB) approaches. The hardware complexity of RE compared to that of TB is lower, but the power consumption is usually much higher compared to that of trace back. On the contrary, power consumption of TB is lower. However, larger memory and register requirement as well as higher latency are its drawbacks.

4.1 Architecture of K=7, rate r=1/3, 64 states Viterbi Decoder

The block diagram of basic Viterbi decoder is given in figure 4.1, but its architecture varies according to the constraint length, code rate and truncation length of convolutional codes. As the constraint length increases, its complexity also increases due to the computation of ACSUs. The VD starts with a single state, at initial stage. At each trellis stage less than K, the number of states enabled will be doubled and at trellis stage greater or equal to K, the number of states enabled will be $2^{(K-1)}$.

4.1.1 Branch Metric Unit (BMU)

This unit calculates the hamming distance between the received convolutional code from the channel and expected code for all the state transition from present state to the next state in the trellis diagram. This hamming distance is the branch metric for each of the 64 states.

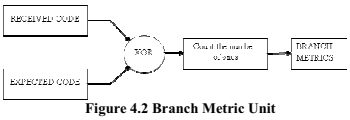


Figure 4.2 Branch Metric Unit

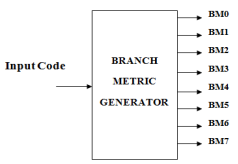


Figure 4.3 Branch Metric Generator

Each state may lead to two next states creating two paths upon receiving an input code. These paths may have expected code which could be understood from state transition table given below. Out of 64 states, 128 paths would be created and 16 paths

have same branch metric. Hence the branch metric generator has one input port and eight output ports as shown below.

4.1.2 Add Compare Select Unit (ACSU)

The ACSU is the important block in Viterbi decoder architecture as it computes the decision bit for each surviving path of each state. It selects the optimal path to each state in the Viterbi trellis. The ACS module decodes for each state in the trellis. The entire trellis is multiple images of the same simple element; a single circuit called Add-Compare-Select may be assigned to each trellis state.ACS is being used repeatedly in the decoder. A separate ACS circuit can be dedicated to every element in the trellis, resulting in a fast, massively parallel implementation. For a given code with rate 1/n and total memory M, the number of ACS required to decode a received sequence of length L is $L \times 2^M$. The ACSU is composed of 32 butterfly blocks and is given below. Thus for a constraint length K=7 decoder which has 64 states, there are 64 sub-blocks in the ACS block.

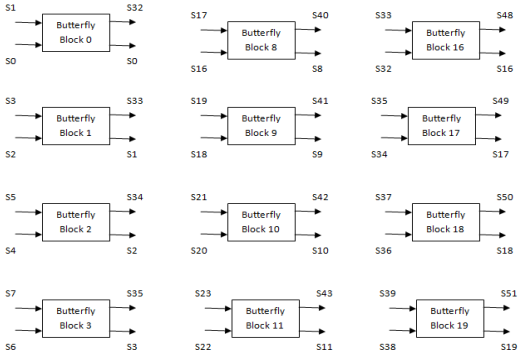


Figure 4.4 ACS components for 64 states in K=7 VD

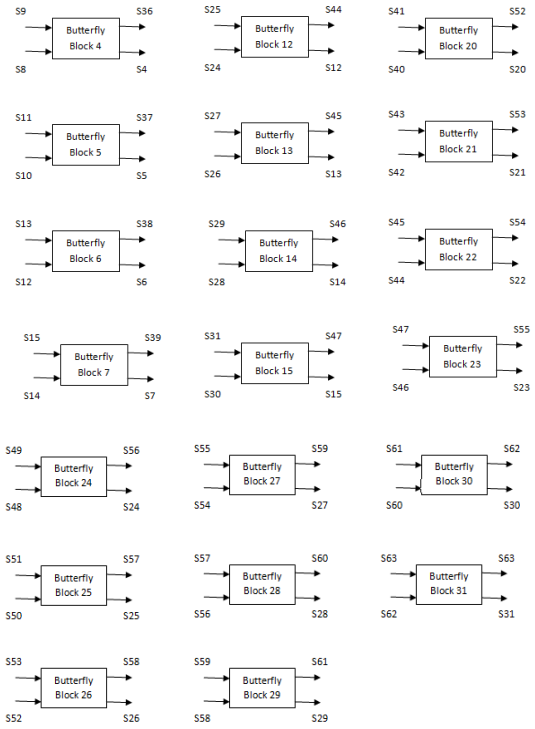


Figure 4.4 ACS components for 64 states in K=7 VD (cont.,)

