DESIGN AND IMPLEMENTATION OF

AN OPTIMIZED VITERBI DECODER

A thesis submitted in partial fulfillment of the requirements

for the award of degree of

MASTER OF TECHNOLOGY

In

VLSI Design and CAD

Submitted By

Suneha Gupta

Roll No. 601061025

Under guidance of

Ms. Sakshi

Assistant Professor, ECED

T.U, Patiala

Department of Electronics and Communication Engineering

THAPAR UNIVERSITY, PATIALA

July 2012

CERTIFICATE

I hereby declare that the work which is being presented in the thesis entitled, **"An Optimized Viterbi Decoder"** in partial fulfillment of the requirement for the award of degree of M.Tech in VLSI Design & CAD submitted in Electronics and Communication Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Ms. Sakshi, Assistant Professor, ECED.

The matter presented in this thesis has not been submitted in any other University/Institute for the award of degree.

Date: 06/06/2012

(SUNEHA GUPTA)

Roll No: 601061025

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

(Ms. Sakshi)

Assistant Professor

ECED, Thapar University

Countersigned by:

(Dr.Rajesh Khanna)

Professor & Head

ECED, Thapar University

Patiala-147004

(Dr. S. K. Mohapatra)

Dean of Academic Affairs

Thapar University

Patiala-147004

ACKNOWLEDGEMENT

To discover, analyze and to present something new is to venture on an untrodden path towards and unexplored destination is an arduous adventure unless one gets a true torchbearer to show the way. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various people. Words are often too less to reveals one's deep regards. I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this thesis. I acknowledge with gratitude and humility my indebtedness to **Ms. Sakshi, Assistant Professor,** Electronics and Communication Engineering Department, Thapar University, Patiala, under whose guidance I had the privilege to complete this thesis. I wish to express my deep gratitude towards her for providing individual guidance and support throughout the thesis work.

I convey my sincere thanks to **Head of the Department, Dr. Rajesh Khanna** as well as **PG Coordinator, Dr. Kulbir Singh, Assistant Professor,** Electronics and Communication Engineering Department, entire faculty and staff of Electronics and Communication Engineering Department for their encouragement and cooperation.

My greatest thanks are to all who wished me success especially my parents. Above all I render my gratitude to the Almighty who bestowed self-confidence, ability and strength in me to complete this work for not letting me down at the time of crisis and showing me the silver lining in the dark clouds. I do not find enough words with which I can express my feelings of thanks to my dear friends for their help, inspiration and moral support which went a long way in successful competition of the present study.

(Suneha Gupta)

ABSTRACT

The vision of wireless communication providing high-speed and high-quality information exchange between two portable devices located anywhere in the world is the communications frontier of the current century**. In this busy and unsecure world you need to be sure your data is not only safe and secure but that you are working with it at the highest speed possible.** Convolutional encoding is a forward error correction technique that is used for correction of errors at the receiver end. Viterbi decoding is the technique for decoding the convolutional codes. The Viterbi Algorithm, an application of dynamic programming, is widely used for estimation and detection problems in digital communications and signal processing. It is used to detect signals in communications channels with memory, and to decode sequential error control codes that are used to enhance the performance of digital communication systems. The Viterbi decoding algorithm is widely used in radio communication: digital TV (ATSC, QAM, DVB-T,), radio relay and satellite communications.

This thesis represents the implementation of 3-bit soft decision viterbi decoder of constraint length K=7 with a code rate (R) of ½ and its algorithm. The decoder architecture is defined in VHDL and the circuit is simulated and synthesized on Xilinx: XC3S500E-4FG320 FPGA device. Frequency of soft decision viterbi decoder has been increased by using pipelining. Also puncturing technique is implemented in matlab to verify different transmission code rates. The plots of simulation results of bit error rate (BER) with different code rates are obtained from matlab code.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACS	Add compare and select
ASICs	Application specific integrated circuits
ACSTOSM	Add compare select to memory module
ACSU	Add compare and select unit
BM	Branch metric
BMU	Branch metric unit
CDMA	Code division multiple access
CPLD	Complex programmable logic devices
DSP	Digital signal processing
EPROM	Electrical programmable read only memory
EEPROM	Electrically erasable programmable read only
LSB	Least significant bit
ML	Maximum likelihood
MSB	Most significant bit
RE	Register exchange
SNR	Signal to noise ratio
TB	Traceback
VA	Viterbi algorithm
VD	Viterbi decoder

CHAPTER-1

INTRODUCTION

In digital communication system, error detection and error correction is important for reliable communication. Error detection techniques are much simpler than forward error correction (FEC). But error detection techniques have certain disadvantages. Error detection pre supposes the existence of an automatic repeat request (ARQ) feature which provides for the retransmission of those blocks, segments or packets in which errors have been detected. This assumes some protocol for reserving time for the retransmission of such erroneous blocks and for reinserting the corrected version in proper sequence. It also assumes sufficient overall delay and corresponding buffering that will permit such reinsertion. The latter becomes particularly difficult in synchronous satellite communication where the transmission delay in each direction is already a quarter second. A further drawback of error detection with ARQ is its inefficiency at or near the system noise threshold. For, as the error rate approaches the packet length, the majority of blocks will contain detected errors and hence require retransmission, even several times, reducing the throughput drastically. In such cases, forward error correction, in addition to error detection with ARQ, may considerably improve throughput. Forward error correction may be desirable in place of, or in addition to, error detection for any of the following reasons:

(1) When a reverse channel is not available or the delay with ARQ would be excessive.

(2) The retransmission strategy is not conveniently implemented.

1.1 CHANNEL CODING

It is known that noise-immunity is one of the basic attributes of information transmission systems. Since errors are possible in communication channels during the data transmissions we must apply error-correcting codes to combat these errors [1]. The purpose of forward error correction (FEC) is to improve the capacity of channel by adding some carefully designed redundant information to the data being transmitted through the channel. The process of adding this redundant information is known as channel coding.

The various building blocks of digital communication system shown in figure 1.1 are channel encoder, binary modulator, channel, demodulator, detector and channel decoder.

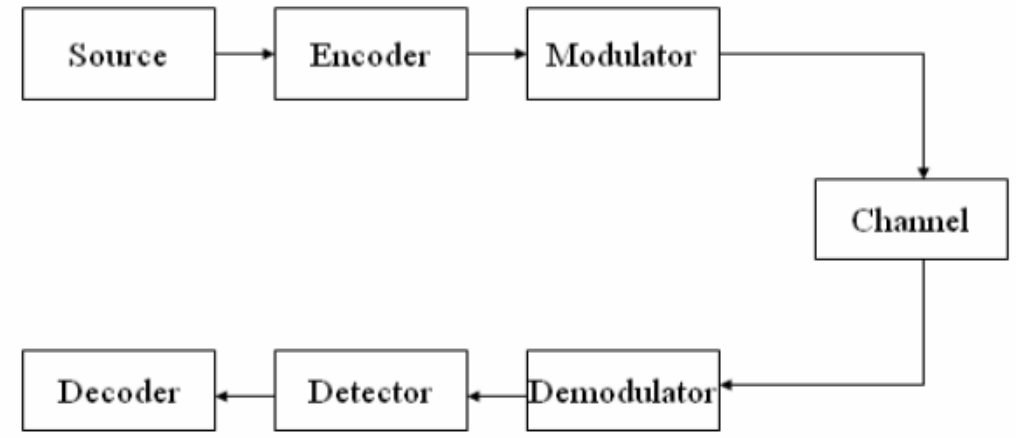

Figure 1.1: Digital Communication System[2]

Transmitter consists of encoder followed by modulator. The function of encoder is to introduce some redundancy in the binary information sequence in a controlled manner, which can be used at receiver to overcome the effects of noise and interference encountered in the transmission of signal through the channel. The encoding process generally involves taking k information bits at a time and mapping each K-bit sequence into a unique n-bit sequence, called a codeword. The amount of redundancy introduced by the encoding of the data in this manner is measured by the ratio n/k. The reciprocal of this ratio is called code rate. The binary sequence at the output of the channel encoder is fed to the modulator that modulates the signal.

At the receiving end of the digital communication system, the demodulator processes the corrupted waveform. The detector, which follows the demodulator, may decide on whether the transmitted bit is 0 or 1. This is known as hard decision. If quantization is used for the decision process, it is known as soft decision. The quantized output from the detector is fed to the channel decoder, which exploits the available redundancy to correct for channel disturbances [2].

1.2 MOTIVATION AND OBJECTIVES

Recently, convolutional codes have become more and more important in digital transmission. The Viterbi decoding algorithm is widely used in radio communication: digital TV (ATSC, QAM, DVB-T,), radio relay, satellite communications, PSK31 digital mode for amateur radio ,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), computer storage devices such as hard disk drives, automatic speech recognition . The vision of wireless communication providing high-speed and high-quality information exchange between two portable devices located anywhere in the world is the communications frontier of the current century**.** In this busy and unsecure world you need to be sure your data is not only safe and secure but that you are working with it at the highest speed possible**.**

The main objective of this thesis is first, to increase the performance of viterbi decoder. Second, to check and verify the functionality of viterbi decoder on Model Sim. Finally, to implement the design on Spartan 3E xc3s500E device.

1.3 ORGANISATION OF THESIS REPORT

Chapter1 deals with introduction of error correction coding in digital communication systems and objective of thesis.

Chapter2 discusses convolutional codes and viterbi algorithm. Further it gives introduction to pipelining.

Chapter3 discusses FPGA and its design flow.

Chapter4 discusses the proposed structure of viterbi decoder. Also shows the synthesis and simulation results of the design.

Chapter5 derives the conclusion and tells about the future scope.

CHAPTER-2

CONVOLUTIONAL CODES AND VITERBI DECODING

2.1 TYPES OF CHANNEL CODING

Convolution coding and block coding are two major forms of channel coding. Block codes operate on relatively large message blocks. Convolution codes operate on serial data, one or a few bits at a time [3].

2.1.1 BLOCK CODES

Early attempts at designing error control techniques were based on block codes. For every block of k information bits, n-k redundant parity-check bits are generated as linear (modulo-2) combinations of the information bits and transmitted along with information bits as a code of rate k/n bits/symbol. These can be generated by means of a linear feedback shift register encoder. Error detection can be easily implemented with any parity-check block code. At the decoder the received information bits are re-encoded into parity checks and compared bit-by-bit with the received redundant parity check bits. If any discrepancy occurs, a block error is declared. Shift register encoders and decoders in the form of a cyclic redundancy code can most easily implement this technique, called syndrome decoding. Some of the commonly used block codes are Hamming Codes, Golay Codes, BCH Codes, and Reed Solomon Codes. Emphasis in the last decade has turned to convolutional codes. Convolutional encoder may be viewed as a digital filter, whose output is the convolution of the input data and the filter impulse response. In almost every application, convolutional codes outperform block codes for the same implementation complexity of the encoder-decoder [4].

2.1.2 CONVOLUTIONAL CODES

Convolution coding with Viterbi decoding is a FEC technique that is particularly suited to a channel in which transmitted signal is corrupted mainly by additive white Gaussian noise (AWGN) [3]. In most of real time applications like audio and video applications, the convolutional codes are used for error correction.

Convolutional code definition parameters are the following: code rate (R), generating polynomial g (n), number of input bits (k), number of output bits (n) and constraint length (K). The code rate is the number of transmitted bits per input bit, e.g., a rate ½ encodes 1 bit and produces 2 bits for transmission. One generating polynomial stands for one output. The constraint length is the length of the generating polynomial in bits. The higher it is, the more robust is the code. Passing the information sequence to be transmitted through a linear finite shift register generates a convolutional code. The shift register consists of k bit stages and n linear algebraic function generators. The contents of shift register are multiplied by respective term in generator matrix and are then added together to generate respective code words.

The information symbols are encoded by using a convolution operation. Symbols, which are defined by coefficients in the generator polynomial, are added to each other and form output signal [5].

There are three alternative methods that are often used to describe the convolutional code. These are tree diagram, state diagram and trellis diagram. There are four basic convolutional codes decoding techniques : sequential, threshold, maximal-likelihood and the Viterbi algorithm. The sequential algorithm can provide very strong correcting capabilities while it needs relatively large memory, which strongly depends on communication channel error density. The threshold algorithm is extensively good for channels with mid to good signal to noise ratios (SNR). The Viterbi algorithm is an optimum decoding technique. It is optimum as it results in the minimum probability of error. It is also the relatively straight algorithm to implement in hardware and is the best decoding technique. Viterbi algorithm is a maximum likelihood algorithm and performs decoding through searching the minimum cost path in a weighted oriented graph, called trellis [6]. The basic building blocks of Viterbi decoder are branch metric unit (BMU), path metric unit (PMU), add compare and select unit (ACSU) and survivor memory management unit (SMU).

However, the complexity of the Viterbi decoder increases exponentially with the constraint length, so it is impractical to use codes with constraint lengths more than K>15 (3 to 9 is common practice) [1].

2.2 CONVOLUTIONAL CODES

The convolutional encoder is basically a finite state machine. The k bit input is fed to the constraint length K shift register and the n outputs are calculated from the generator polynomials by the modulo-2 addition. The generator polynomial specifies the connections of the encoder to the modulo-2 adder. The 1 in the generator polynomial indicates the connections and zero indicates no connections between the stage and the modulo 2 adder. The figure 2.1 below illustrates a simple convolutional encoder example with k=1, K=3, n=2, u1 = (1 1 1), u2 = (1 0 1) and R=1/2.

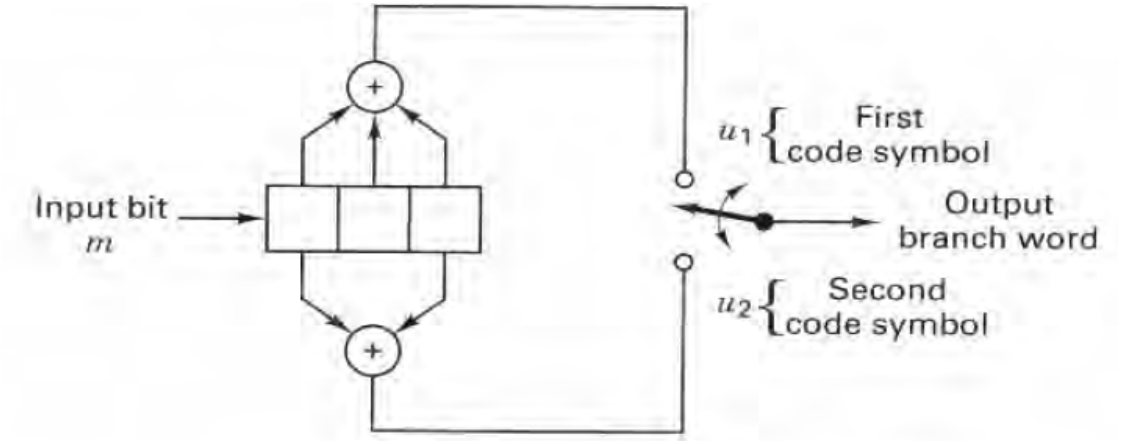

Figure 2.1: Convolutional Encoder[15]

Convolutional encoder can be described in terms of state table, state diagram and trellis diagram. The State is defined as the contents of the shift register of the encoder. In state table shown in table 2.1, output symbol can be described as a function of input symbol and the state. State diagram shows the transition between different states as shown in figure 2.2. Trellis diagram is the description of state diagram of the encoder by a time line i.e. to represent each time unit with a separate state diagram as shown in figure 2.3.

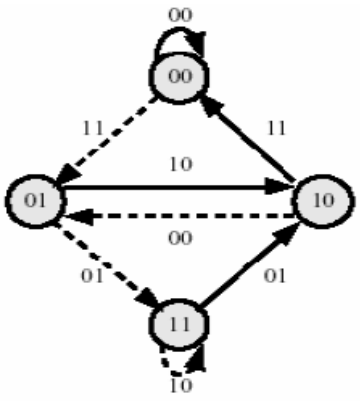

Figure 2.2: State Diagram[16]

Table 2.1: State Table[15]

INPUT U	PRESENT STATE (S1 ,S0)	NEXT STATE (S1,S0)	OUTPUT (V1,V2)
0	00	00	00
1	00	01	11
0	01	10	11
1	01	11	00
0	10	00	10
1	10	01	01
0	11	10	01
1	11	11	10

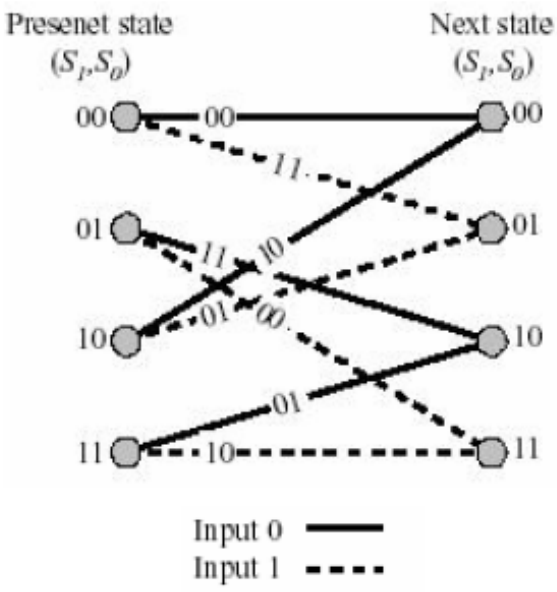

Figure 2.3: Trellis Diagram[3]

The encoding for the sequence 0 1 1 0 1 0 0 results in the output sequence 00 11 00 01 01 11 10 as shown in the figure 2.4.