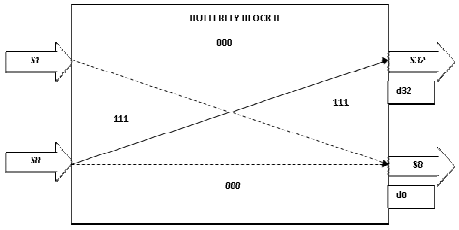


Figure 4.5 Butterfly Block 0 of ACSU

The above figure 4.5 shows the butterfly block 0 of ACSU. It has two input states S0, S1 with path metrics, which denotes the present state of Viterbi decoder. It has two output states S0, S32 with path metric, which denotes the possible next states for each present state in this block. S0 may lead to two next states {S0, S32} and the expected codes for these two transitions are {000, 111}. S1 may lead to two next states {S0, S32} and the expected codes for these two transitions are {111, 000}. These butterfly blocks are created from the state transition table given below. The Viterbi decoder does the reverse process of convolutional encoder. At state S0, the encoder's shift register consists of the value "000000". When input 1(0) bit is given to it, the shift register content would be "100000" ("000000") and the output code would be {000, 111}. Therefore the next states would be {S0, S32}. Like that the next states could be found out for all the present states in the Viterbi decoder. Let us denote the states with even number as even states and states with odd number as odd states. When the output states S0, S1 receives the path metric from input states S0, S32; it selects the path with the minimum value. This path is called surviving path. The decision bit d0 indicates whether the surviving path is from which state either even or odd and its value would be 0 for even and 1 for odd state. These decision bits from all the 32 butterfly blocks are stored in Survivor Memory Unit. A single wing for one butterfly block is given in the figure 4.6.

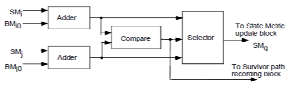


Figure 4.6 Single Butterfly Wing

The branch metrics for each two present states are added with its state metric for each transition to the same state and then compared and the minimum path metric is selected and the state is updated with its new value.

4.1.3 Survivor Memory Unit

The survivor memory unit is used for storing the decision bits which is used to produce the decoded bits. Here a RAM of size 64*22 and 64*38 is used for truncation length TL=22 and TL=38. When the decoder receives the complete set of codes equal to truncation length, then it starts producing the decoded bits using traceback technique. For TL=38, the path metrics of all the states at trellis stage 37 will be stored in memory array of size 64*1 of width 7 bits. Then the minimum path metric and its corresponding state are identified and used in traceback unit.

4.1.4 Trace Back Unit:

The final block in the decoder is traceback block. The actual decoding of symbols into original data is accomplished by tracing the maximum likelihood path backwards through the trellis. Up to a limit, a longer sequence of tracing results in a more accurate path through the trellis. After a number of symbols equal to at least six times the constraint length, the decoded data is output. The trace back starts from best state; the best state is estimated from the ACS costs.

4.1.5 State Table for 64 States Viterbi Decoder

The following is the state table for Viterbi decoder of constraint length K=7 and rate r=1/3. This table is derived from the state table of convolutional encoder.

Table 4.1 State Table for K=7, r=1/3 Viterbi Decoder

Present State	Next State SL Lower Path	Next State SU Upper Path	Expected Code Lower Path	Expected Code Upper Path
S0	S0	S32	000	111
S1	S0	S32	111	000
S2	S1	S33	011	100
S3	S1	S33	100	011
S4	S2	S34	111	000
S5	S2	S34	000	111
S6	S3	S35	100	011
S7	S3	S35	011	100
S8	S4	S36	101	010
S9	S4	S36	010	101
S10	S5	S37	110	001
S11	S5	S37	001	110
S12	S6	S38	010	101
S13	S6	S38	101	010
S14	S7	S39	001	110
S15	S7	S39	110	001
S16	S8	S40	010	101
S17	S8	S40	101	010
S18	S9	S41	001	110
S19	S9	S41	110	001
S20	S10	S42	101	010
S21	S10	S42	010	101
S22	S11	S43	110	001
S23	S11	S43	001	110
S24	S12	S44	111	000
S25	S12	S44	000	111
S26	S13	S45	100	011
S27	S13	S45	011	100
S28	S14	S46	000	111
S29	S14	S46	111	000
S30	S15	S47	011	100
S31	S15	S47	100	011
S32	S16	S48	100	011
S33	S16	S48	011	100

S34	S17	S49	111	000
S35	S17	S49	000	111
S36	S18	S50	011	100
S37	S18	S50	100	011
S38	S19	S51	000	111
S39	S19	S51	111	000
S40	S20	S52	001	110
S41	S20	S52	110	001
S42	S21	S53	010	101
S43	S21	S53	101	010
S44	S22	S54	110	001
S45	S22	S54	001	110
S46	S23	S55	101	010
S47	S23	S55	010	101
S48	S24	S56	110	001
S49	S24	S56	001	110
S50	S25	S57	101	010
S51	S25	S57	010	101
S52	S26	S58	001	110
S53	S26	S58	110	001
S54	S27	S59	010	101
S55	S27	S59	101	010
S56	S28	S60	011	100
S57	S28	S60	100	011
S58	S29	S61	000	111
S59	S29	S61	111	000
S60	S30	S62	100	011
S61	S30	S62	011	100
S62	S31	S63	111	000
S63	S31	S63	000	111

The Viterbi decoder of K=7 has 64 states. The following table shows the computations for which all states takes place at each trellis stage. The Viterbi decoder has a maximum of 64 states.

Table 4.2 States enabled for computation of survivor paths at each Trellis Stage

Trellis Stage 0	Trellis Stage 1	Trellis Stage 2	Trellis Stage 3	Trellis Stage4	Trellis Stage 5	Trellis Stage 6	Trellis Stage 7
Present States	Present States	Present States	Present States	Present States	Present States	Present States	Present States
S0	S0	S0	S0	S0	S0	S0	S0
	S32	S16	S8	S4	S2	S1	S1
		S32	S16	S8	S4	S2	S2
		S48	S24	S12	S6	S3	S3
			S32	S16	S8	S4	S4
			S40	S20	S10	S5	S5
			S48	S24	S12	S6	S6
			S56	S28	S14	S7	S7
				S32	S16	S8	S8
				S36	S18	S9	S9
				S40	S20	S10	S10
				S44	S22	S11	S11
				S48	S24	S12	S12
				S52	S26	S13	S13
				S56	S28	S14	S14
				S60	S30	S15	S15
					S32	S16	S16
					S34	S17	S17
					S36	S18	S18
					S38	S19	S19
					S40	S20	S20
					S42	S21	S21
					S44	S22	S22
					S46	S23	S23
					S48	S24	S24
					S50	S25	S25
					S52	S26	S26
					S54	S27	S27
					S56	S28	S28
					S58	S29	S29
					S60	S30	S30
					S62	S31	S31
						S32	S32
						S33	S33

						S34	S34
						S35	S35
						S36	S36
						S37	S37
						S38	S38
						S39	S39
						S40	S40
						S41	S41
						S42	S42
						S43	S43
						S44	S44
						S45	S45
						S46	S46
						S47	S47
						S48	S48
						S49	S49
						S50	S50
						S51	S51
						S52	S52
						S53	S53
						S54	S54
						S55	S55
						S56	S56
						S57	S57
						S58	S58
						S59	S59
						S60	S60
						S61	S61
						S62	S62
						S63	S63

CHAPTER 5
ADAPTIVE VITERBI DECODER

The Viterbi algorithm (VA) is often used to perform decoding procedure for convolutional code since it can obtain the maximum-likelihood decoding results. In the past, some high-speed VLSI designs for Viterbi decoder (VD) implementation were proposed to meet the need of high throughput applications. The main drawback of them is that they do not consider the issue of power consumption. VD may consume more than one-third of power for baseband processing in communication applications. Hence, it is desirable to have a low-power VD since the battery capacity of the most portable electronic communication devices is limited.

By computing and keeping all possible 2^{K-1} survivor paths, VA achieves an optimal performance of bit error rate (BER) with higher computational complexity and larger path storage requirement. In the hardware, higher complexity means more power is consumed. To reduce the power consumption required for VD, the Adaptive Viterbi Algorithm (AVA) is presented.

5.1 Adaptive Viterbi Algorithm

In this project, path pruning technique which is to be used in each trellis stage is the main concern. The AVA reduces the average computation and path storage by computing and retaining only the path which satisfies the following conditions:

1) A path is retained only when its cost is less than the sum of threshold T and minimum path cost d_m of the previous trellis stage.
2) The total number of surviving paths per trellis stage is limited to N_{max}.

The first criterion does not allow the high costs paths to be transmitted to the next trellis stage. The second criterion restricts the number of surviving paths to N_{max}. At each stage, the minimum cost of the previous stage d_m, threshold T, and maximum survivors N_{max} are used to prune the number of surviving paths. The effective use of AVA depends on the careful calculation of T and N_{max}. The average number of surviving paths retained at each trellis stage will be reduced for a small value of T which results in an increased

BER, since the decision on the most likely path has to be taken from a reduced number of survivor paths. For a larger value of T, the average number of survivor paths increases resulting in a reduced BER. Hence an optimal value of T and N_{max} so that the performance is not affected.

5.2 Architecture of Adaptive Viterbi Decoder

A high-level view of the implemented Adaptive Viterbi Decoder architecture is shown in figure. 5.1. The decoder contains a datapath and an associated control path. Like most Viterbi decoders, the datapath is split into four parts: the branch metric generators (BMG), add–compare– select (ACS) units, the survivor memory unit, and path-metric storage and control. A BMG unit determines distances between received and expected symbols. The ACS unit determines path costs and identifies lowest-cost paths. The survivor memory stores lowest-cost bit-sequence paths based on decisions made by the ACS units, and the path metric array holds per-state path metrics. The flow of data in the datapath and the storage of results is determined by the control path.

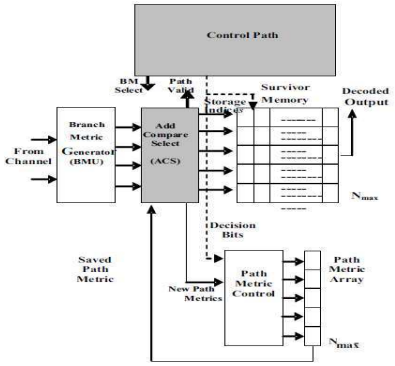


Figure 5.1. Adaptive Viterbi Decoder Architecture

In this Adaptive Viterbi decoder, the expected symbol value (*BM select*) is used to select the appropriate branch metric from the BMG, as shown at the left in figure 5.2. This branch metric value is combined with the path metric of its parent present state to form a new path metric, di. At each trellis stage, the minimum-value surviving path metric among all path metrics for the preceding trellis stage, dm, is computed. New path metrics are compared with the sum dm+T to identify path metrics with excessive cost.