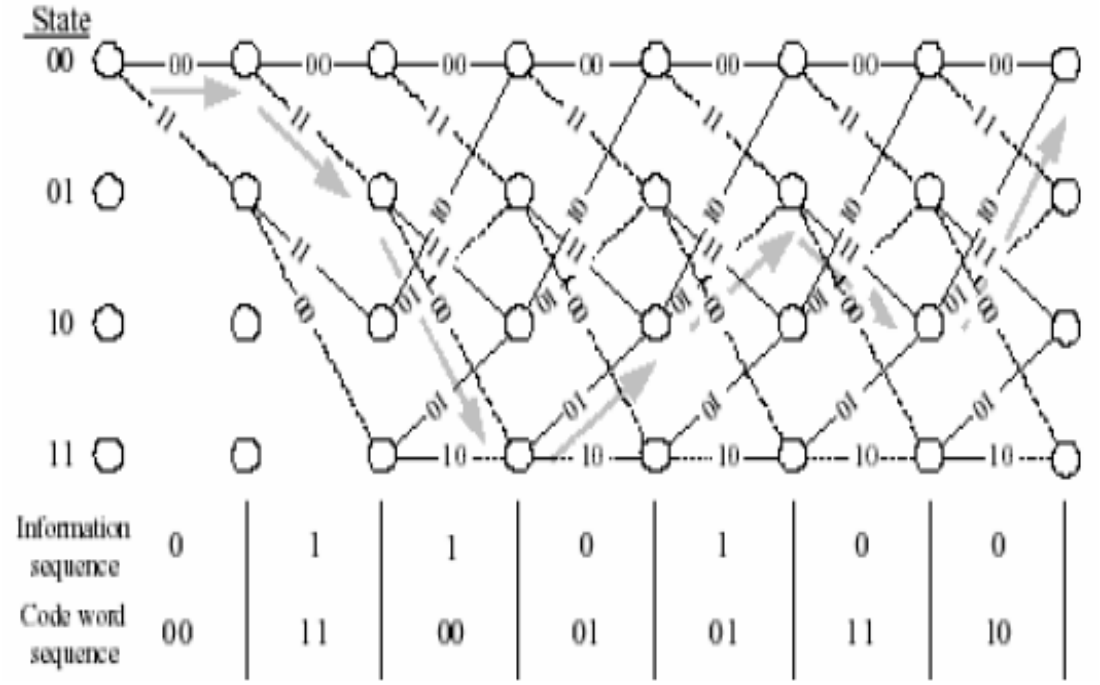

Figure 2.4: Trellis Diagram for Encoder[3]

2.3 VITERBI ALGORITHM

Viterbi algorithm was introduced in 1967 by Viterbi. Viterbi algorithm is called as optimum algorithm because it minimizes the probability of error. The algorithm can be broken down into the following three steps.

1. Weigh the trellis, that is, calculate the branch metrics.

2. Recursively computes the shortest paths up to time n, in terms of the shortest paths up to time n-1. In this step, decisions are used to recursively update the survivor path of the signal. This is known as add-compare-select (ACS) recursion.

3. Recursively finds the shortest path leading to each trellis state using the decisions from Step 2. The shortest path is called the survivor path for that state and the process is referred to as survivor path decode. Finally, if all survivor paths are traced back in time, they merge into a unique path, which is the most likely signal path. Figure 2.5 shows the flowchart for viterbi algorithm.

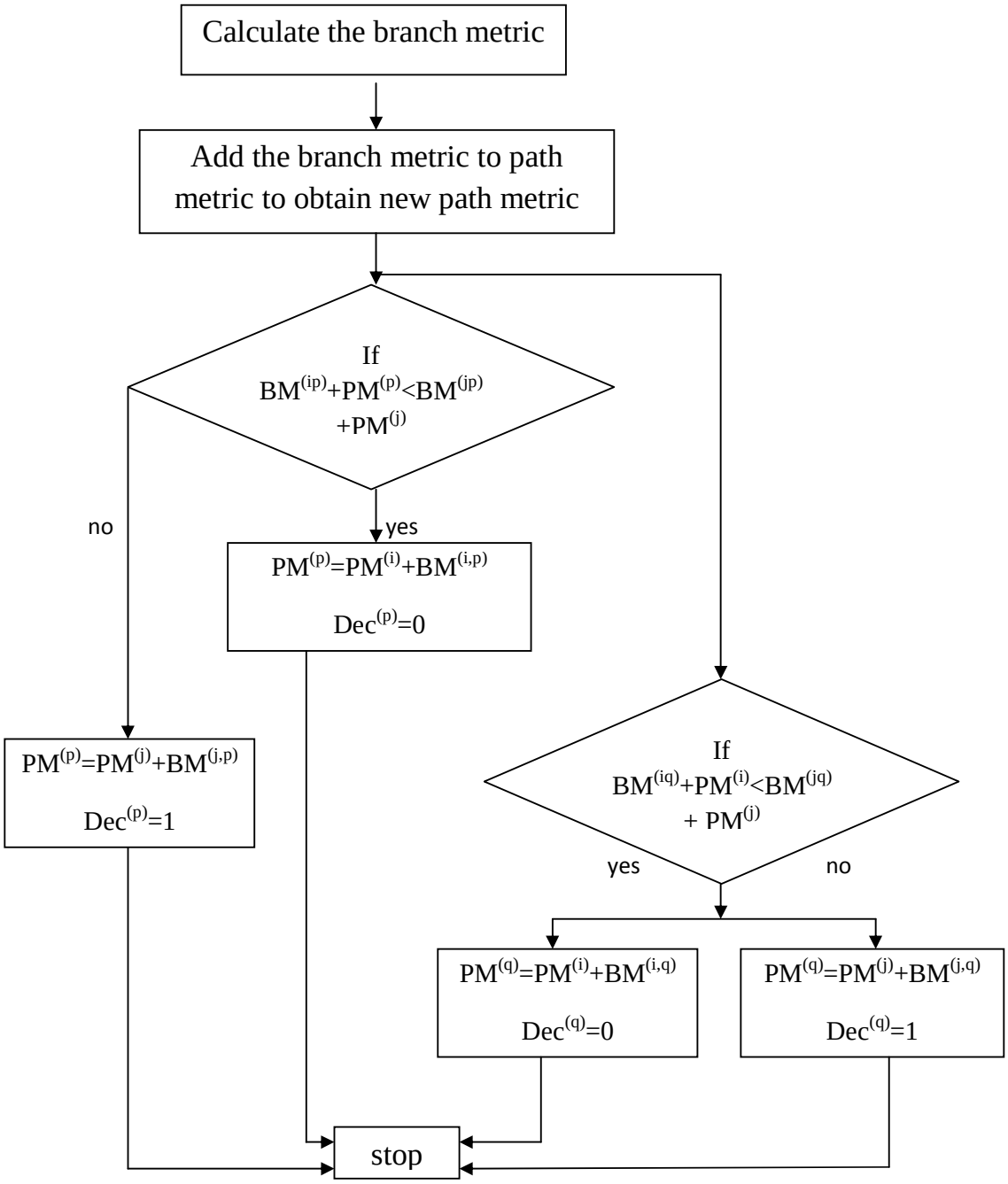

Figure 2.5: Flowchart for Viterbi Algorithm[15]

2.4 VITERBI DECODING FOR ERROR FREE CHANNEL

The sequence of figure 2.4 is assumed. Figure 2.6 shows the decoding sequence for two codewords received. The algorithm involves calculating measure of similarity or distance between received signal at time t_i and all trellis path entering each state at time

t_i. The hamming distance between the codewords being received and the output of encoder is calculated.

Consider the third code word being received. The path metrics are simply the addition of branch metric and previous path metric. For state (00), the partial path metric through the line marked 00 is 2, while the partial path metric through the line marked 10 is 3. The former is less than the later. So the survivor path of state (00) is through the solid line and path metric for the state is updated as 2 as shown in figure 2.7. The same operations are done for each state. The viterbi algorithm removes from consideration those trellis paths that could not possibily be candidates for maximum likelihood choice. When two paths enter the same state , the one having best metric is chosen , this path is called survivor path. The selection for survivor path is done for all states. The decoder continues in this way to advance deeper into trellis, making decisions by eliminating least likely paths.

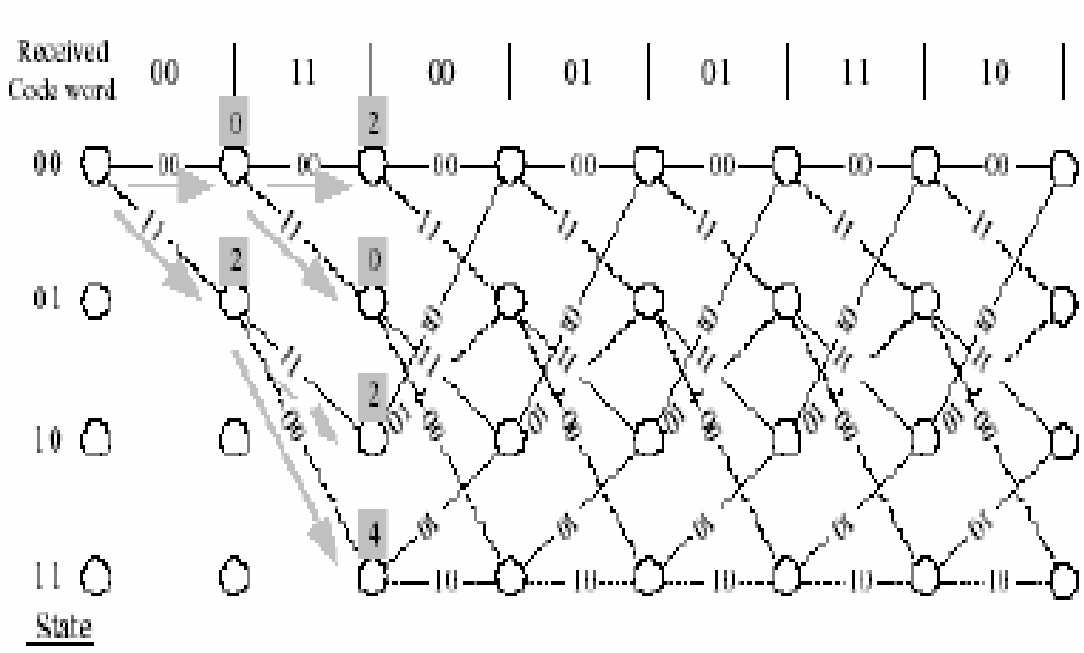

Figure 2.6: Decoding Example for Calculation of Hamming Distance[3]

When the computation of path metric is done, the next step is to decode the most similar sequence. The decoding process is denoted as traceback because this process is like tracing the sequence back. We trace the best path from the state with the minimum path metric. The final path is highlighted in figure 2.8.

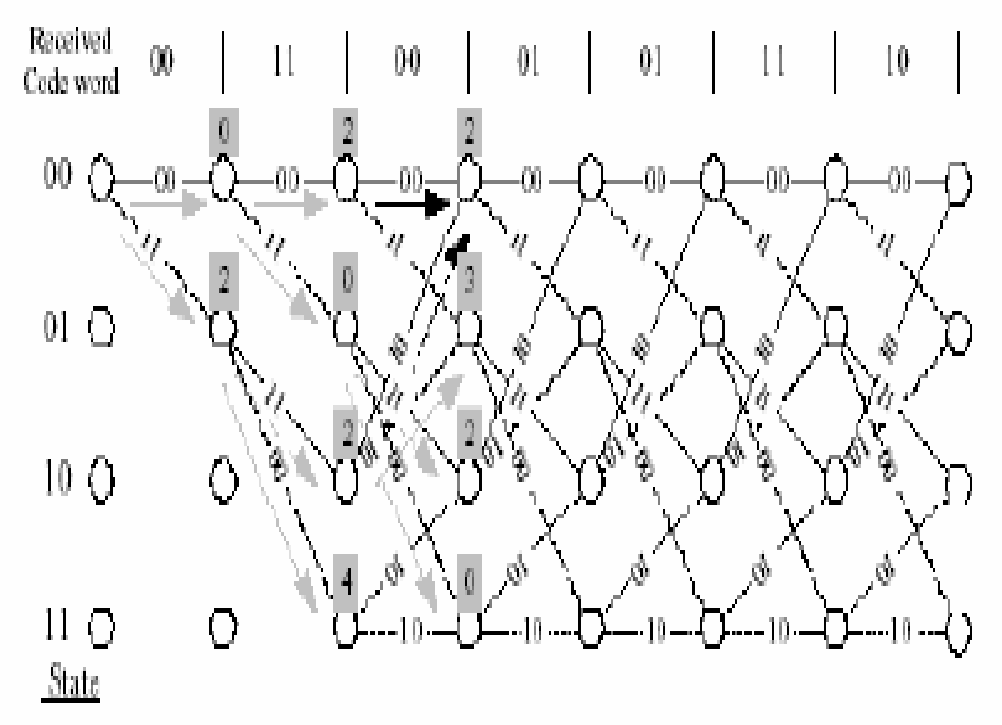

Figure 2.7: Decoding Example for Calculation of Path Metric[3]

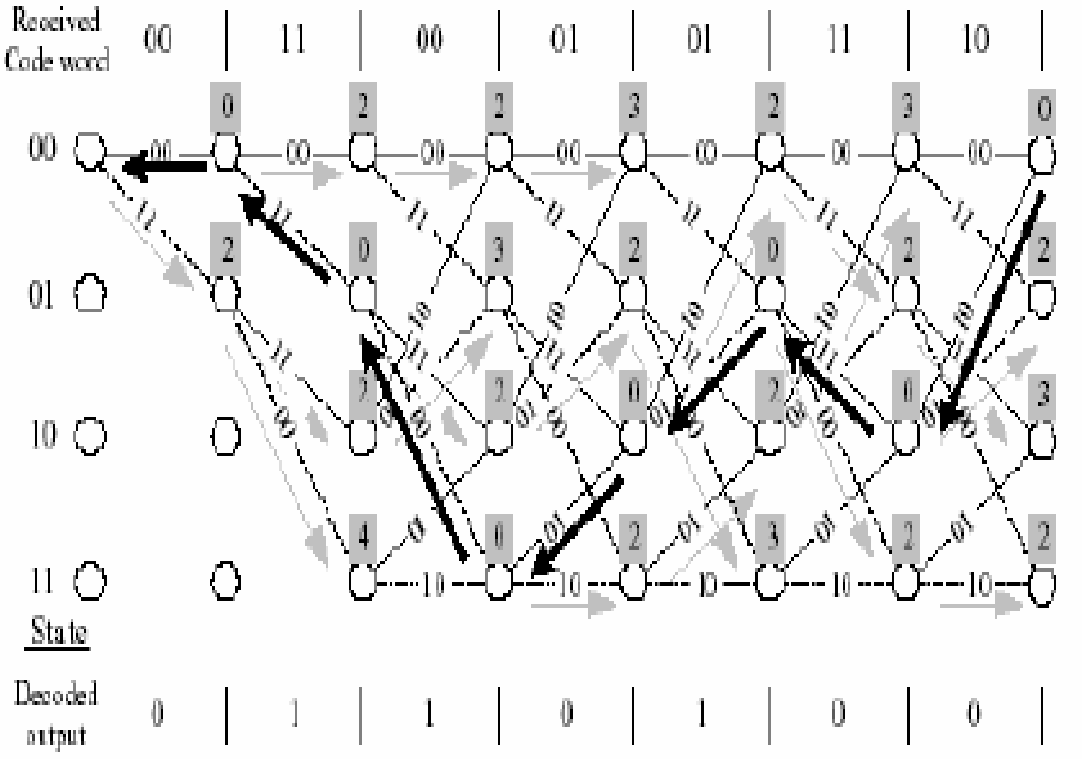

Figure 2.8: Decoding by Traceback Method[3]

2.5 VITERBI DECODING FOR NOISY CHANNEL

In trellis diagram for decoder it is convenient at each time interval to label each branch with hamming distance between received code symbols and branch words corresponding to same branch from encoder trellis. These encoder branch words are the code symbols that would be expected to come from encoder output as a result of each of state transition. As the code symbols are received , each branch of decoder trellis is labeled with a hamming distance between received code symbols and each of branch words for that time interval. The metric entered on the decoder trellis branch represents the difference between what was received and what should have been received had the branch word associated with that branch been transmitted. We continue labeling the decoder trellis branches in this way as the symbols are received at each time t_i.

The basis of viterbi decoding is if any two paths in trellis merge to a single state, one of them can always be eliminated in search for optimum path. Convolutional code in which k bits are shifted at a time into the shift register with K stages generates a trellis that has $2^{k(K-1)}$ states. Consequently, the decoding of such a code by means of VA requires keeping track of $2^{k(K-1)}$ surviving paths and $2^{k(K-1)}$ metrics. At each stage of the trellis, there are 2^k paths that merge at each node. Since each path that converges at common node requires the computation of a metric, there are 2^k metrics computed for each node. At each time t_i, there are 2^{K-1} states in the trellis , where K is constraint length and each states can be entered by two paths. The computation is done for each of 2^{K-1} states or nodes at time t_i , then the decoder moves to time t_{i+1} and repeats the process. At a given time the winning path metric for each state is designated as the state metric for that state.

Consider now the case in which an error is made on channel and one bit of the received symbol is incorrect. The fourth codeword has error. Figure 2.9 shows first four steps of decoding process. Figure 2.10 shows the complete decoding diagram.