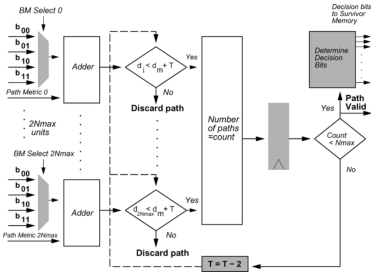


Fig.5.2. ACS unit of the Adaptive Viterbi Decoder

Comparators are then used to determine the life of each path based on the threshold, T. If the threshold condition is not satisfied by path metric dm + T, the corresponding path is discarded. Once the paths that meet the threshold condition are determined, the lowest-cost Nmax paths are selected. Sorting circuitry is eliminated by allowing feedback adjustments to the parameter T for each received symbol. If the number of paths that survive the threshold is less than Nmax, no iteration is required. As shown in figure 5.2, for stages when the number of paths surviving the threshold condition is greater than Nmax, T is iteratively reduced by two for the current trellis stage, until the number of paths surviving the threshold condition is equal to or less than Nmax. The AVA decoders for higher constraint length codes requires larger amount of logic resources and consumes more power than decoders for codes with smaller constraint length.

CHAPTER 6
RESULTS AND DISCUSSION

The simulation of this project has been done using MODELSIM SE 6.1f and synthesized it on a Xilinx SPARTAN 3 FPGA using Xilinx ISE 8.1i.

Modelsim is a simulation tool for programming {VLSI} {ASIC}s, {FPGA}s, {CPLD}s, and {SoC}s. Modelsim provides a comprehensive simulation and debug environment for complex ASIC and FPGA designs. Support is provided for multiple languages including Verilog, System Verilog, VHDL and System C.

The Spartan-3 generation of FPGAs includes the Extended Spartan-3A family (Spartan-3A, Spartan-3AN, and Spartan-3A DSP platforms), along with the earlier Spartan-3 and Spartan-3E families. These families of Field Programmable Gate Arrays (FPGAs) are specifically designed to meet the needs of high volume, cost-sensitive electronic applications, such as consumer products. The Spartan-3 generation includes 25 devices offering densities ranging from 50,000 to 5 million system gates.

The Spartan-3 platform was the industry's first 90 nm FPGA, delivering more functionality and bandwidth per dollar than was previously possible, setting new standards in the programmable logic industry. The Spartan-3E platform builds on the success of the earlier Spartan-3 platform by adding new features that improve system performance and reduce the cost of configuration. Because of their exceptionally low cost, Spartan-3 generation FPGAs are ideally suited to a wide range of consumer electronics applications, including broadband access, home networking, display/projection, and digital television equipment. The Spartan-3 generation FPGAs provide a superior alternative to mask-programmed ASICs. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. Also, FPGA programmability permits design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs.

The ISE Design Suite is the central electronic design automation (EDA) product family sold by Xilinx. The ISE Design Suite features include design entry and synthesis supporting Verilog or VHDL, place-and-route (PAR), completed verification and debug using Chip Scope Pro tools, and creation of the bit files that are used to configure the chip.

The VHDL code for K=5 VD; K=7 VD with TL=22 and TL=38; K=7, TL=38 VD with state enabling logic and K=7, TL=38 Adaptive Viterbi Decoder has been written. The simulation results for these decoders and convolutional encoder is given in the upcoming sections while the synthesis report will be listed in the latter sections.

6.1 SIMULATION RESULTS

6.1.1 Simulation Results for (3,1,7) Convolutional Encoder

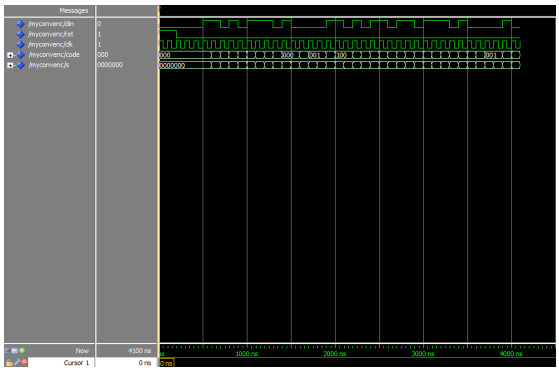


Figure 6.1 Simulation Results of (3,1,7) Convolutional Encoder

In the figure 6.1, "din" is the input sequence given to the encoder and "code" is the output 3 bit convolutional codes obtained according to the given input sequence. When one bit is given as input to the encoder, 3 bit output code is produced according to the generator polynomial of the encoder at rising edge of each clock cycle. When reset is given, the encoder's shift register contents would be set to all zeros therefore producing "000" at the output.

6.1.2 Simulation Result for K=5 Viterbi Decoder

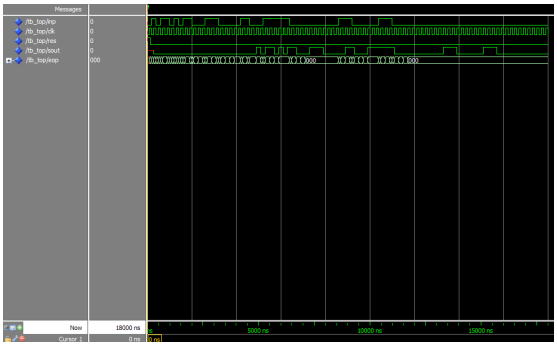


Figure 6.2 Simulation Result for K=5 Viterbi Decoder

The figure 6.2 shows the output of K=5 Viterbi Decoder where “inp” is the input sequence to the convolutional encoder, “eop” is the output convolutional code and “sout” is the decoded output sequence from the Viterbi decoder. After 44 clock cycles, the decoded output is produced at the “sout” output port of the K=5 Viterbi Decoder corresponding to the convolutional codes given to it. The truncation length used in this decoder is 22.

6.1.3 Simulation Result of Branch Metric Unit

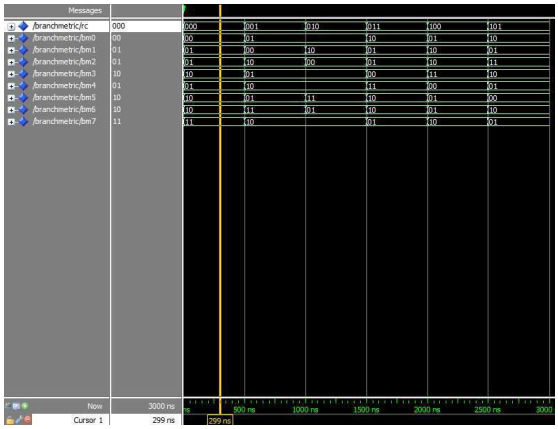


Figure 6.3 Simulation Result of Branch Metric Unit

In the figure 6.3, “rc” is the received code from the channel to the Viterbi Decoder and the eight branch metrics generated are bm0, bm1, bm2, bm3, bm4, bm5, bm6, bm7. When “000” is given as input to Branch Metric Generator, the outputs produced are “00”, “01”, “01”, “10”, “01”, “10”, “10”, “11”.

6.1.4 Simulation Result of Add Compare Select Unit

Figure 6.4 Simulation Result of Add Compare Select Unit

In the figure 6.4, the path metric for each two input states in butterfly block are added and compared to find the minimum path metric for each of the two output states. The path metrics of two input states are “1110001” and “0010101”. The branch metrics “01” and “10” are another two inputs which are added with each of the path metrics of two input states. From this addition, each output state has two metrics and they select the minimum path metric. The output decision bit du = ’1’ and dl = ’1’ denotes that two output states come from odd state.

6.1.5 Simulation Result of K=7 Viterbi Decoder for TL=22

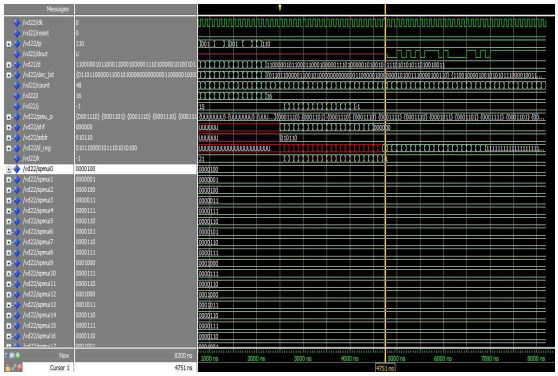


Figure 6.5 Simulation Result of K=7 Viterbi Decoder for TL=22

In the figure 6.5, an input sequence of 22 convolutional codes were given as input to the Viterbi decoder and after 49 clock cycles, it produces the decoded output bit corresponding to given convolutional codes. Here the truncation length used is of 22 codes.

6.1.6 Simulation Result of K=7 Viterbi Decoder for TL=38

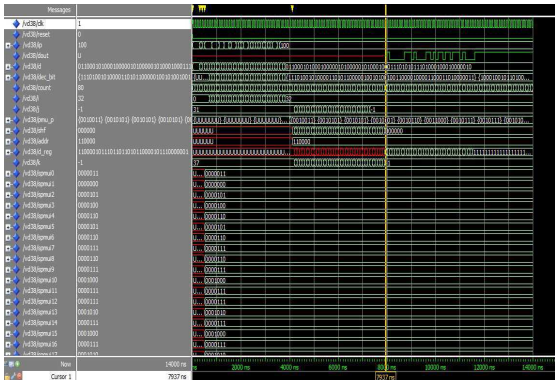


Figure 6.6 Simulation Result of K=7 Viterbi Decoder for TL=38

In the figure 6.6, an input sequence of 38 convolutional codes were given as input to the Viterbi decoder and after 81 clock cycles, it produces the decoded output bit corresponding to given convolutional codes. Here the truncation length used is of 38 codes.

In figure 6.7 a and 6.7 b, an input sequence of 38 convolutional codes were given as input to the Viterbi decoder and after 117 clock cycles, it produces the decoded output bit corresponding to given convolutional codes. Here the truncation length used is of 38 codes. Here at each clock cycle, the number of states enabled will be increasing to 64 at trellis stage 6 and thereafter it will be 64.