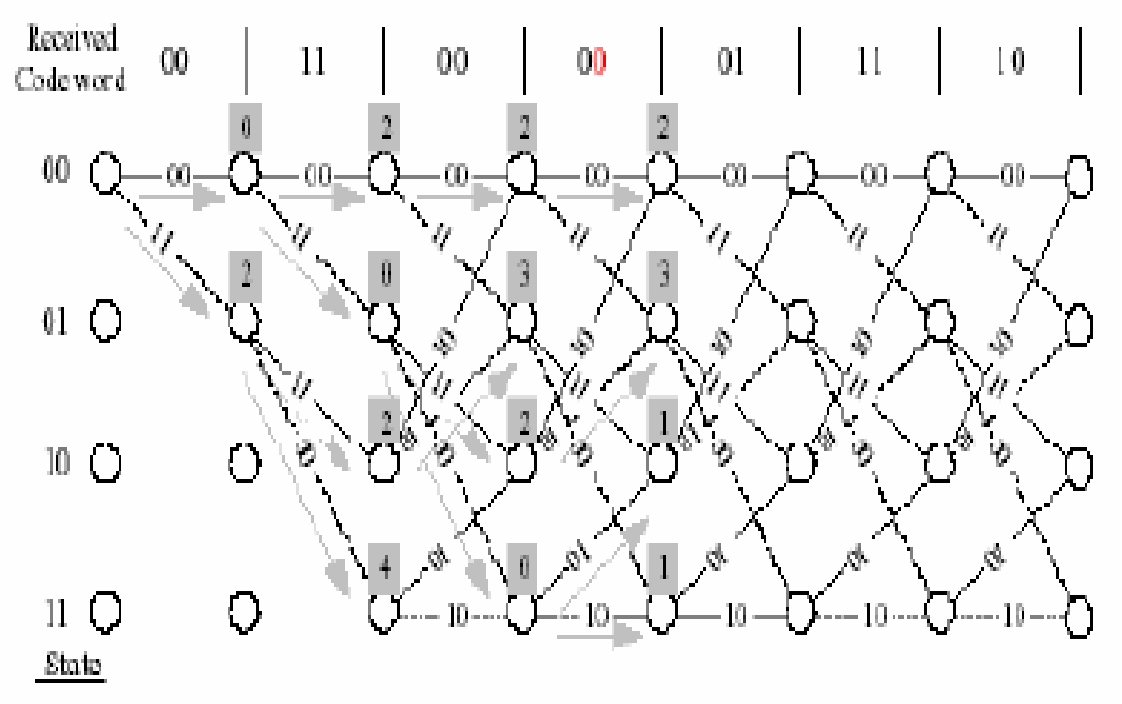

Figure 2.9: Decoding for Noisy Channel[3]

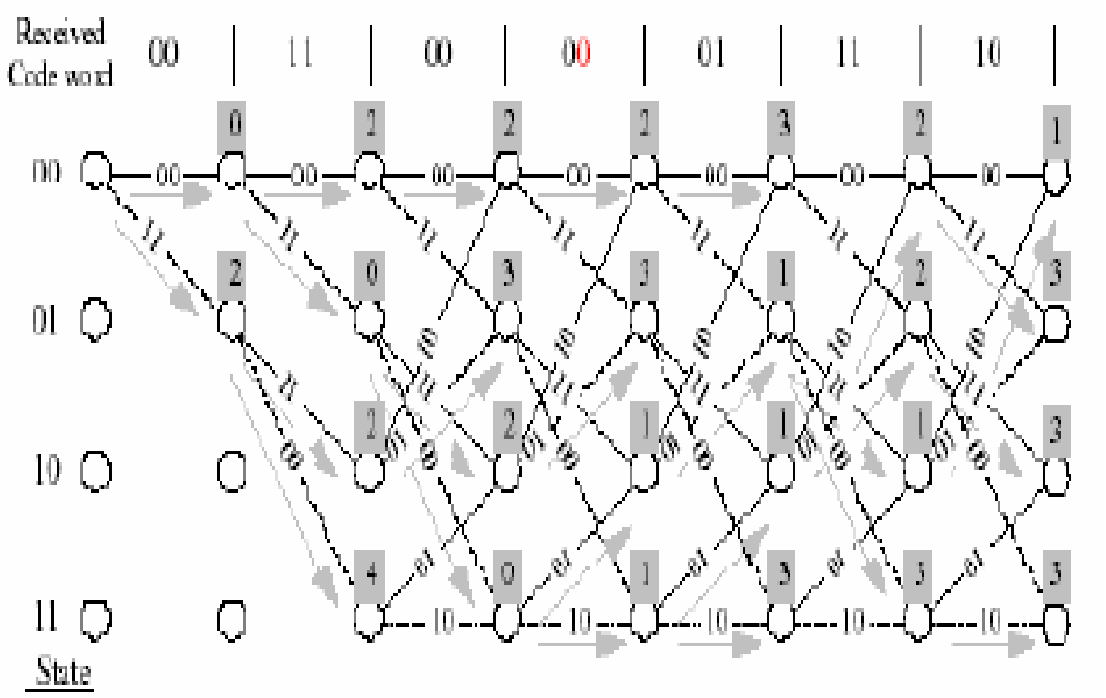

Figure 2.10: Complete Decoding Diagram[3]

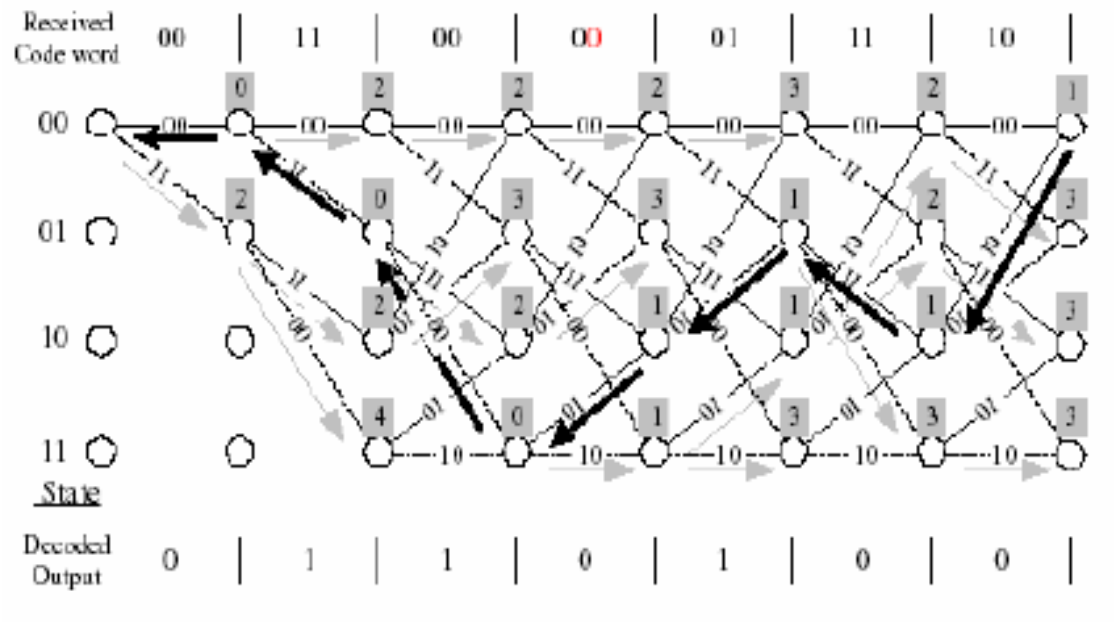

Figure 2.11: Decoding for Noisy Channel Using Traceback Method[3]

The figure 2.11 shows that when the final path is traced back even with one bit of error, output is correct.

2.6 VITERBI DECODER

The basic units of Viterbi decoder shown in figure 2.12 are branch metric unit, add compare and select unit and survivor memory management unit.

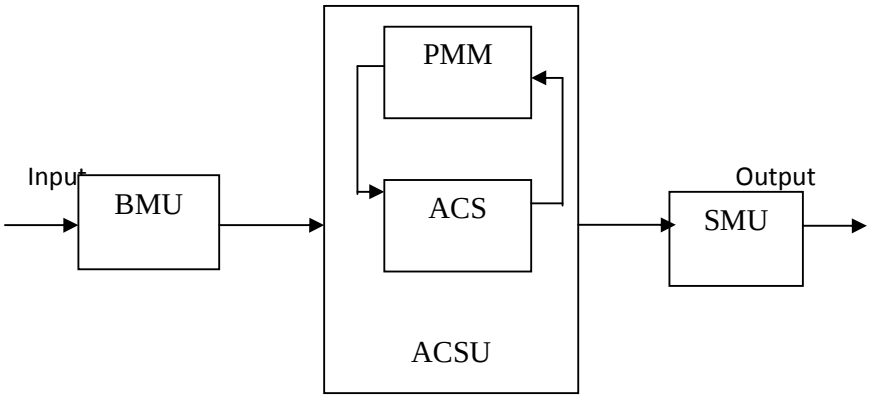

Figure 2.12: Block Diagram of Viterbi decoder[11]

2.6.1 BRANCH METRIC UNIT

The first unit is called branch metric unit. Here the received data symbols are compared to the ideal outputs of the encoder from the transmitter and branch metric is calculated. Hamming distance or the Euclidean distance is used for branch metric computation.

2.6.2 PATH METRIC UNIT

The second unit, called path metric computation unit, calculates the path metrics of a stage by adding the branch metrics, associated with a received symbol, to the path metrics from the previous stage of the trellis [10].

2.6.3 SURVIVOUR MEMORY MANAGEMENT UNIT

The final unit is the trace-back process or register exchange method, where the survivor path and the output data are identified . The trace-back (TB) and the register-exchange (RE) methods are the two major techniques used for the path history management in the chip designs of Viterbi decoders. The TB method takes up less area but requires much more time as compared to RE method because it needs to search or trace the survivor path back sequentially. Also, extra hardware is required to reverse the decoded bits. The major disadvantage of the RE approach is that its routing cost is very high especially in the case of long-constraint lengths and it requires much more resources [4].

2.6.3.1 Traceback method

In the TB method, the storage can be implemented as RAM and is called the path memory. Comparisons in the ACS unit are not the actual survivors that are stored. After at least L branches have been processed, the trellis connections are recalled in the reverse order and the path is traced back through the trellis diagram.

The TB method extracts the decoded bits, beginning from the state with the minimum PM. Beginning at this state and tracing backward in time by following the survivor path, which originally contributed to the current PM, a unique path is identified. While tracing back through the trellis, the decoded output sequence, corresponding to the traced branches, is generated in the reverse order.

2.6.3.2 Register exchange method

The register exchange (RE) method is the simplest conceptually and a commonly used technique. Because of the large power consumption and large area required in VLSI

implementations of the RE method, the trace back method (TB) method is the preferred method in the design of large constraint length, high performance Viterbi decoders[1]. In the register exchange, a register assigned to each state contains information bits for the survivor path from the initial state to the current state. In fact, the register keeps the partially decoded output sequence along the path, as illustrated in figure 2.13. The register of state S1 at t=3 contains '101'. This is the decoded output sequence along the hold path from the initial state .

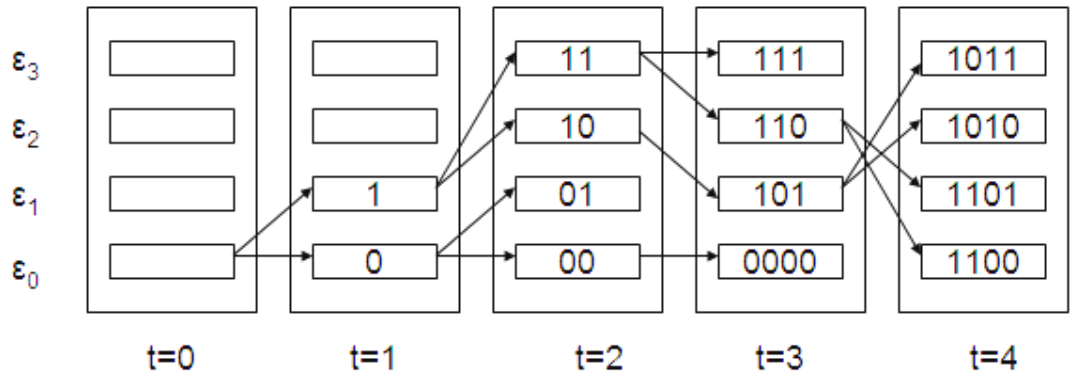

Figure 2.13: Register Exchange Method[1]

The register-exchange method eliminates the need to trace back since the register of the final state contains the decoded output sequence. However, this method results in complex hardware due to the need to copy the contents of all the registers in a stage to the next stage. The survivor path information is applied to the least significant bit of each register, and all the registers perform a shift left operation at each stage to make room for the next bits. Hence, each register fills in the survivor path information from the least significant bit toward the most significant bit. The scheme is called shift update. The shift update method is simple in implementation but causes high switching activity due to the shift operation and, hence, results in high power dissipation.

2.7 TYPES OF VITERBI DECODING

In order to realize a certain coding scheme a suitable measure of similarity or distance metric between two code words is vital. The two important metrics used to measure the distance between two code words are the Hamming distance and Euclidian distance adopted by the decoder depending on the code scheme, required accuracy, channel characteristics and demodulator type.

2.7.1 HARD DECISION VITERBI DECODING

In the hard-decision decoding, the path through the trellis is determined using the Hamming distance measure. Thus, the most optimal path through the trellis is the path with the minimum Hamming distance. The Hamming distance can be defined as a number of bits that are different between the observed symbol at the decoder and the sent symbol from the encoder. Furthermore, the hard decision decoding applies one bit quantization on the received bits.

2.7.2 SOFT DECISION VITERBI DECODING

In the Viterbi Algorithm, there are two ways to calculate the distance to choose a most likelihood path. One is hamming distance which is related to the hard decision. The other one is Euclidean distance related with soft decision. Soft-decision decoding is applied for the maximum likelihood decoding, when the data is transmitted over the Gaussian channel. On the contrary to the hard decision decoding, the soft-decision decoding uses multi-bit quantization for the received bits, and Euclidean distance as a distance measure instead of the hamming distance. The demodulator input is now an analog waveform and is usually quantized into different levels in order to help the decoder decide more easily. A 3-bit quantization results in 8 output values.

The Euclidean distance is used to calculate the straight line distance between two points. In a plane with p_1 at (x_1, y_1) and p_2 at (x_2, y_2), it is $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$. Figure 2.14 shows an example of Euclidean distance.

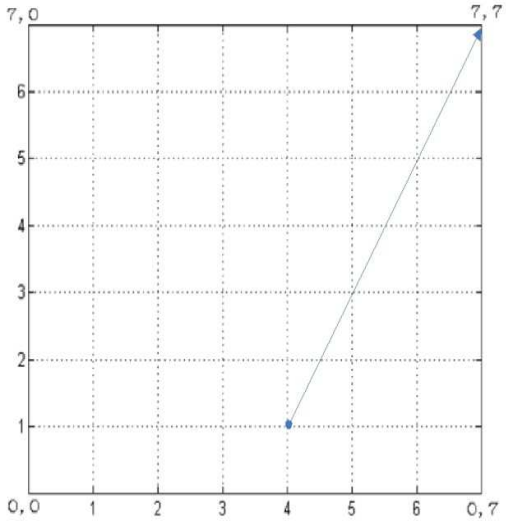

Figure 2.14: an example of Euclidean distance[20]

2.8 LIMITATION OF VITERBI ALGORITHM

When a binary convolutional code with k=1 and constraint length K is decoded by means of VA, there are 2^{K-1} states. Convolutional code in which k bits are shifted at a time into the shift register with K stages generates a trellis that has $2^{k(K-1)}$ states. Consequently, the decoding of such a code by means of VA requires keeping track of $2^{k(K-1)}$ surviving paths and $2^{k(K-1)}$ metrics. At each stage of the trellis, there are 2^k paths that merge at each node. Since each path that converges at common node requires the computation of a metric, there are 2^k metrics computed for each node. Of the 2^k paths that merge at each node, only one survives and this is the minimum distance path. Thus the number of computations in decoding performed at each stage increases exponentially with k and K. The exponential increase in computations and storage required to implement make it impractical for convolutional codes with large constraint length [2].

2.9 OTHER CODING TECHNIQUES

There are other structures and coding techniques that form excellent channel coding schemes in many practical applications. These structures include punctured convolutional codes, which effectively increase the rate R of a code by deleting periodically a code bit and concatenated codes, which combine two different code structures usually an RS outer code with an inner binary convolutional code to produce a powerful code. Also, another very important method, which was partly developed by Ramsey, is the convolutional interleaving scheme that finds many applications especially in channels where the errors occur in bursts. This technique uses an interleaver fed straight from the encoder, which is essentially a bank of registers and effectively separates each code bit from the other by changing their order in time before they are sent over the channel. At the receiver end, a synchronized de interleaver is used which performs the inverse operation and reassembles the original code sequence before it is fed to the decoder.

For the special case of k = 1, the codes of rates 1/2, 1/3, 1/4, 1/5, 1/7 are sometimes called mother codes. We can combine these single bit input codes to produce punctured codes, which give us code rates other than 1/n.

By using two rate 1/2 codes together, and then just not transmitting one of the output bits we can convert this rate 1/2 implementation into a 2/3 rate code. 2 bits come and 3 go out. This concept is called puncturing. On the receiver side, dummy bits that do not

affect the decoding metric are inserted in the appropriate places before decoding. This technique allows producing codes of many different rates using just a one simple hardware [1].

2.10 OTHER DECODING ALGORITHMS

Prior to the discovery of the VA, a number of other algorithms had been proposed for decoding convolutional codes. These were sequential decoding algorithm proposed by Wozen craft and subsequently modified by Fano. The Fano sequential algorithm searches for most probable path through the trellis by examining one path at a time. In this algorithm an additional negative constant is added to the each branch metric. The value of this constant is selected such that the metric of the correct path will increase on the average, while the metric of incorrect path will decrease on the average. By comparing the metric of a candidate path with an increasing threshold, Fano's algorithm detects and discards incorrect path. Its error performance is comparable to that of Viterbi decoding. In comparison with Viterbi decoding algorithm, sequential decoding has larger decoding delay. On the positive side, sequential decoding requires less storage than Viterbi decoding and hence it is more attractive for convolutional codes with large constraint length.

A third alternative is feedback decoding. In feedback decoding, decoding delay is significantly smaller than decoding delay in Viterbi decoder [2].

2.11 APPLICATIONS OF CONVOLUTIONAL CODES

2.11.1 SATELLITE APPLICATIONS

Applications in satellite communications can be more challenging in terms of high performance and high data rate requirements since both power and bandwidth impose significant limitations. The fixed INTELSAT and the mobile INMARSAT satellite systems employ mainly R = ½ and R = ¾, K = 7 convolutional codes of different data rates within 0.6-64 Kbit/s.

2.11.2 DIGITAL MOBILE APPLICATIONS

Convolutional codes are extensively used in digital mobile communication systems such as GSM, TIA IS 136 (telecommunication industry association interim standard 136) and

MPT (Ministry of posts and telecommunications) in cellular telephony of Europe, North America and Japan respectively. In GSM a combination of different block (cyclic) and he (2, 1, 5) convolutional codes together with interleaving techniques and puncturing are typically used, operating at data rates of 2.4-9.6 Kbits/s for data traffic channels and 5.6-13 Kbit/s for speech traffic channels.

2.11.3 DEEP SPACE APPLICATIONS

In deep space communications the power of the transmitted signal is the most severe limitation while there is no bandwidth restriction and therefore complicated coding schemes can be used to provide large error correcting capabilities.

Table 2.2: NASA Missions Convolutional Encoding Characteristics[7]

Mission	**Launch**	**R**	**K**	**Generator vectors**	**Maximum data rate**
PIONEER 9	1968	1/2	25	$g_1 = (40000000000)_8$ $g_2 = (71547370000)_8$	521 bit/s
PIONEER 9-11	1972-1973	1/2	32	$g_1 = (35565573735)_8$ $g_2 = (25565573735)_8$	2 Kbit/s
VOYAGER	1977	1/2	7	$g_1 = (171)_8$ $g_2 = (133)_8$	100 Kbit/s
VOYAGER	1977	1/3	7	$g_1 = (133)_8$ $g_2 = (171)_8$ $g_3 = (165_8$	100 Kbit/s
GALILEO	1989	1/4	15	$g_1 = (46321)_8$ $g_2 = (51271)_8$ $g_3 = (63667)_8$ $g_4 = (70535)_8$	Variable

Convolutional codes have been used in many NASA (National Aeronautics and Space administration) missions since the l960. Table 2.2 summarizes the characteristics of the codes used in Pioneer 9-11, Voyager and Galileo projects launched for solar orbit, Jupiter and Saturn missions and exploration of other planets. In Voyager and Galileo missions the convolutional codes shown in the table were also combined with an outer RS code in concatenation in order to improve their performance. The RS (255, 223) code

was concatenated with the inner (3, 1, 7) convolutional code when Voyager was transmitting images of Pluto and Neptune to earth. The same RS code was used in concatenation with the (4, 1, 15) convolutional code employed by Galileo spacecraft.

2.11.4 DIGITAL BROADCASTING APPLICATION

Digital broadcasting evolved in the early 1990's improved significantly the transmission of audio and video information signals. Nowadays, both digital audio (DAB) and video (DVB) broadcasting employ convolutional coding in their error correction scheme. Specifically, DAB uses a punctured convolutional code where a higher code rate $R_p > R$ can be achieved from a mother code, which is decoded using the same decoder. Also, Terrestrial (DVB-T) and satellite (DVB-S) utilize an inner convolutional code with puncturing concatenated with an outer modified (255, 239) RS code. Table 2.3 summarizes the characteristics of convolutional codes employed by DAB, DVB-S and DVB-T.

Table 2.3: Encoding Characteristics Employed in Digital Broadcasting[7]

Service	R	K	Generator vectors	R_P	Data rate
DAB	1/4	7	$g_1 = (133)_8$ $g_2 = (171)_8$ $g_3 = (145)_8$ $g_4 = (133)_8$	1/2 ,1/3	2.304
DVB-T	1/2	7	$g_1 = (171)_8$ $g_2 = (133)_8$	1/2, 2/3, 3/4, 5/6, 7/8	26-54
DVB-S	1/2	7	$g_1 = (171)_8$ $g_2 = (133)_8$	1/2, 2/3, 3/4, 5/6, 7/8	Variable

2.11.5 CODE DIVISION MULTIPLE ACCESS

CDMA being interference based use forward error correction schemes like convolutional coding to increase cell capacity. Viterbi algorithm is main building block of CDMA. CDMA uses Viterbi decoder with constraint length 9 and data rate 1/3. The generator polynomials are (557,663,711). A three bit soft decision is used[19].

2.11.6 ULTRA WIDE BAND APPLICATIONS

Convolutional codes are widely used for ultra wide band applications (UWB). Viterbi decoder is used for multiple orthogonal frequency division multiplexing. UWB communication systems have widely been used for communication usage. Multiple systems have been proposed for high speed wireless personal area networks. In these systems, convolutional codes are widely selected as channel codes to correct transmission errors. MB-OFDM based system widely uses puncture convolutional codes to provide different error correcting capability and data rates. In MB-OFDM system, the input to decoder is soft information from frequency domain analyzer. To achieve different data rates and operate on different distance, 1/3 convolutional codes with k=7 is punctured to provide rate 1/2, 5/8 and 3/4 codes. Due to large variation in data rates and code rate, it is desirable to design a VD to support the highest data rate with high efficiency and reduce power consumption in lower data rates [7].

2.11.7. VOICE-BAND DATA APPLICATION

Convolutional codes also find applications in voice-band data communication systems such as modems used in the general switched telephone network (GSTN). Under bandwidth limited conditions (300-3400 Hz) multilevel coded modulation (MCM) techniques have been used to achieve the required performance without lowering the data rate. During the decade 1984-1994 the international telephone and telegraph consultative committee (ITTCC), later called International Telecommunication Union Telecommunication Standardization Sector (ITU-T) produced three major standards for voice-band data modems. They all featured trellis coded modulation (TCM) combined with non-linear convolutional coding schemes at transmission rates of 9.6, 14.4 and 28.8 Kbit/s respectively

2.12 PUNCTURED CODES

Punctured convolutional codes are derived from ordinary codes by dropping some output bits (i.e. not sending them to a channel). The resulting code has higher rate (less redundancy) but lower correcting capability.

The order of dropping is defined by a puncturing matrix. The number of rows in a puncturing matrix is equal to the number of output polynomials. The elements of this matrix are 1's (which means that we transmit a given bit and 0's which means that we

drop it). The puncturing matrix is applied to the output stream cyclically. The standard puncturing matrices for different rate are shown in table 2.4.

Table 2.4: Puncturing matrices for different rate

Rate	Puncturing Matrix
2/3	10 11
3/4	101 110
5/6	10101 11010
7/8	1000101 1111010

For example, consider a puncturing matrix for the rate 2/3. It means that we will transmit only very first output bit from the first polynomial, but every bit from the second polynomial. In order to decode a punctured convolutional code, one need first to insert erasure marks to the places where deleted bits should be. This task is performed by a depuncturer. Then, the Viterbi decoder should ignore the erased bits during branch metric calculation.

2.13 PIPELINING

Pipelining is a technique of decomposing a sequential process into suboperations , with each subprocess being executed in a special dedicated segment that operates concurrently with all other segments In pipelining an operation is divided into segments and each segment can execute its operation concurrently with the other segments. When a segment completes an operation, it passes the result to the next segment in the pipeline and fetches the next operation from the preceding segment. The final results of each operation emerge at the end of the pipeline in rapid succession.

The pipeline technique is widely used to improve the performance of digital circuits. As the number of pipeline stages is increased, the path delays of each stage are decreased and the overall performance of the circuit is improved. An instruction pipeline is a technique used in the design of computers and other digital electronic devices to increase their

instruction throughput (the number of instructions that can be executed in a unit of time).The fundamental idea is to split the processing of a computer instruction into a series of independent steps, with storage at the end of each step. By breaking the logic into smaller pieces and inserting flip flops between the pieces of logic, the delay before the logic gives valid outputs is reduced. When the clock signal arrives, the flip flops take their new value and the logic then requires a period of time to decode the new values. Then the next clock pulse arrives and the flip flops again take their new values, and so on. By breaking the logic into smaller pieces and inserting flip flops between the pieces of logic, the delay before the logic gives valid outputs is reduced

2.13.1 ADVANTAGES OF PIPELINING

1. The cycle time of the processor is reduced, thus increasing instruction issue-rate in most cases. This speedup is possible because more the number of pipeline stages, more instructions the processor can work on simultaneously and the more instructions it can complete in a given period of time.

2. Some combinational circuits such as adders or multipliers can be made faster by adding more circuitry. If pipelining is used instead, it can save circuitry area.

2.13.2 DISADVANTAGES OF PIPELINING

A non-pipelined processor executes only a single instruction at a time. This prevents branch delays (in effect, every branch is delayed) and problems with serial instructions being executed concurrently. Consequently the design is simpler and cheaper to manufacture.

1. The instruction latency in a non-pipelined processor is slightly lower than in a pipelined equivalent. This is because extra flipflops must be added to the data path of a pipelined processor.
2. A non-pipelined processor will have a stable instruction bandwidth. The performance of a pipelined processor is much harder to predict and may vary more widely between different programs.

2.13.3 STAGES OF PIPELINIG

In general, the speedup in completion rate versus a single-cycle implementation gain from pipelining is ideally equal to the number of pipeline stages. A three-stage pipeline yields a three-fold speedup in the completion rate versus single-cycle, a four-stage pipeline yields a four-fold speedup, a five-stage pipeline yields a five-fold speedup, and so on. Example of three and four stage pipelining is shown in figure 2.15 and 2.16 respectively. This speedup is possible because more the number of pipeline stages , the more instructions the processor can work on simultaneously and the more instructions it can complete in a given period of time.

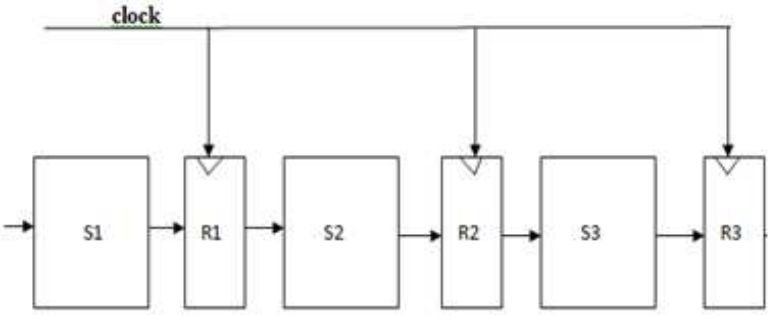

Figure 2.15: 3stage pipelining[8]

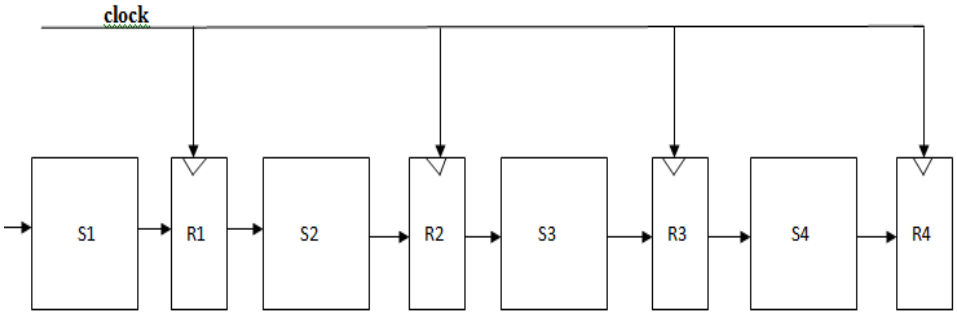

Figure 2.16: 4stage pipelining[8]

CHAPTER 3

FIELD PROGRAMMABLE GATE ARRAY

This chapter introduces about the FPGA concepts and FPGA Synthesis Flow. An FPGA is a device that consists of thousands or even millions of transistors connected to perform logic functions. They perform functions from simple addition and subtraction to complex digital filtering and error detection and correction.

3.1 INTRODUCTION TO FPGA

A field programmable gate array (FPGA) is a semiconductor device that can be configured by the customer or the designer after manufacturing hence the name "field-programmable". Field Programmable gate arrays (FPGAs) are truly revolutionary devices that blend the benefits of both hardware and software. FPGAs are programmed using a logic circuit diagram or a source code in Hardware Description Language (HDL) to specify how the chip will work. They can be used to implement any logical function that an Application Specific Integrated Circuit (ASIC) could perform but the ability to update the functionality after shipping offers advantages for many applications. FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together" somewhat like a one chip programmable breadboard. Logic blocks can be configured to perform complex combinational functions or merely simple logic gates like AND and XOR. In most FPGAs, the logic block also includes memory elements, which may be simple flip flops or more complete blocks of memory.

FPGAs blend the benefits of both hardware and software. They implement circuits just like hardware performing huge power, area and performance benefits over softwares, yet can be reprogrammed cheaply and easily to implement a wide range of tasks. Just like computer hardware, FPGAs implement computations spatially, simultaneously computing millions of operations in resources distributed across a silicon chip. Such systems can be hundreds of times faster than microprocessor-based designs. However unlike in ASICs, these computations are programmed into a chip, not permanently frozen by the manufacturing process. This means that an FPGA based system can be programmed and reprogrammed many times. FPGAs are being

incorporated as central processing elements in many applications such as consumer electronics, automotive, image/video processing military/aerospace, base stations, networking/communications, super computing and wireless applications.

3.2 FPGA TECHNOLOGY TRENDS

- General trend is bigger and faster.
- This is being achieved by increases in device density through even smaller fabrication process technology.
- New generations of FPGAs are geared towards implementing entire systems on a single device.
- Features such as RAM, dedicated arithmetic hardware, clock management and transceivers are available in addition to the main programmable logic.
- FPGAs are also available with the embedded processors (embedded in silicon or as cores within the programmable logic fabric).

3.3 XILINX SPECIFICS

All Xilinx FPGAs contain the same basic resources like

- Configurable logic blocks (CLBs).
- Input/output blocks (IOBs).
- RAM blocks.
- Programmable Interconnections (PIs).
- Other resources like three-state buffers, clock buffers, boundary scan logic, and so on.

3.3.1 CONFIGURABLE LOGIC BLOCKS

The basic building block of Xilinx CLBs is the slice. Spartan III hold four slices per CLB. Each slice contains two 4-input function generators (F/G), carry logic, and two storage elements. Each function generator output drives both the CLB output and the D-input of a flip-flop. Besides the four basic function generators, the Spartan III CLB contains logic that combines function generators to provide functions of five or six inputs. The look-up tables and storage elements of the CLB have the following characteristics:

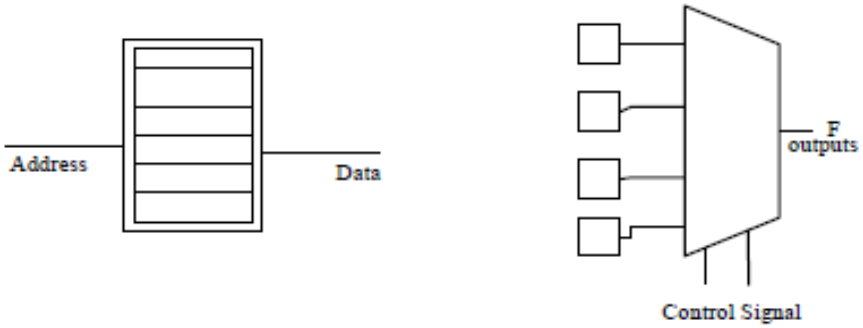

Figure 3.1**:** Look-up table implemented as (a) Memory (b) Multiplexers and Memory [21]

3.3.1.1 Look-Up Tables

The way logic functions are implemented in a FPGA is another key feature. Logic blocks that carry out logical functions are look-up tables, implemented as memory, or multiplexer and memory. Figure 4.1 shows these alternatives, together with an example of memory contents for some basic operations. A 2^n*1 ROM can implement any n-bit function. Typical sizes for n are 2, 3, 4, or 5. In figure 3.1(a), an n-bit LUT is implemented as a 2^n*1 memory; the input address selects one of 2n memory locations. The memory locations are normally loaded with values from the user's configuration bitstream. In figure 3.1(b), the multiplexer control inputs are the LUT inputs. The result is a general-purpose "logic gate." An n-LUT can implement any n-bit function.

3.3.1.2 Storage Elements

The storage elements in a slice can be configured either a edge-triggered D-type flipflops or as level-sensitive latches. The D-input can be driven either by the function generators within the slice or directly from the slice inputs, bypassing the function generators. As well as clock and clock enable signals, each slice has also synchronous set and reset signals.