6.1.7 Simulation Results of K=7 Viterbi Decoder for TL=38 with State Enabling Block

Figure 6.7 a) Simulation Result of K=7 Viterbi Decoder with State Enabling Block

Figure 6.7 b) Simulation Result of K=7 Viterbi Decoder with State Enabling block

6.1.8 Simulation Results of K=7 Adaptive Viterbi Decoder for TL=38

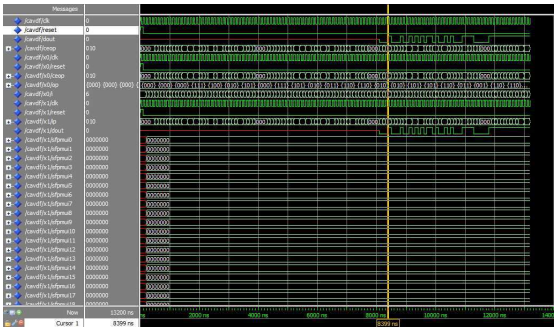


Figure 6.8 a) Simulation Result of K=7 Adaptive Viterbi Decoder

In figure 6.8 a, an input sequence of 38 convolutional codes were given as input to the Viterbi decoder and after 117 clock cycles, it produces the decoded output bit corresponding to given convolutional codes. Here the truncation length used is of 38 codes. Here at third clock cycle, the number of states enabled and number of survivor paths computed will be eight. At fourth and other clock cycles, the number of survivor paths will be kept less than or equal to eight, by calculating the state with maximum path metric and state with minimum path metric; taking the average of them and keeping it as a threshold value. The states with path metric value greater than threshold are discarded and those states which are less than it are transmitted to the next trellis stage.

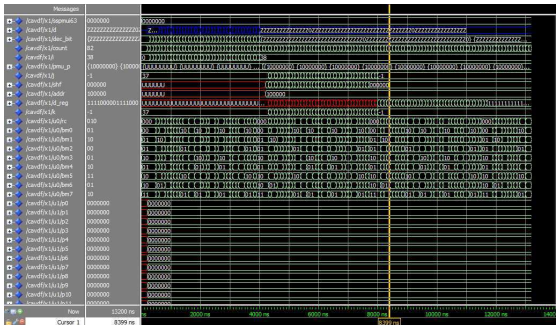


Figure 6.8 b) Simulation Result of K=7 Adaptive Viterbi Decoder

In figure 6.8 b, the decision bits whose states are enabled are stored in memory and others are discarded. At clock cycle = 39 the path metric of all the states are taken in an array, and the state with minimum path metric is found out. From that state, the traceback is done for clock cycles 41 to 78; the output bit decoded will be in the reverse order, so a last in first out register is implemented to produce the output in correct order. This register is activated for clock cycles 79 to116. At clock cycle = 117, the decoder produces the correct output bit sequence.

6.2 SYNTHESIS REPORT

The K=7 Viterbi Decoder and Adaptive Viterbi Decoder has been implemented on Xilinx Spartan 3E FPGA using Xilinx ISE 8.1i design tool. The device utilisation and timing report for the convolutional encoder, Viterbi Decoder and Adaptive Viterbi Decoder is as follows.

6.2.1 CONVOLUTIONAL ENCODER

Device utilization summary:

Selected Device : 3s1000fg320-4

Number of Slices: 6 out of 7680 0%

Number of Slice Flip Flops: 9 out of 15360 0%

Number of 4 input LUTs: 6 out of 15360 0%

Number of bonded IOBs: 6 out of 221 2%

Number of GCLKs: 1 out of 8 12%

Total equivalent gate count for design: 111

Additional JTAG gate count for IOBs: 288

Peak Memory Usage: 179 MB

Timing Summary:

Speed Grade: -4

Minimum period: 3.566ns (Maximum Frequency: 280.426MHz)

Minimum input arrival time before clock: 3.947ns

Maximum output required time after clock: 7.165ns

Maximum combinational path delay: No path found

6.2.2 K=7 VITERBI DECODER

Device utilization summary:

Selected Device : 3s1000fg320-4

Number of Slices: 3237 out of 7680 42%

Number of Slice Flip Flops: 1513 out of 15360 9%

Number of 4 input LUTs: 5882 out of 15360 38%

Number used as logic: 5498

Number used as RAMs: 384

Number of bonded IOBs: 6 out of 221 2%

Number of GCLKs: 6 out of 8 75%

Total equivalent gate count for design: 71,828

Additional JTAG gate count for IOBs: 288

Peak Memory Usage: 220 MB

Timing Summary:

Speed Grade: -4

Minimum period: 295.727ns (Maximum Frequency: 3.381MHz)

Minimum input arrival time before clock: 11.902ns

Maximum output required time after clock: 7.165ns

Maximum combinational path delay: No path found

6.2.3 K=7 ADAPTIVE VITERBI DECODER

Device utilization summary:

Selected Device: 3s1000fg320-4

Number of Slices: 1995 out of 7680 25%

Number of Slice Flip Flops: 764 out of 15360 4%

Number of 4 input LUTs: 3111 out of 15360 20%

Number used as logic: 2727

Number used as RAMs: 384

Number of bonded IOBs: 6 out of 221 2%

Number of GCLKs: 1 out of 8 12%

Total equivalent gate count for design: 49,946

Additional JTAG gate count for IOBs: 288

Peak Memory Usage: 203 MB

Timing Summary:

Speed Grade: -4

Minimum period: 285.267ns (Maximum Frequency: 3.505MHz)

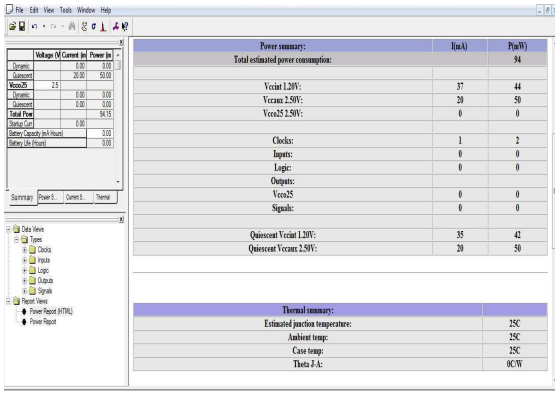
Minimum input arrival time before clock: 11.736ns

Maximum output required time after clock: 7.165ns

Maximum combinational path delay: No path found

6.3 POWER REPORT

6.3.1 CONVOLUTIONAL ENCODER



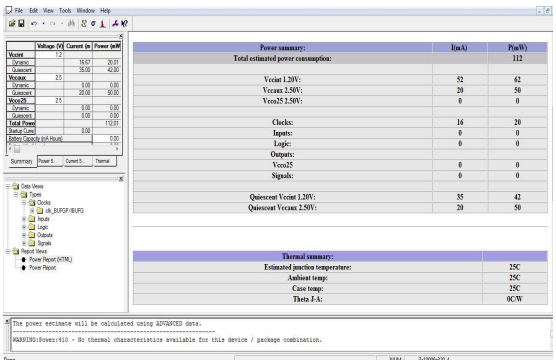
Power summary:	I(mA)	P(mW)
Total estimated power consumption:		94
Vccint 1.20V:	37	44
Vccaux 2.50V:	20	50
Vcco25 2.50V:	0	0
Clocks:	1	2
Inputs:	0	0
Logic:	0	0
Outputs:		
Vcco25	0	0
Signals:	0	0
Quiescent Vccint 1.20V:	35	42
Quiescent Vccaux 2.50V:	20	50

Thermal summary:	
Estimated junction temperature:	25C
Ambient temp:	25C
Case temp:	25C
Theta J-A:	0C/W

Figure 6.9 Power Report of (3,1,7) Convolutional encoder

The figure 6.9 shows that the power dissipated for (3,1,7) convolutional encoder is 94mW.

6.3.2 K=7 VITERBI DECODER



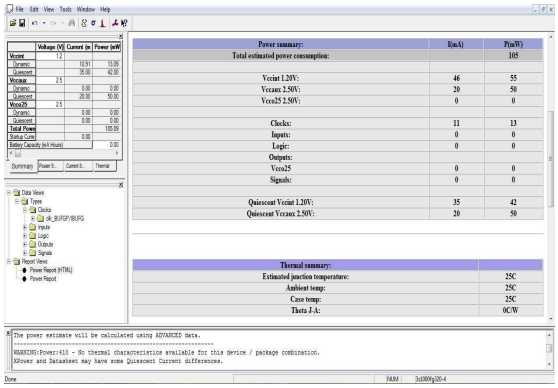
Power summary:	I(mA)	P(mW)
Total estimated power consumption:		112
Vccint 1.20V:	52	62
Vccaux 2.50V:	20	50
Vcco25 2.50V:	0	0
Clocks:	16	20
Inputs:	0	0
Logic:	0	0
Outputs:		
Vcco25	0	0
Signals:	0	0
Quiescent Vccint 1.20V:	35	42
Quiescent Vccaux 2.50V:	20	50

Thermal summary:	
Estimated junction temperature:	25C
Ambient temp:	25C
Case temp:	25C
Theta J-A:	0C/W

Figure 6.10 Power Report of K=7 Viterbi Decoder

The figure 6.9 shows that the power dissipated for K=7 Viterbi Decoder is 112mW.

6.3.3 K=7 ADAPTIVE VITERBI DECODER



Power summary:	I(mA)	P(mW)
Total estimated power consumption:		105
Vccint 1.20V:	46	55
Vccaux 2.50V:	20	50
Vcco25 2.50V:	0	0
Clocks:	11	13
Inputs:	0	0
Logic:	0	0
Outputs:		
Vcco25	0	0
Signals:	0	0
Quiescent Vccint 1.20V:	35	42
Quiescent Vccaux 2.50V:	20	50

Thermal summary:	
Estimated junction temperature:	25C
Ambient temp:	25C
Case temp:	25C
Theta J-A:	0C/W

Figure 6.11 Power Report of K=7 Adaptive Viterbi Decoder

The figure 6.9 shows that the power dissipated for K=7 Adaptive Viterbi Decoder is 105mW.

6.4 COMPARISONS

The results obtained from the synthesis report and power report of Viterbi Decoder and Adaptive Viterbi decoder are compared in the given table.