3.3.2 INPUT/OUTPUT BLOCKS

The Xilinx IOB includes inputs and outputs that support a wide variety of I/O signaling standards. The IOB storage elements act either as D-type flip-flops or as latches. For each flip-flop, the set/reset (SR) signals can be independently configured as synchronous set, synchronous reset, asynchronous preset, or asynchronous clear. Pull-up and pull-down resistors and an optional weak-keeper circuit can be attached to each pad. IOBs are programmable and can be categorized as follows:

3.3.2.1 Input Path

A buffer in the IOB input path is routing the input signals either directly to internal logic or through an optional input flip-flop.

3.3.2.2 Output Path

The output path includes a 3-state output buffer that drives the output signal onto the pad. The output signal can be routed to the buffer directly from the internal logic or through an optional IOB output flip-flop. The 3-state control of the output can also be routed directly from the internal logic or through a flip-flop that provides synchronous enable and disable signals.

3.3.2.3 Bidirectional Block

This can be any combination of input and output configurations.

3.3.3 RAM BLOCKS

Xilinx FPGA incorporates several large RAM memories (block select RAM). These memory blocks are organized in columns along the chip. The number of blocks ranging from 8 up to more than 100, depending on the device size and family.

3.3.4 PROGRAMMABLE ROUTING

Adjacent to each CLB stands a general routing matrix (GRM). The GRM is a switch matrix through which resources are connected, the GRM is also the means by which the CLB gains access to the general-purpose routing. Horizontal and vertical routing resources for each row or column include:

- Long Lines : Bidirectional wires that distribute signals across the device. Vertical and horizontal long lines span the full height and width of the device.
- Hex Lines: Route signals to every third or sixth block away in all four directions.
- Double Lines: Route signals to every first or second block away in all four directions.
- Direct Lines: Route signals to neighbouring blocks vertically, horizontally & diagonally.
- Fast Lines: Internal CLB local interconnections from LUT outputs to LUT inputs.

3.4 FPGA IMPLEMENTATION

The FPGA that is used for the implementation of the circuit is the Xilinx Spartan 3E (Family), XC3S5000 (Device). The working environment/tool for the design is the Xilinx ISE 8.2i is used for FPGA Design flow of VHDL code.

OVERVIEW OF FPGA DESIGN FLOW

As the FPGA architecture evolves and its complexity increases. Today, most FPGA vendors provide a fairly complete set of design tools that allows automatic synthesis and compilation from design specifications in hardware specification languages, such as Verilog or VHDL, all the way down to a bit stream to program FPGA chips. A typical FPGA design flow includes the steps and components shown in figure 3.2 Inputs to the design flow typically include the HDL specification of the design, design constraints, and specification of target FPGA devices [17]. We further elaborate on these components of the design input in the following: Design constraints typically include the expected operating frequencies of different clocks, the delay bounds of the signal path delays from input pads to output pads (I/O delay), from the input pads to registers (setup time), and from registers to output pads (clock-to-output delay). In some cases, delays between some specific pairs of registers may be constrained. The second design input component is the choice of FPGA device. Each FPGA vendor typically provides a wide range of FPGA devices, with different performance, cost, and power tradeoffs. The selection of target device may be an iterative process. The designer may start with a small (low capacity) device with a nominal speed-grade. But, if synthesis effort fails to map the design into the target device, the designer has to upgrade to a high-capacity device. Similarly, if the synthesis result fails to meet the operating frequency, he has to upgrade to a device with higher speed-grade. In both the cases, the cost of the FPGA device will increase in some cases by 50% or even by 100%. This clearly underscores the need to have better synthesis tools since their quality directly impacts the performance and cost of FPGA[17].

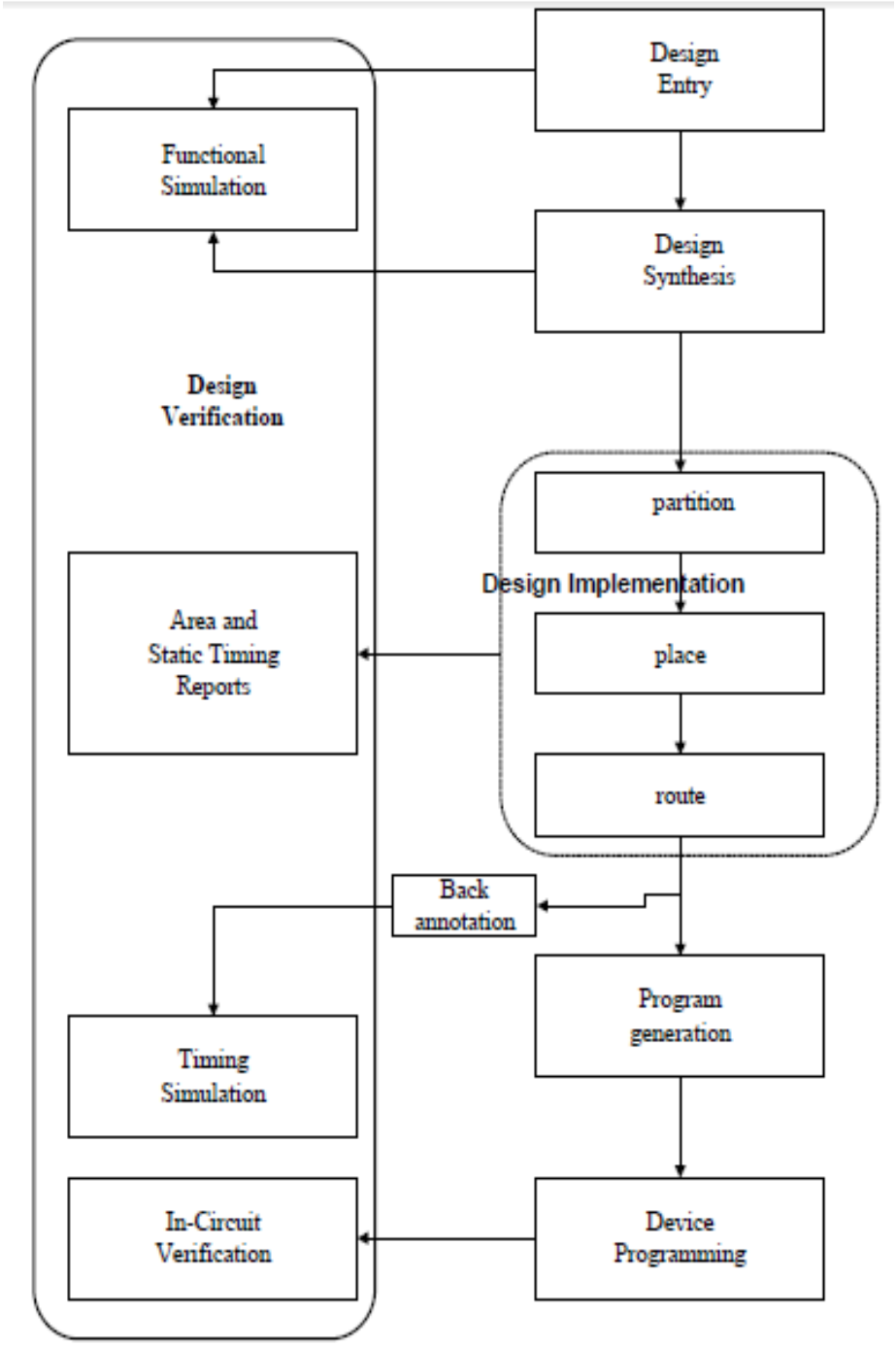

Figure 3.2: FPGA Design Flow[18]

3.4.1 DESIGN ENTRY

The basic architecture of the system is designed in this step which is coded in a Hardware description Language like Verilog or VHDL. A design module is split into two parts, each of which is called a design unit in Verilog. The module declaration

represents the external interface to the design module. The module internals represents the internal description of the design module-its behavior, its structure, or a mixture of both.

3.4.2 BEHAVIORAL SIMULATION

After the design phase, create a test bench waveform containing input stimulus to verify the functionality of the verilog code module using a simulation software i.e. Modelsim SE for different inputs to generate outputs and if it verifies then proceed further, otherwise modifications and necessary corrections will be done in the HDL code. This is called as the behavioral simulation.

3.4.3 DESIGN SYNTHESIS

After the correct simulations results, the design is then synthesized. During synthesis, the Xilinx ISE tool does the following operations:

- HDL Compilation: The tool compiles all the sub-modules in the main module if any and then checks the syntax of the code written for the design.
- Design Hierarchy Analysis: analysis of the hierarchy of the design.
- HDL Synthesis: The process which translates VHDL or Verilog code into a device netlist formate, i.e. a complete circuit with logical elements such as Multiplexer, Adder/subtractors, counters, registers, flip flops Latches, Comparators, XORs, tristate buffers, decoders, etc. for the design. If the design contains more than one sub designs, ex. to implement a processor, we need a CPU as one design element and RAM as another and so on, and then the synthesis process generates netlist for each design element. Synthesis process will check code syntax and analyze the hierarchy of the design which ensures that the design is optimized for the design architecture, the designer has selected. The resulting netlist is saved to an NGC (Native Generic Circuit) file (for Xilinx® Synthesis Technology (XST)).
- Advanced HDL Synthesis(Low Level synthesis): The blocks synthesized in the HDL synthesis and the Advanced HDL synthesis are further defined in terms of the low level blocks such as buffers, lookup tables. It also optimizes the logic entities in the design by eliminating the redundant logic, if any. The tool then generates a 'netlist' file (NGC file) and then optimizes it. The final netlist

output file has an extension of .ngc. This NGC file contains both the design data and the constraints. The optimization goal can be pre-specified to be the faster speed of operation or the minimum area of implementation before running this process. The level optimization effort can also be specified. The higher the effort, the more optimized is the design but higher effort can also be specified. The higher the effort, the more optimized is the design but higher effort requires larger CPU time (i.e. the design time) because multiple optimization algorithms are tried to get the best result for the target architecture.

3.4.4 DESIGN IMPLEMENTATION

Design implementation process consists of the following sub processes

3.4.4.1 Translation**:** The Translate process combines all the input netlists and constraints to a logic design file. This information is saved an a NGD (Native Generic Database) file. This can be done using NGD Build program and .ngd file describes the logical design reduced to the Xilinx device primitive cells. Here, defining constraints is nothing but, assigning the ports in the design to the physical elements (ex. pins, switches, buttons etc) of the targeted device and specifying time requirements of the design. This information is stored in a file named UCF (User Constraints File). Tools used to create or modify the UCF are PACE, Constraint Editor Etc.

3.4.4.2 Mapping**:** The Map process is run after the Translate process is complete. Map process divides the whole circuit with logical elements into sub blocks such that they can be fit into the FPGA logic blocks. That means map process fits the logic defined by the NGD file into the targeted FPGA elements (Combinational Logic Blocks (CLB), Input Output Blocks (lOB)) and generates an NCD (Native Circuit Description) file which physically represents the design mapped to the components of FPGA. MAP program is used for this purpose.

3.4.4.3 Place and Route: Place and Route (PAR) program is used for this process. The place and route process places the sub blocks from the map process into logic blocks according to the constraints and connects the logic blocks. Example if a sub block is placed in a logic block which is very near to IO pin, then it may save the time but it may affect some other constraint. So tradeoff between all the constraints is taken

account by the place and route process .The PAR tool takes the mapped NCD file as input and produces a completely routed NCD file as output. Output NCD file consist the routing information.

3.4.4.4 Bitstream Generation: The collection of binary data used to program the reconfigurable logic device is most commonly referred to as a "bitstream," although this is somewhat misleading because the data are no more bit oriented than that of an instruction set processor and there is generally no "streaming." While in an instruction set processor the configuration data are in fact continuously streamed into the internal units, they are typically loaded into the reconfigurable logic device only once during an initial setup phase.

3.3.4.5 Functional Simulation: Post-Translate (functional) simulation can be performed prior to mapping of the design. This simulation process allows the user to verify that the design has been synthesized correctly and any differences due to the lower level of abstraction can be identified.

3.3.4.6 Static time Analysis :three types of Static time Analysis performed are:

(i) Post-Fit Static Timing Analysis : The timing results of the Post-fit process can be analyzed. The Analyze Post-Fit Static timing process opens the timing Analyzer window, which interactively select timing paths in design for tracing.

(ii) Post-Map Static Timing Analysis: Analyze the timing results of the Map process. Post Map timing reports can be very useful in evaluating timing performance. Although route delays are not accounted for the logic delays can provide valuable information about the design. If logic delays account for a significant portion (>50%) of the total allowable delay of a path, the path may not be able to meet your timing requirements when routing delays are added. Routing delays typically account for 45% to 65% of the total path delays. By identifying problem paths, redesign the logic paths to use fewer levels of logic, tag the paths for specialized routing resources, move to a faster device, or allocate more time for the path. If logic-only-delays account for much less (35%) than the total allowable delay for a path or timing constraint, then the place-and-route software can use very low placement effort levels. In these cases, reducing effort levels allow to decrease runtimes while still meeting performance

requirements.

(iii) Post Place and Route Static Timing Analysis: Analyze the timing results of the Post- Place and Route process. Post-PAR timing reports incorporate all delays to provide a comprehensive timing summary. If a placed and routed design has met all of timing constraints, then proceed by creating configuration data and downloading a device. On the other hand, identify problems and the timing reports, try fixing the problems by increasing the placer effort level, using re-entrant routing, or using multi-pass place and route. Redesign the logic paths to use fewer levels of logic, tag the paths for specialized routing resources, move to a faster device, or allocate more time for the paths.

(iv) Timing Simulation: Perform Post-Place and Route simulation after the design has been and routed. After the design has been through all of the Xilinx implementation tools, a timing simulation netlist can be created. This simulation process allows to see how the design will behave in the circuit. Before performing this simulation it will benefit to create a test bench or test fixture to apply stimulus to the design. After this a .ncd file will be created that is used for the generation of power results of the design.

CHAPTER 4

IMPLEMENTATION OF PROPOSED VITERBI DECODER

4.1 CONVOLUTIONAL ENCODER

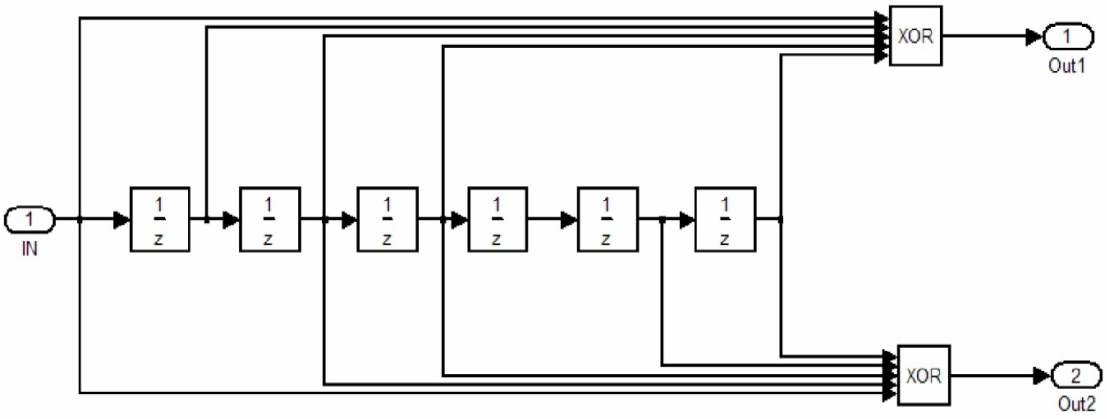

Figure 4.1: Convolutional encoder for proposed Viterbi decoder[24]

The convolutional encoder, which corresponds to our Viterbi decoder considered in this thesis, is shown in the figure 4.1. This encoder in the system full fills IEEE the 802.11a/g and the 802.16 standards. The parameters of the convolutional encoder are shown in the table 4.1.

Table 4.1: Parameters of the convolutional encoder

definition	Symbol	value
input number	K	1
output number	N	2
encoder rate	K/N	1/2
constrain length	L	7
generator sequence	G	[1 1 1 1 0 0 1; 1 0 1 1 0 1 1]

From the following state number formula, we can compute, there are 64 initial states of convolutional encoder $N=2^{(L-1)*K}$

4.2 DE-MODULATOR

In this block, the noise analog signal was received. By the real number value of the signal, we can simulate the 3 bit soft decision. The 3 bit 8-level soft decision is shown in figure 4.2.