Table 6.1 Comparison of Viterbi Decoder and Adaptive Viterbi Decoder

SUMMARY	K=7 VITERBI DECODER	K=7 ADAPTIVE VITERBI DECODER	% REDUCTION
POWER	112mW	105mW	6.25
AREA Total Gate Count	71828	49946	30.46

In the table 6.1, the power and area for Viterbi Decoder and Adaptive Viterbi Decoder are presented and compared. The comparison shows that the Adaptive Viterbi decoder consumes less area and less power to that of Viterbi Decoder.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

A Viterbi decoder (VD) which uses Adaptive Viterbi Algorithm (AVA) for decoding (3,1,7) convolutional codes has been implemented using VHDL, simulated using ModelSim SE 6.1f and synthesized on Xilinx Spartan 3 FPGA using Xilinx ISE 8.1i design tool. This adaptive Viterbi decoder reduces the memory requirements and power consumption of VD using Viterbi Algorithm by reducing the number of surviving paths calculated at each trellis stage.

The future scope of this project is to design an adaptive Viterbi Decoder with specifications for variable constraint length, variable code rate and variable constraint length for decoding convolutional codes.

BIBLIOGRAPHY

1 S. Y. Ameen, M. H. Al-Jammas, A. S. Alenezi, “FPGA Implementation of Adaptive Viterbi Decoder”, *IEEE Transactions on Very Large scale Integration (VLSI) Systems,* September2011.

2 Sunil P. Joshi and Roy Paily “ Low Power Viterbi Decoder by Modified ACSU architecture and Clock Gating Method”, *IEEE International Symposium on VLSI Design Automation, and Test*, 2010, pp. 236-241.

3 S. W. Shaker, S. H. Alramely and K. A. Shehata, “Design and implementation of low-power Viterbi decoder for software-defined WiMAX receiver”, 17th Telecommunication Forum TELFOR, Serbia, Belgrade, 2009.

4 M. Pramod and M. K. Patil, “Implementation of Viterbi encoder and decoder on FPGA”, Project Report, Digital System Design with FPGAs, Indian Institute of Science, Nov. 2009.

5 Y. Tang, , D. Hu, , W. Wei,, W. Lin and H. Lin “A Memory-Efficient Architecture for Low Latency Viterbi Decoders” International Symposium on VLSI design, Automation and Test, VLSI-DAT09, Hsinchu, July, 2009.

6 C.-Y. Chu, Y.-C. Huang and A.Y. Wu, “Power Efficient Low Latency Survivor Memory Architecture for Viterbi Decoder*”. IEEE International Symposium on VLSI Design Automation, and Test*, 2008, pp. 228-231.

7 H. .S, Suresh and B. . .V, Ramesh, “FPGA implementation of Viterbi decoder”, Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, Corfu Island, Greece, February 16-19, 2007

8 M. Kamuf, V. Öwall and J. B. Anderson “Survivor Path Processing in Viterbi Decoders Using Register Exchange and Traceforward”, *IEEE Transaction on circuit and systems*, Vol. 54, No. 6, June 2007.

9 S. Swaminathan, R. Tressier, D. Goeckel., and W. Burleson, "A dynamically reconfigurable Viterbi decoder", *IEEE Transactions on Very Large scale Integration (VLSI) Systems, Vol. 13, No. 4, pp. 484-488 , April 2005.*

10 Obeid A. M., Ortiz A. G., Ludewig R., and Glenser M., "Prototype of a high performance generic Viterbi decoder", *Proceedings. 13th IEEE International Workshop on Rapid System Prototyping I2002.*

11 M.F.Batcha, “Implementation of Reconfigurable Viterbi Decoders in Hardware”, Universiti Teknologi Malaysia, April 2010.

12 “RTL implementation of Viterbi decoder”, Dept. Of Computer Engineering at Linkpings university, June 2006.

13 D. A. Wahed, “Low Power Register Exchange Viterbi Decoder for Wireless Applications,” University of Waterloo, April 2004.

14 S. Ranpara, “On a viterbi decoder design for low power dissipation,” Master’s thesis, Virginia Polytechnic Institute and State University, 1999.

15 Xilinx (2011), “ LogiCORE IP Viterbi Decoder v7.0 Product Specification ” , March 2011.

16 Stephen Fleming, “A Tutorial on Convolutional Coding with Viterbi Decoding” Spectrum Applications, June 2002.

17 Charan Langton, “Coding and decoding with convolutional codes”, http://www.complextoreal.com

18 S. Haykin. “*Communication Systems*” Wiley, 1994.

19 G.D.Forney, Jr. “ The Viterbi algorithm,” *Proceedings of the IEEE*, vol. 61, pp. 268 – 278, 1973.

20 A. J. Viterbi, “Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm,” IEEE Trans. on Information Theory, vol. IT-13, pp. 300-303, April 1967.

21 Michael J Smith, “Application Specific Integrated Circuits”, Pearson Education, 2008.

22 Zainalabedin Navabi, “VHDL Modular Design and Synthesis”, McGraw Hill, 2008.

23 Peter J. Ashenden, “The Designers Guide to VHDL”, Elsevier, 2008.

24 M. Morris Mano, “Digital design”, Pearson Education, 2006.

25 Volnei A. Pedroni, “Circuit Design with VHDL”, PHI, 2005.

26 Douglas L. Perry, “VHDL Programming by Example”, McGraw. Hill, 2002.

27 J. Bhasker, “A VHDL Synthesis Primer”, B S Publications, 2001.