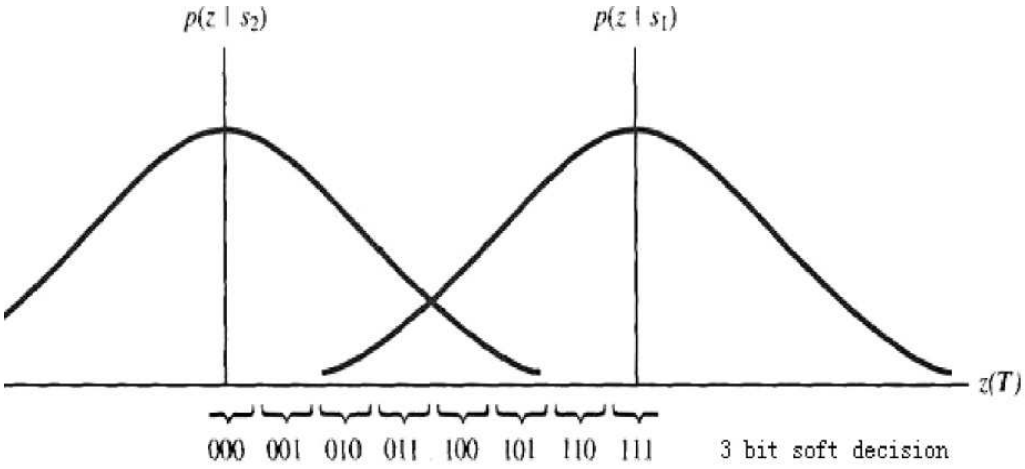

Figure 4.2: 3-bit soft decisions[20]

Table 4.2: Interpretations of eight possible input values[20]

Decision Value	**Interpretation**
0	Most confident 0
1	Second most confident 0
2	Third most confident 0
3	Least confident 0
4	Least confident 1
5	Third most confident 1
6	Second most confident 1
7	Most confident 1

The “000” and “111” presents the most confident ‘0’ and ‘1’. On the contrary, the “011” and “110” presents the most uncertain ‘0’ and ‘1’. Since the soft decision will provide more information than the hard decision, it will prove 2-3 db performance improvement for whole system .The table 4.2 lists the interpretations of the eight possible input values for this example.

4.3 THE NEXT STATE ROM:

When the reset signal is set high, Viterbi decoder needs to initialize the Next state Rom. The architecture of the Next state ROM is shown in the figure 4.3.

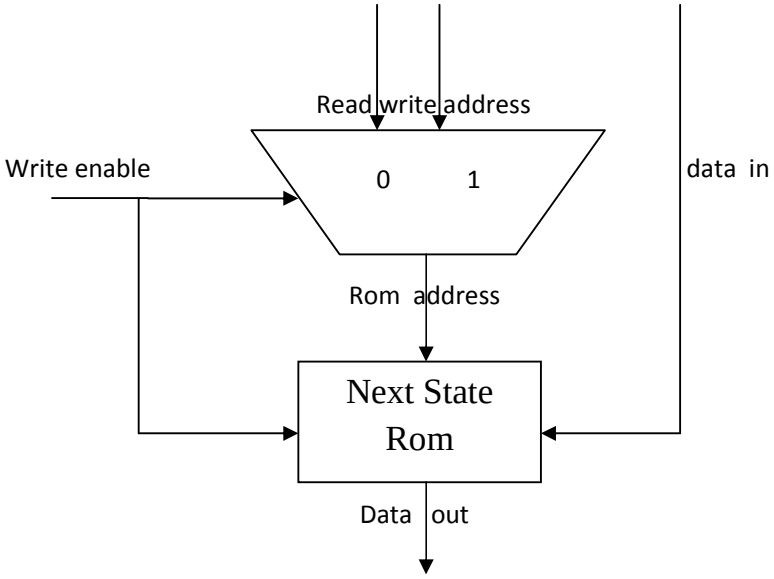

Figure 4.3: The Next State ROM hardware implementation[23]

With this metric, the decoder can check both the next state (the first 6 bits) of each state and the two output bits (the last two bits) with current state and input bit.
The contexts of the ROM is shown in the table 4.3:

Table 4.3: Contexts of the ROM

Previous state	**Next state and 2output-bit With Input 0**	**Next state and 2output-bit With Input 1**	**Previous state**	**Next state and 2output-bit With Input 0**	**Next state and 2output-bit With Input 1**
000000	00000000	10000011	100000	01000010	11000001
000001	00000011	10000000	100001	01000001	11000010
000010	00000101	10000110	100010	01000111	11000100
000011	00000110	10000101	100011	01000100	11000111
000100	00001000	10001011	100100	01001010	11001001
000101	00001011	10001000	100101	01001001	11001010
000110	00001101	10001110	100110	01001111	11001100
000111	00001110	10001101	100111	01001100	11001111
001000	00010011	10010000	101000	01010001	11010010
001001	00010000	10010011	101001	01010010	11010001
001010	00010110	10010101	101010	01010100	11010111
001011	00010101	10010110	101011	01010111	11010100
001100	00011011	10011000	101100	01011001	11011010
001101	00011000	10011011	101101	01011010	11011001
001110	00011110	10011101	101110	01011100	11011111
001111	00011101	10011110	101111	01011111	11011100
010000	00100011	10100000	110000	01100001	11100010
010001	00100000	10100011	110001	01100010	11100001
010010	00100110	10100101	110010	01100100	11100111
010011	00100101	10100110	110011	01100111	11100100
010100	00101011	10101000	110100	01101001	11101010
010101	00101000	10101011	110101	01101010	11101001
010110	00101110	10101101	110110	01101100	11101111
010111	00101101	10101110	110111	01101111	11101100
011000	00110000	10110011	111000	01110010	11110001
011001	00110011	10110000	111001	01110001	11110010
011010	00110101	10110110	111010	01110111	11110100

011011	00110110	10110101	111011	01110100	11110111
011100	00111000	10111011	111100	01111010	11111001
011101	00111011	10111000	111101	01111001	11111010
. 011110	00111101	10111110	111110	01111111	11111110 .
011111	00111110	10111101	111111	01111100	11111111

4.4 UNPIPELINED VITERBI DECODER

4.4.1 BMU BLOCK:

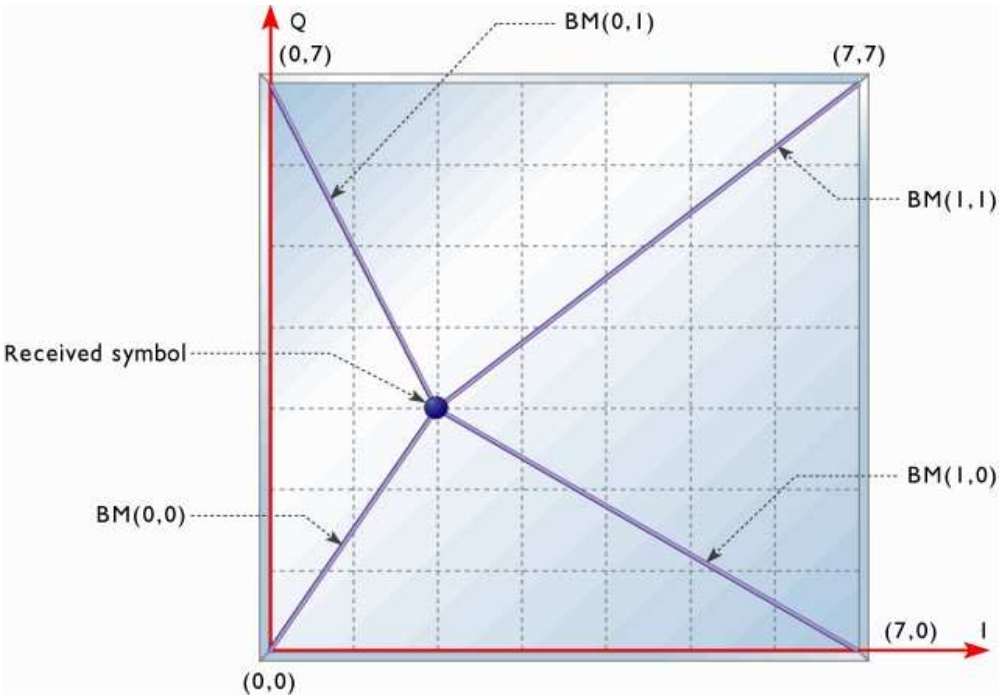

Figure 4.4: Branch matrix generator for a Viterbi decoder[20]

The Euclidean distance between ideal code words and an arbitrary point in this space (the received code word) gives the branch metrics. By increasing the number of soft bits the number of possible values for branch metrics also increases. Figure 4.4 shows the branch metric diagram for an 8-level decoder. The branch metric uses the Hamming distance for the four possible paths. First, we initialize four different received hamming distance lookup table. Then at each time instant, we check the input symbol, we get the four

possible distances[20]. The BMU perform simple check and select operations on the decision bits to generate the output. The hardware implementation is shown in the figure 4.5.

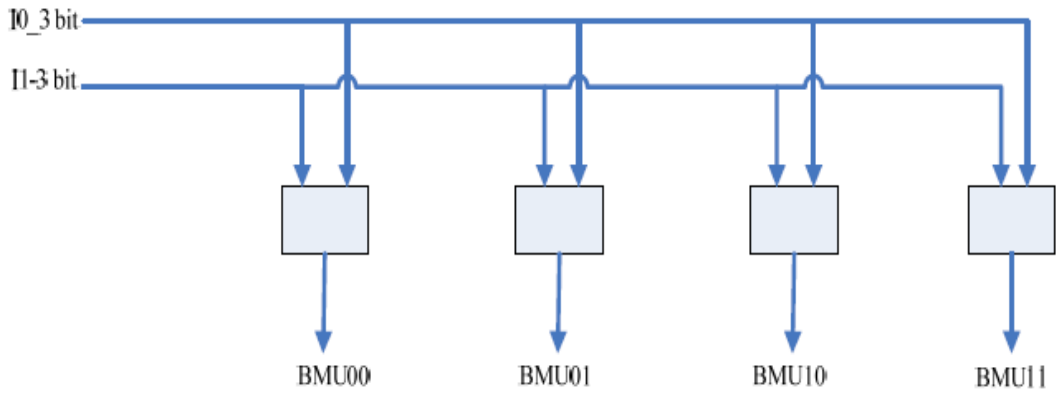

Figure 4.5: BMU block[21]

4.4.2 THE ACS BLOCK

When the 4 possible input distance is ready, the ACS block, butterfly module adds the results of previous state metric and the related distance value to get two paths for each 64 states. The butterfly module is shown in the figure 4.6. We have total 32 butterflies. Each butterfly computes 4 possible paths and selects the two smaller distance paths.

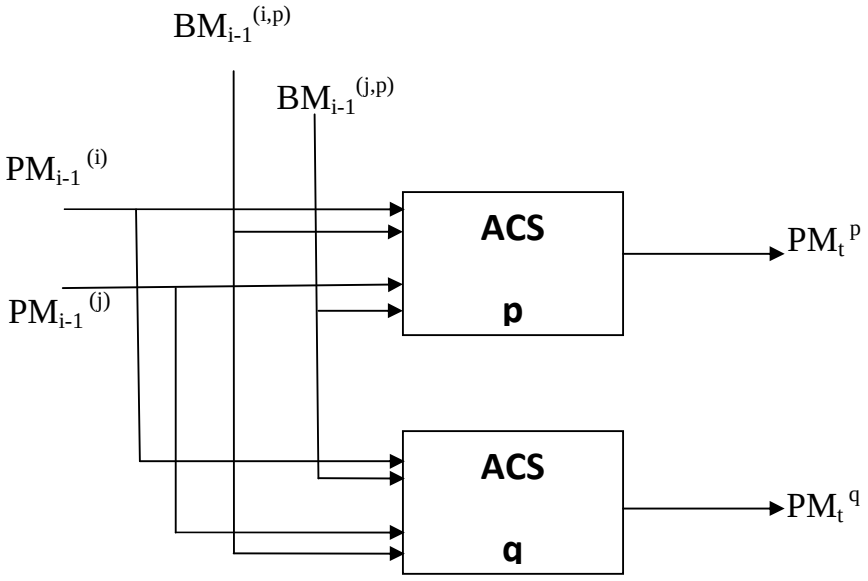

Figure 4.6: Single butterfly module[22]

For each nod (state), the ACS module selects a smaller one as the survival path and stores them to the accumulated state metric storage block and the survivor path metric. The ACS module is shown in figure 4.7. The decoder continues in this way to advance deeper into the trellis and to make decisions on the input data bits by eliminating all paths but one.

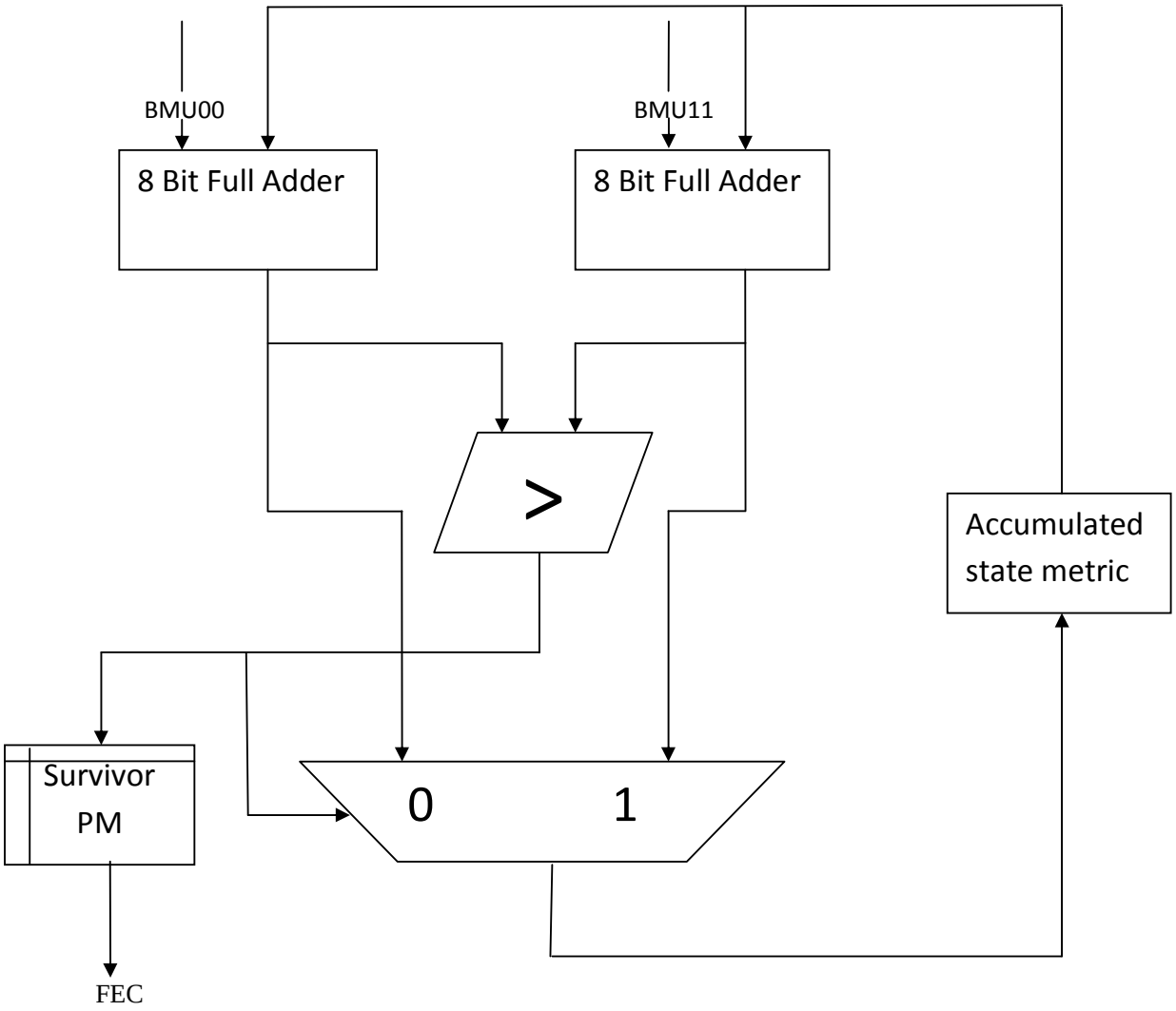

Figure 4.7: The ACS Module[23]

4.4.3 THE TRACE-BACK BLOCK

When the trellis diagram is finished, the trace-back module will search the Maximum Likelihood (ML) path from the final state which is state0 to the beginning state which is also state0. Each time, the trace-back block just left shifts one bit of the binary state number and add one bit from the survivor path metric to compute the previous state. By

doing this, the most likelihood path is found. The trace-back module is shown in figure 4.8.

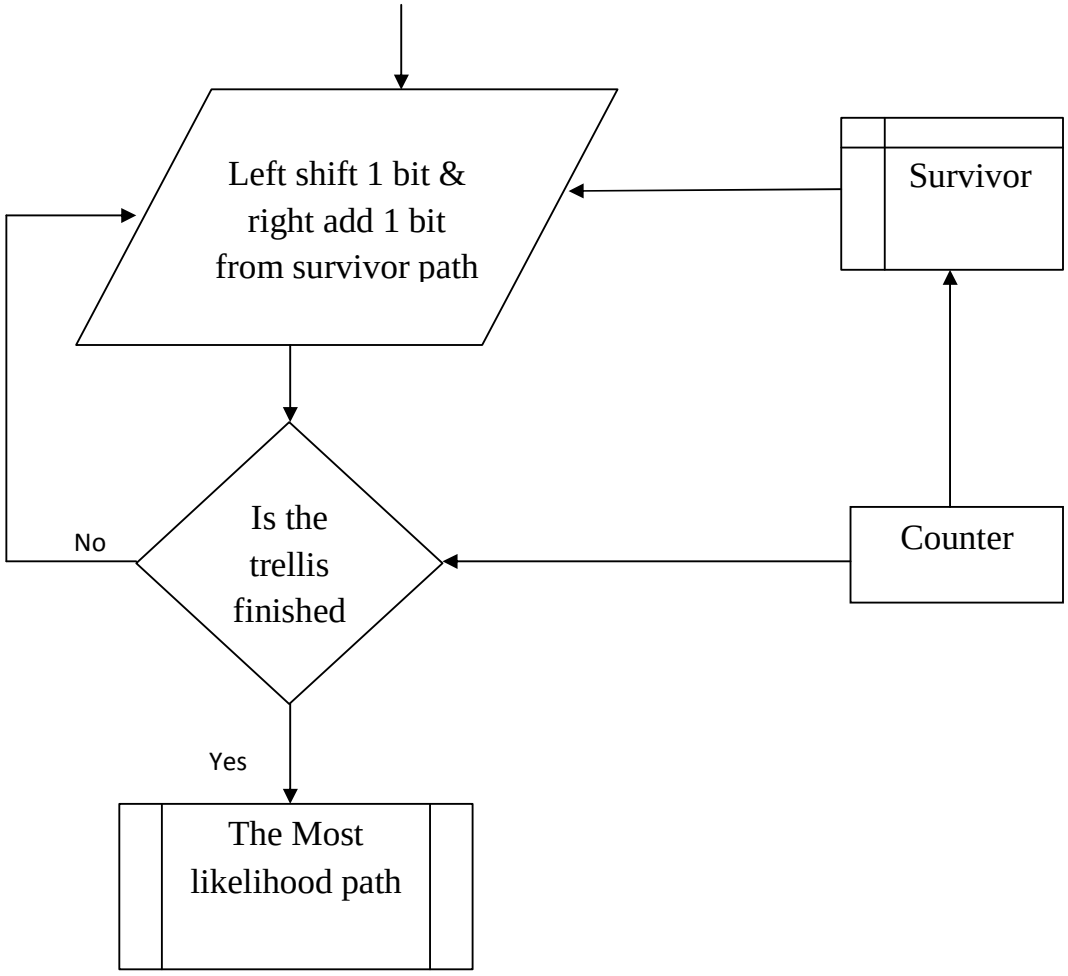

Figure 4.8: The trace-back block[23]

4.4.4 THE DECODING BLOCK

After finding the most likelihood path, the last step is to decode the original data. The decoding data block is shown in figure 4.9. First step begins with time sample 0 at the state0. Each time, by checking the next state in ML path and comparing it with the corresponding state in the Next state table ROM, we could find the input data, one or zero. Hence ,the Maximum likelihood path is decoded.

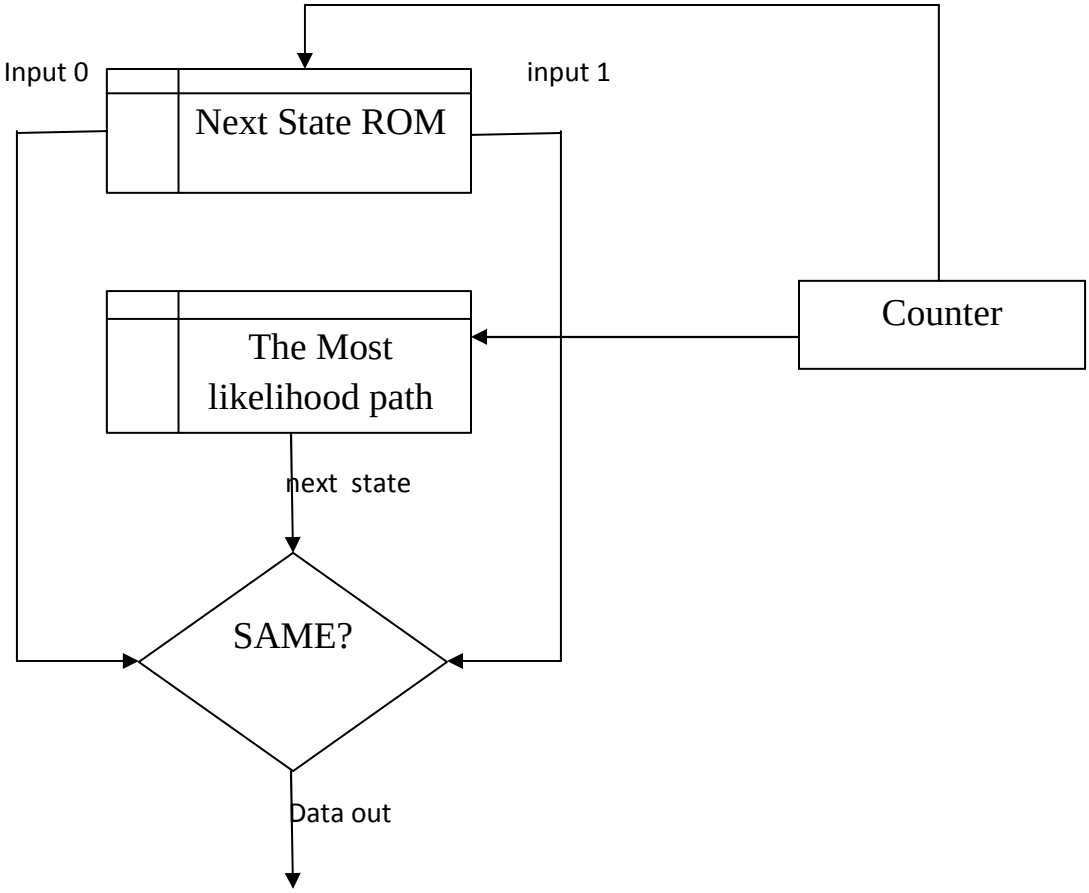

Figure 4.9: The decode data block[23]

Simulation results of 3-bit soft decision viterbi decoder is shown in the figure 4.10 and synthesis results in table 4.4:

Data : input 35-bit data

Clk : input clock

Output : output 35-bit data

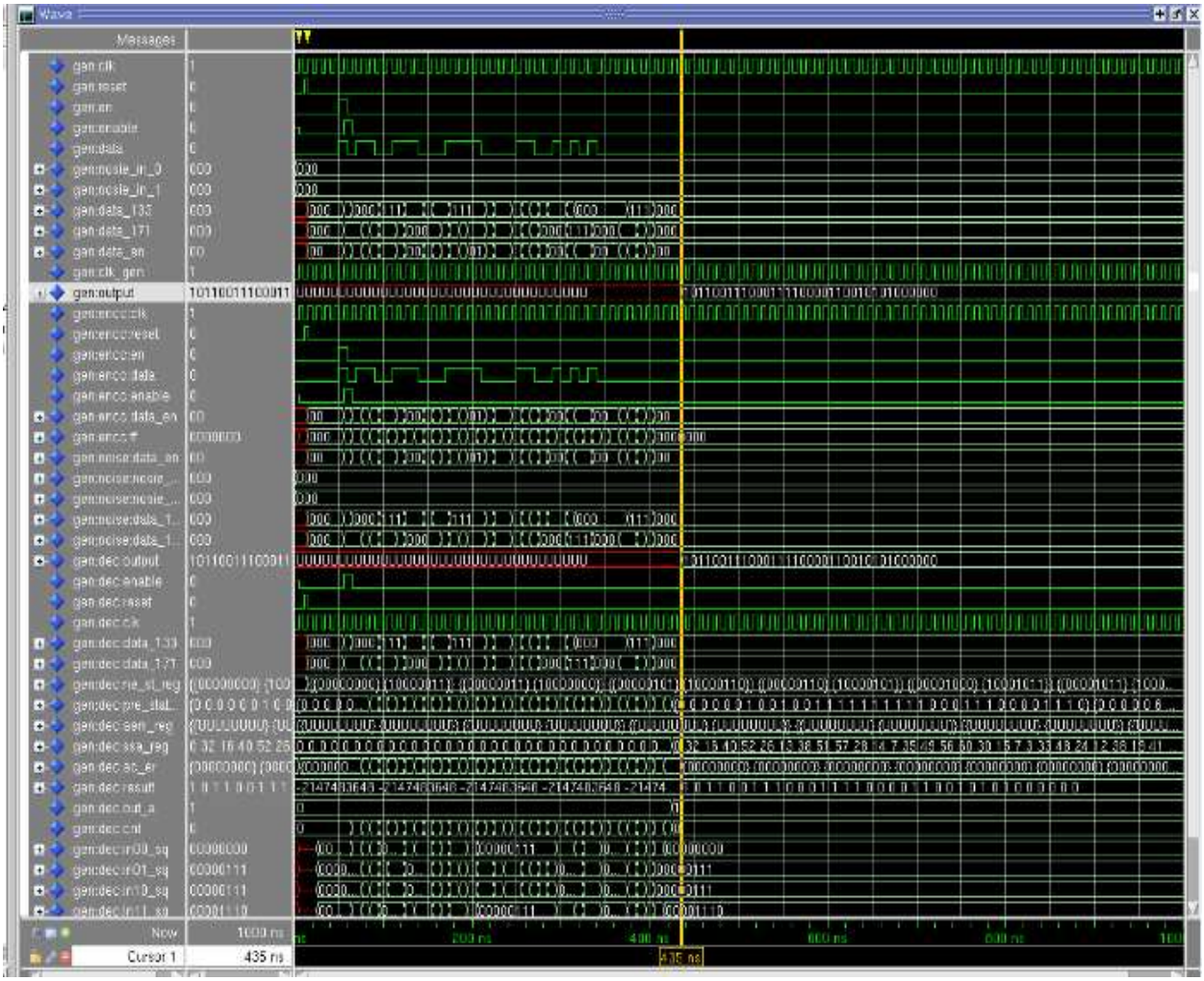

Figure 4.10: Simulation result of 3-bit soft decision viterbi decoder

Table 4.4: Synthesis Report of 3-bit soft decision viterbi decoder

	Spartan 3E xc3s500E
No. of Slices	3556/4656 (76%)
No. of LUTs	5539/9312 (59%)
Minimum Period	135.902ns
Maximum Frequency	7.358MHz

4.5 PIPELINED VITERBI DECODER

The highest possible implementation efficiency is usually achieved if the critical path is shortened. For purely feed-forward structures this can be easily accomplished by inserting extra pipeline registers into the critical path. . In this section, we pipeline the VD into three blocks at first and analyze the gate delay for each pipelined block[8].

The datapath of the Viterbi decoder has been partitioned into **3** pipeline stages by introducing latches along the datapath as shown in figure 4.11[13]. Clock period can be reduced as long as data from a particular clock cycle does not overwrite data from the previous clock cycle and hence increasing the frequency. Reduced clock period can be observed from Synthesis Report as shown in table 4.5.

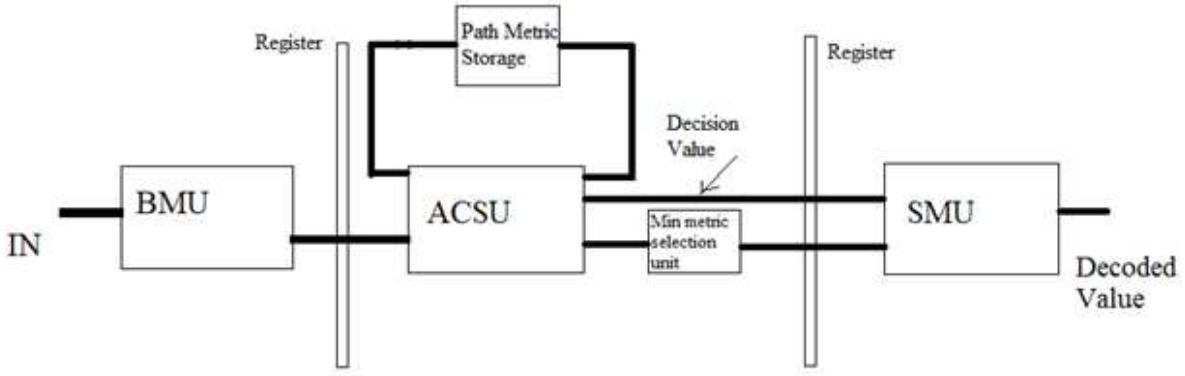

Figure 4.11: Datapath of pipelined viterbi decoder[13]

Table 4.5: Synthesis Report of 3-bit soft decision pipelined viterbi decoder

	Spartan 3E xc3s500E
No. of Slices	4273/4656 (91%)
No. of LUTs	6895/9312 (74%)
Minimum Period	13.725ns
Maximum Frequency	72.859MHz

Traditionally, we can find that the decoding speed of pipelined VD is dominated by its ACS recursion. In other words, breaking the ACS recursion is necessary to improve the decoding speed of VD. Therefore ,we introduce a latch in feedback loop as shown in figure 4.12 which reduces the delay and hence further increases the frequency to `144.341MHz` as shown in synthesis reports of table 4.6 .

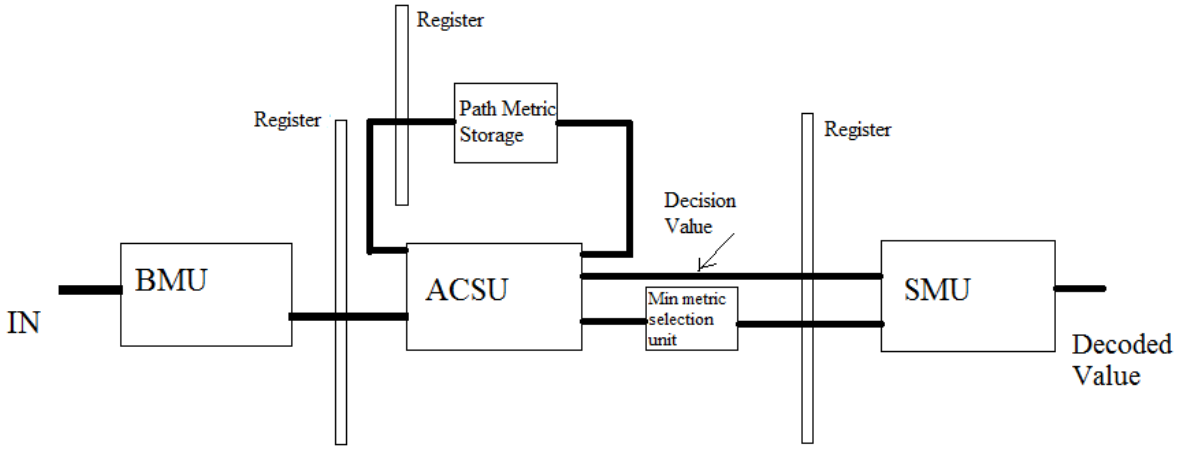

Figure 4.12: Datapath of proposed viterbi decoder

Table 4.6: Synthesis Report of 3-bit soft decision proposed viterbi decoder

	Spartan 3E xc3s500E
No. of Slices	`3214/4656 (69%)`
No. of LUTs	`4838/9312 (51%)`
Minimum Period	`6.928ns`
Maximum Frequency	`144.341MHz`

RTL view of convolutional encoder of proposed architecture is shown in figure 4.13 .

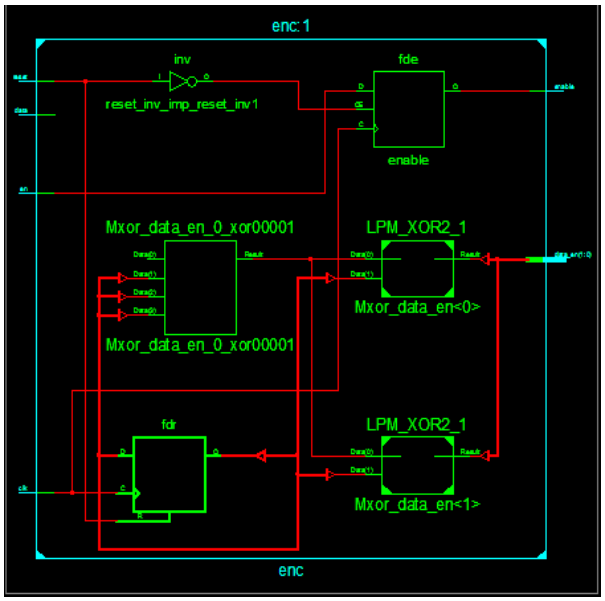

Figure 4.13: Rtl view of convolutional encoder

RTL view of 3-bit soft decision demodulator is shown in figure 4.14 .

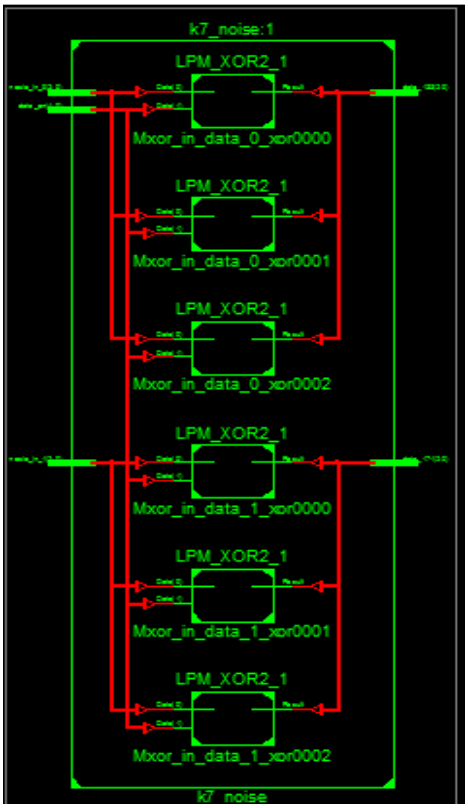

Figure 4.14: Rtl view of 3-bit soft decision demodulator

Simulation waveforms of proposed viterbi decoder is shown as following in figure 4.15:

Data : input 35-bit data

Clk : input clock

Output : output 35-bit data

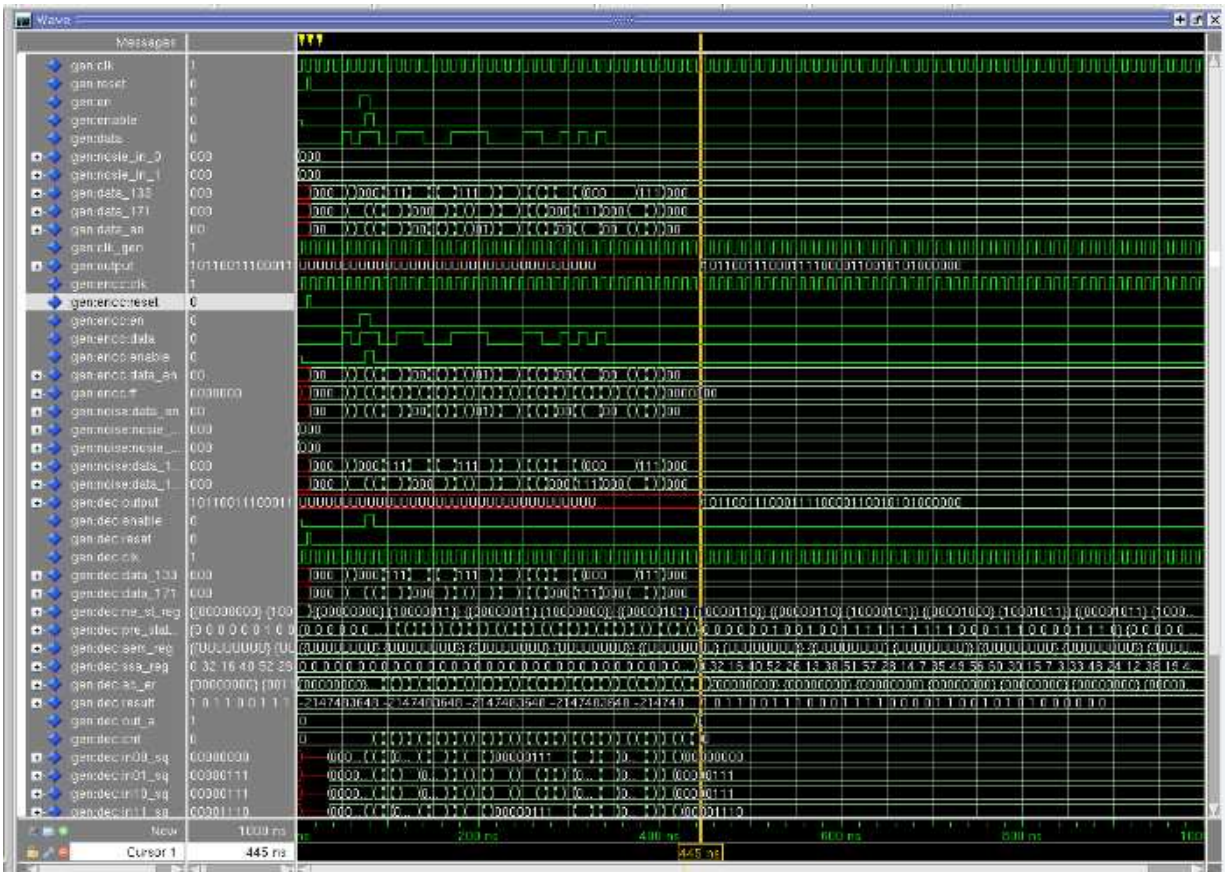

Figure 4.15: Simulation result of proposed viterbi decoder

Puncturing technique is implemented in matlab and plots of BER with respect to E/N varying from 1db to 7db with different code rates are obtained. The different code rates used are 1/2, 2/3 and 3/4. As we increase the code rate, there is an increase in throughput but bit error rate also increases.

Plots of BER with E/N varying from 1db to 7db with different code rates are shown in figures from 4.16 to 4.19.

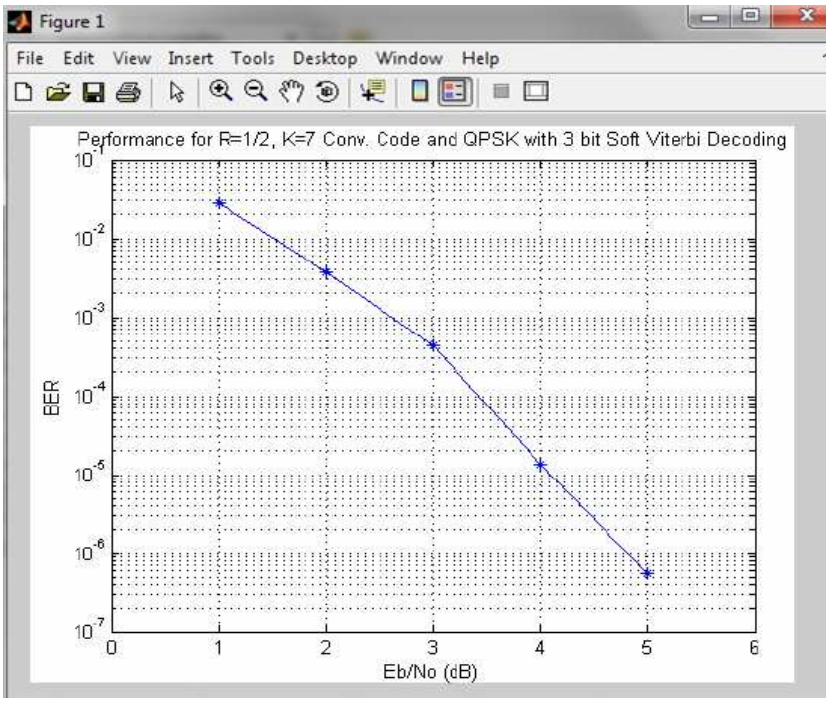

Figure 4.16: BER Performance with rate ½

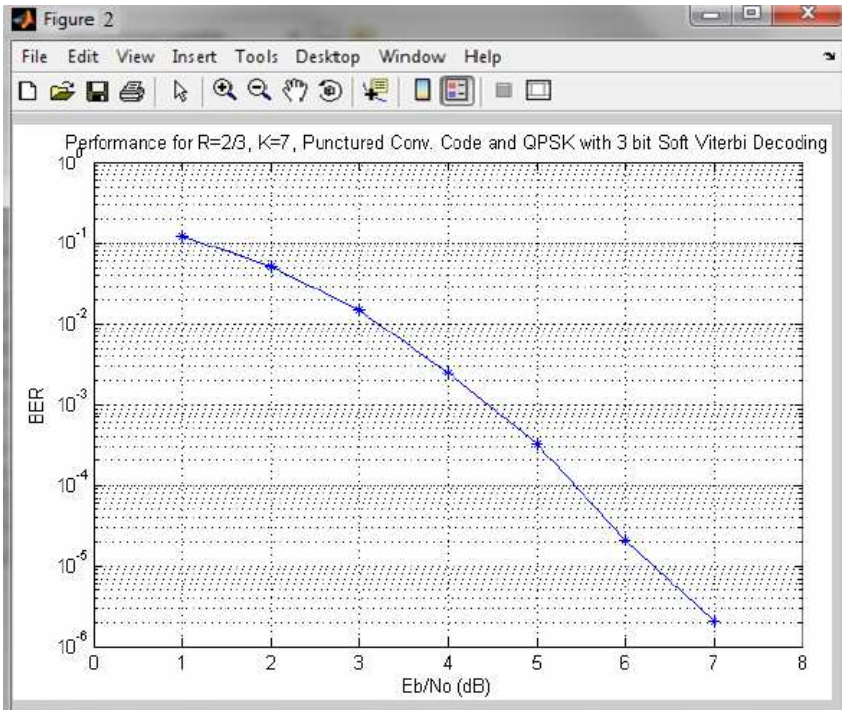

Figure 4.17: BER Performance with rate 2/3

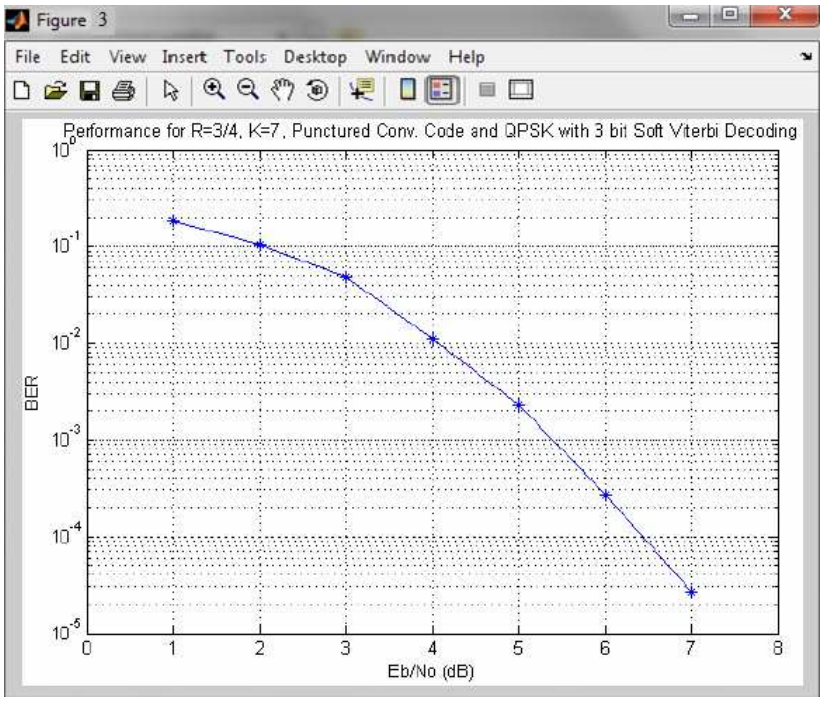

Figure 4.18: BER Performance with rate 3/4

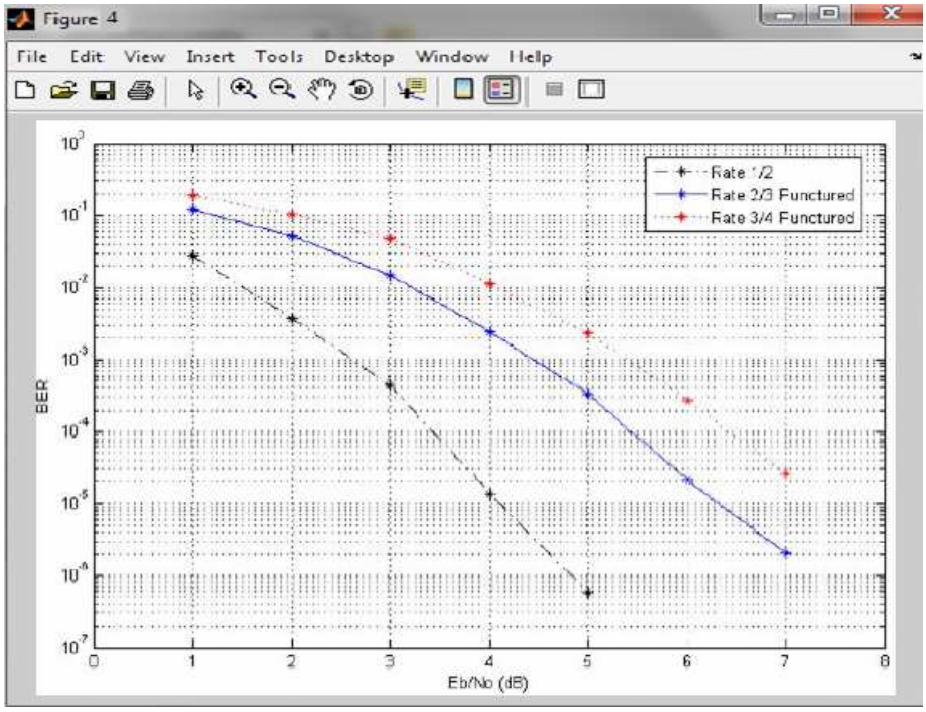

Figure 4.19: Comparison of BER performance of 3 rates

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

Viterbi decoder allows safe data transmission via error correction and original message can be recovered accurately. The main consideration of this thesis is to increase the speed of viterbi decoder. Pipelined structure is used to balance the delay time of each unit in order to realize the high speed operation. The design of various viterbi decoders with constraint length 7 and code rate 1/2 have been successfully implemented on Spartan 3E xc3s500E device. Puncturing technique is implemented in matlab and plots of BER with different transmission rates are obtained.

Frequency of 3-bit soft decision viterbi decoder obtained is 7.358MHz with delay of 135.902ns. Clock period has been reduced by introducing latches between the data path of viterbi decoder increasing the throughput of viterbi decoder. Frequency of this pipelined viterbi decoder obtained is 72.859MHz .

It has been found that the decoding speed of pipelined VD is dominated by its ACS recursion i.e. its feedback loop. Therefore, a latch has been introduced in feedback loop in order to break the ACS recursion which reduces the delay and hence further increases the frequency to 144.341MHz .

Table 5.1: Synthesis results of Spartan 3E xc3s500E device

	3-bit soft decision viterbi decoder	**Pipelined viterbi decoder**	**Proposed viterbi decoder**
Maximum Frequency	7.358MHz	72.859MHz	144.341MHz
Minimum period	135.902ns	13.725ns	6.928ns

FUTURE SCOPE

In the future, bit level pipelined Viterbi decoder for high-performance UWB applications can be implemented. Another possibility can be to implement a Viterbi decoder in the Turbo code algorithm. With different specification, there must be a modification in the original design. Trellis decoding could involve further improvements for trellis and survivor path processing blocks, which solely depend on the radix of the trellis.

REFERENCES

[1] Vasily P. Pribylov, Alexander I. Plyasunov, “A Convolutional Code Decoder Design using Viterbi Algorithm with Register Exchange History Unit,” SIBCON, IEEE, 2005.

[2] John G. Proakis, “Digital Communication,” McGraw Hill, Singapore. Pp 502-507,471-475 ,2001.

[3] S. K. Hasnain, Azam Beg and S. M. Ghazanfar Monir, “Performance Analysis of Viterbi Decoder Using a DSP Technique,” INMIC, IEEE, pp 201-206 ,2004.

[4] Irwin M. Jacobs, “Practical Applications of Coding,” IEEE, pp 305-310,1974.

[5] Tommy Oberg, “Modulation Detection and Coding,” Wiley and Sons. Pp 81- 86, 2001.

[6] YOU Yu-xin, WANG Jin-xiang, LAI Feng-chang and YE Yi-zheng, “VLSI Design and Implementation of High Speed Viterbi Decoder,” IEEE,pp 64- 66, 2002.

[7] Jun Tang and Keshab K. Parhi , “V.D for High Speed Ultra Wideband Communication Systems,” IEEE,Vol.37,2005.

[8] Computer system architecture 3rd edition, M. Morris Mano.

[9] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, Wiley-Interscience Pub.,1999.

[10] KIM MIM WOO, “High-speed Viterbi Decoder Architecture,” 2005.

[11] Yun-Nan Chang, Hiroshi Suzuki, and Keshab K. Parhi, “A 2-Mb/s 256- State 10-mW Rate-1/3 Viterbi Decoder,” IEEE Journal of Solid-State Circuits, Vol. 35, pp.826-833,2000.

[12] M.kivioja, J.isoaho and L.vanska, “Design and Implementation of Viterbi Decoder with FPGA,” Journal of VLSI Signal Processing, pp.5-14,1999.

[13] Byonghyo Shim and Jung Chul Suh, “Pipelined VLSI Architecture of The Viterbi Decoder for IMT-2000,” IEEE,1999.

[14] Nat. Chiao Tung Univ, Hsinchu Taiwan, “A high-speed low-power pipelined Viterbi Decoder: breaking the ACS-bottleneck,” IEEE ,2010.

[15] Digital communication Fundamentals and Applications Second Edition BERNARD SKLAR .

[16] G. D.Forney Jr. ,“The Viterbi algorithm,” Proc. IEEE, vol. 61,no. 3, pp. 268-278, Mar.1973.

[17] D.Chen. J.Cong, and P. Pan, “FPGA Design Automation: A Survey,” Foundations and Trends in Electronic Design Automation,2006.

[18] Anubhuti Khare, Manish Saxena, Jagdish Patel, “FPGA Based Efficient Implementation of Viterbi Decoder,” International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-1, Issue-1,2011.

[19] H. Lou, “Viterbi Decoder Design for the IS-95 CDMA Forward Link,” AT&T Bell Laboratories. Mountain Ave, Murray Hill,1996.

[20] Abdul-Rafeeq, Abdul-Shakoor and Valek Szwarc, “A high performance soft decision viterbi decoder for wlan and broadband applications,” CCECE/CCGEI, Ottawa, IEEE,2006.

[21] T.M.Chaitanyat, P Cyril Prasanna Raj2, Veena Sanath Kumar, “ASIC Implementation of Soft Decision Viterbi Decoder for GSM Application,” sasTech, Vol. VI, No.1,2007.

[22] Tsung-Sheng Kuo, Chau-Yun Hsu ,”A Butterfly Structure for Rate k/n Convolutional Codes,” IEEE,2005.

[23] Bhupendra Singh1, Sanjeev Agarwal2 and Tarun Varma3 ,”Hardware Implementation of Viterbi Decoder for Wireless Applications,” International Journal of Computer Communication and Information System (IJCCIS)– Vol2. No1. ISSN: 0976–1349,2010.

[24] Ehsan Rohani, S. Mehdi Fakhraie, ”Optimum Design of Viterbi Decoder for High Speed Data Telecommunication Receivers with 802.16a,” Case Study IEEE, 2008.